ФЕДЕРАЛЬНОЕ АГЕНТСТВО
ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р 53556.5— 2013

Звуковое вещание цифровое

КОДИРОВАНИЕ СИГНАЛОВ ЗВУКОВОГО ВЕЩАНИЯ С СОКРАЩЕНИЕМ ИЗБЫТОЧНОСТИ ДЛЯ ПЕРЕДАЧИ ПО ЦИФРОВЫМ КАНАЛАМ СВЯЗИ. ЧАСТЬ III (MPEG-4 AUDIO)

Структурированное кодирование звуковых сигналов (SA)

ISO/IEC 14496-3:2009 (NEQ)

Издание официальное

Москва
Стандартинформ
2014

Предисловие

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

2 ВНЕСЕН Техническим комитетом по стандартизации № 480 «Связь»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 22 ноября 2013 г. № 1785

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)

Содержание

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Звуковое вещание цифровое

КОДИРОВАНИЕ СИГНАЛОВ ЗВУКОВОГО ВЕЩАНИЯ С СОКРАЩЕНИЕМ ИЗБЫТОЧНОСТИ ДЛЯ ПЕРЕДАЧИ ПО ЦИФРОВЫМ КАНАЛАМ СВЯЗИ. ЧАСТЬ III (MPEG-4 AUDIO)

Структурированное кодирование звуковых сигналов (SA)

Sound broadcasting digital. Coding of signals of sound broadcasting with reduction of redundancy for transfer on digital communication channels. A part III (MPEG-4 audio). Structured audio (SA)

Дата введения — 2014—09—01

1 Область применения

1.1 Обзор

1.1.1 Назначение

Комплект инструментальных средств структурированного аудио дает возможность передавать и декодировать синтетические звуковые эффекты и музыку, нормируя несколько различных компонентов. Используя структурированное аудио, может быть создан высококачественный звук при чрезвычайно низкой пропускной способности. Типичная синтетическая музыка для чрезвычайно тонкого выразительного исполнения с использованием многочисленных инструментов может быть кодирована в формате на скоростях передачи в пределах от 0 Кбит/с до 2 или 3 Кбит/с.

MPEG-4 стандартизирует не определенный набор методов синтеза, а метод для описания методов синтеза. Любой современный или будущий метод синтеза звука может быть описан в формате *MPEG*-4 структурированное аудио.

1.1.2 Введение в главные элементы

В комплекте инструментальных средств структурированного аудио есть пять главных элементов:

1. Язык оркестра структурированного аудио или *SAOL*. *SAOL* является языком обработки цифровых сигналов, который позволяет описывать произвольные алгоритмы синтеза и управления как часть потока битов контента. Синтаксис и семантика *SAOL* стандартизируются нормативным способом.

2. Язык оценки структурированного аудио или *SASL*. *SASL* является простым языком оценки и управления, который используется в определенных типах объектов для описания способа, который используют алгоритмы генерации звука описанные в *SAOL*, для воспроизведения звука.

3. Формат банка сэмплов структурированного аудио, или *SASBF*. Формат банка сэмплов позволяет передавать банки аудиосэмплов, которые будут использоваться в табличном синтезе, и описание простых алгоритмов обработки, чтобы пользоваться ими.

4. Нормативное описание планировщика. Планировщик является диспетчерским элементом во время выполнения процесса декодирования структурированного аудио. Он отображает структурное управление звуком, определенное в *SASL* или *MIDI*, и диспетчеризированные события в реальном времени, используя нормативные алгоритмы генерации звука.

5. Нормативная ссылка на стандарты *MIDI*, стандартизированные внешне *MIDI Manufactures Association*. *MIDI* является альтернативным средством структурного управления, которое может использоваться в сочетании с или вместо *SASL*, хотя и менее мощное и гибкое чем *SASL*. Поддержка *MIDI* в этом стандарте предоставляет важную обратную совместимость с существующим контентом и инструментами авторинга. Поддержка *MIDI* в этом стандарте состоит из списка распознаваемых сообщений *MIDI* и нормативной семантики для каждого.

Издание официальное

2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на ГОСТ Р 53556.0—2009 «Звуковое вещание цифровое. Кодирование сигналов звукового вещания с сокращением избыточности для передачи по цифровым каналам связи. Часть III (MPEG-4 *audio*). Основные положения».

3 Термины и определения

В настоящем стандарте применены термины и сокращения с соответствующими определениями, используемые в ГОСТ Р 53556.0—2009.

4 Символы и сокращения

4.1 Математические операции

Используемые математические операторы, подобны используемым в языке программирования *C*:

+	дополнение
-	вычитание
x или *	умножение
/	деление
Exp	экспоненциальная функция exp (основание *e*),
log	натуральный логарифм
log 10	десятичный логарифм
abs	абсолютное значение
floor (*x*)	наибольшее целое число, меньшее чем или равное *x*
cell (*x*)	наименьшее целое число, большее чем или равное *x*
>	больше чем
<	меньше чем
> =	больше чем или равно
<=	меньше чем или равно
<> или !=	не равно

4.2 Методы описания

4.2.1 Синтаксис потока битов

Синтаксис потока битов структурированного аудио описывается, используя *SDL MPEG–4 Sintactic Description Language*.

4.2.2 Синтаксис *SAOL*

Текстовый синтаксис *SAOL* описывается, используя нотацию расширенного формата *Backus-Naur (BNF)*. *BNF* является описанием для контекстно-свободных грамматик языков программирования.

Грамматики *BNF* составляются из терминов, также называемых маркерами (символами). Термины представляют синтаксические элементы языка, такие как ключевые слова и пунктуацию. Правила описывают комбинирование этих элементов в структурные группы.

Правила подстановки, которые отображают продукты в последовательностях других продуктов и терминов, представляются символом - >.

Также описаны правила подстановки, использующие дополнительные элементы, с использованием символов []. Использование этой нотации не увеличивает возможности описания синтаксиса, но делает определенные конструкции более простыми.

Грамматика (начиная с символа *seqhead*) описывает, в дополнение к установленному выше, все строки, начинающиеся с символа *'c'* и с последующей последовательностью *'a's* и *'b's*.

Маркер *NULL* может использоваться, чтобы указать, что последовательность никаких символов (пустая строка) является допустимой перезаписью для определенного продукта.

Другие символы, такие как *ellipsis (...),* будут использоваться тогда, когда их значение будет ясным из контекста.

4.2.3 Синтаксис SASL

Синтаксис *SASL* определяется, используя расширенные грамматики *BNF*.

5 Синтаксис потока битов и семантика

5.1 Введение в синтаксис потока битов

Этот подпункт описывает формат потока битов, определяющий поток битов структурированного аудио *MPEG*–4.

Каждая группа классов записывается с нормативной семантикой, которая определяет значение данных, представленных этими классами.

5.2 Синтаксис потока битов

```
/*****************************
Symbol table definitions
*****************************/
Class symbol {
        unsigned int (16) sym;  //no more than 65535 symbols/orch + score
}
Class sym_name {                //one name in a symbol table
        unsigned int (4) length;        //names up to 15 chars long
        unsigned int (8) name  [length];
}
class symtable {                        //a whole symbol table
        unsigned int (16) length;       //no more than 65535 symbols/orch+score
        sym_name name [length];
}
```

Поток битов может содержать таблицу символов. Таблица символов позволяет текстовому коду *SAOL* и *SASL* быть восстановленным из маркированного представления потока битов. Включение или исключение таблицы символов не влияют на процесс декодирования.

Если таблица символов включена, то все или некоторые из символов в оркестре и таблице должны быть ассоциированы с текстовым именем следующим образом: каждый символ (символ является только целым числом) должен быть связан со строкой символов, соединенной с этим символом в объекте *sym_name*. С данным символом должно быть ассоциировано не более одного имени, иначе поток битов недопустим. Допустимо для таблицы символов быть неполной и содержать имена, ассоциированные с некоторыми, но не всеми, символами, используемые в оркестре и таблице. Присутствие строки нулевой длины в записи таблицы символов указывает, что имя для этого символа не включается в таблицу символов.

Реализации *SAOL* и *SASL*, которые требуют текстового ввода, а не маркируемого ввода, допустимы в декодере совместимости. Тогда декодер может демаркировать поток битов прежде, чем он будет обработан. В этом случае любые символы без ассоциированных имен предполагаются ассоциированными с именем по умолчанию формы *_sym_x*, где *x* является значением символа. Имена этой формы с этой целью резервируются в *SAOL*, и таким образом гарантируется, что имена не будут приходить в противоречие с именами символа, определенными с помощью таблицы символов.

```
/*********************************
orchestra file definitions
*********************************/
class orch_token}                       // a token in an orchestra
        int done;
unsigned int (8) token;                 // see standard token table, Annex A
switch (token) {
case  0xF0:                             // a symbol
        symbol sym;                             // the symbol name
        break;
case 0xF1:                              // a constant value
        float (32) val;                 // the floating-point value
        break;
case 0xF2:                              // a constant int value
        unsigned int (32) val;          // the integer value
        break;
```

```
case 0xF3:                          // a string constant
        int (8) length;
        unsigned int (8) str [length];   // strings no more than 255 chars
        break;
case 0xF4:
        int (8) val;
        break;
case 0xFF:      //end of oech
        done = 1;
        break;
}
}
class orc_file {                    // a whole orch file
        unsigned int (16) length;
        orch_token data [length];
}
```

Файл оркестра (планировки) является строкой маркеров. Маркеры представляют собой синтаксические элементы, такие как зарезервированные слова, базовые имена кода операции и знаки препинания. Кроме того, есть пять специальных маркеров. Маркер 0x*F*0 является маркером символа. Когда он встречается, следующие 16 битов в потоке битов должны быть номером символа. Маркер 0x*F*1 является маркером значения. Когда он встречается, следующие 32 бита в потоке битов должны быть значением с плавающей точкой. Этот маркер должен использоваться для всех символьных констант в пределах программы *SAOL* за исключением тех, с которыми встречаются в специальных целочисленных контекстах. Маркер 0x*F*2 является целочисленным маркером. Когда он встречается, следующие 32 бита в потоке битов должны быть целочисленным значением без знака. Маркер 0x*F*3 является строковым маркером. Когда он встречается, следующие несколько битов в потоке битов должны представлять строку символов (этот маркер в настоящий момент не используется). Маркер 0x*F*4 является маркером байта. Когда он встречается, следующие 8 битов в потоке битов должны быть целочисленным значением без знака. Маркер 0x*FF* является маркером конца оркестра; этот маркер не имеет никакой синтаксической функции в оркестре *SAOL*, но показывает конец раздела файла оркестра потока битов.

Не каждой последовательности маркеров разрешено выступать как файл оркестра. Полная последовательность маркеров оркестра должна соответствовать производству <оркестра>.

```
/*********************************
score file definitions
*********************************/
class instr_event {                 // a note-on event
      bit(1) has_label;
      if (has_label)
        symbol label;
symbol iname_sym;                   // the instrument name
float(32) dur;                      // note duration
unsigned int(8) num_pf;
float(32) pf[num_pf];               // all the pfields (no more than 255) }
class control_event                 // a control event
    bit(1) has_label;
    if (has_label)
       symbol label;
    symbol varsym;                  // the controller name
    float(32) value                 // the new value
}
class table_event {
    symbol tname;                   // the name of the table
    bit(1) destroy;                 // a table destructor
    if (!destroy) {
       token tgen;                  // a core wavetable generator
       bit(1) refers_to_sample;
```

```
        if (refers_to_sample)
            symbol table_sym;              // the name of the sample
    unsigned int(16) num_pf;               // the number of pfields
    if (tgen == 0x7D) {                    // concat
        float(32) size;
        symbol ft[num_pf - 1]; }
        else {
        float (32) pf[num_pf]; }
// when coding sample generator, leave a blank array slot
// for "which" parameter, to maintain alignment for "skip" parameter
        }
}
class end_event {
      // fixed at nothing
}
class tempo_event {  // a tempo event
      float(32) tempo;
}
class score_line {
    bit(1) has_time;
    if (has_time) {
        bit (1) use_if_late;
        float(32) time;                    // the event time
}
bit (1) high_priority;
bit(3) type;
switch (type) {
        case 0b000 : instr_event inst; break;
        case 0b001 : control_event control; break;
        case 0b010 : table_event table; break;
        case 0b100 : end_event end; break;
        case 0b101 : tempo_event tempo; break;
      }
}
class score_file {
        unsigned int(20) num_lines;  // a whole score file
        score_line lines[num_lines];
}
```

Файл отсчета является рядом строк информации об отсчете, предоставленной в заголовке информации о потоке. События, которые известны перед началом передачи потока битов в реальном времени, могут быть включены в заголовок так, чтобы они сразу были доступны декодеру, что может помочь эффективному вычислению в определенных реализациях. Каждая линия должна быть одним из пяти событий. У каждого типа событий есть свои импликации в процессе декодирования и планирования. Инструмент событий определяет время старта, инструмент символа имени, продолжительность, и любые другие параметры звука, проигрываемого на инструменте *SAOL*. Событие управления определяет параметр управления, который передают инструменту или инструментам, уже генерирующим звук. Таблица событий динамически создает или уничтожает глобальную звуковую таблицу в оркестре. Событие конца показывает конец работы оркестра. Событие темпа динамически изменяет темп воспроизведения оркестром.

Файл отсчета не должен быть представлен в порядке увеличения времен событий. События должны быть "сортированы" планировщиком по мере их обработки. В файле отсчета у каждой линии отсчета должна быть отметка времени.

Бит *high_priority* указывает, что событие отсчета является высокоприоритетным событием. Бит *use_if_late* указывает, если бит *has_time* устанавливается, то событие отсчета должно использоваться, прибывает ли оно вовремя или нет.

```
/**********************************
        MIDI definitions
**********************************/
class midi_event {
    unsigned int(24) length
    unsigned int(8) data[length];
}
class midi_file {
    unsigned int(32) length;
    unsigned int(8) data[length];
}
```

Участки *MIDI* позволяют включать информацию отсчета *MIDI* в заголовке потока битов и поток битов. Класс событий *MIDI* содержит единственную инструкцию *MIDI*. Класс файла *MIDI* содержит массив байтов, соответствующих *MIDIFile* стандартного формата 0 или формата 1. Каждая последовательность данных может иметь место в любом случае. Допустимые синтаксисы событий *MIDI* и *MIDIFiles* помещают в нормативные границы на синтаксически допустимых потоках битов структурированного аудио *MPEG*-4. Участки данных могут быть длиной до 2^{24}-1 и 2^{32}-1 байтов. Более длинные сообщения должны быть разбиты на несколько элементов потока битов.

```
/**********************************
         sample data
**********************************/
class sample {
      /* note that 'sample' can be used for any big chunk of data
      that needs to get into a wavetable */
symbol sample_name_sym;
unsigned int(24) length;                    // length in samples
bit(1) has_srate;
if (has_srate)
      unsigned int(17) srate;               // sampling rate (needs to go to 96 KHz)
bit(1) has_loop;
if (has_loop) {
      unsigned int(24) loopstart;           // loop points in samples
      unsigned int(24) loopend; }
bit(1) has_base;
      if (has_base)                         // base freq in Hz
bit(1) float_sample;
if (float_sample) {
      float(32) float_sample_data[length]; // data as floats ..
}
else {
      int(16) sample_data[length]; // ... or as ints
      }
}
```

Участок выборки включает блок данных, который будет включен в звуковую таблицу в оркестре *SAOL*. Каждая выборка состоит из имени, длины блока данных, и четырех дополнительных параметров: частота дискретизации, точки начала цикла и конца цикла, и основная частота. Доступ к данным в выборке обеспечивается через генератор звуковой таблицы выборки.

Данные выборки могут быть представлены как 32-разрядные значения с плавающей точкой, тогда они должны масштабироваться между –1 и 1 или как 16-разрядные целочисленные значения, тогда они должны масштабироваться между -32768 и 32767. В случае, когда данные выборки представляются как целочисленные значения, после включения в звуковую таблицу, они должны повторно масштабироваться к плавающей точке.

Каждая выборка именуется символом. Если у двух выборок в заголовке конфигурации декодера или в модуле одиночного обращения одно и то же имя, результат является неустановленным. Если у выборки в модуле доступа будет то же имя как у выборки в предыдущем модуле доступа или в заголовке конфигурации декодера, то новая выборка должна заменить старую выборку для доступов к нему по этому имени

через генератор звуковой таблицы выборки для любого генератора таблиц, выполняемого в это время или позже, чем время декодирования модуля доступа содержащего новую выборку. На таблицы, которые уже были сгенерированы, это не влияет.

```
/************************
sample bank data
************************/
class sbf {
int(32)                 length;
int(8)                  data[length];
}
```

Участок данных непрозрачен относительно транспортировки системами *MPEG-4*. Это должно соответствовать спецификации формата — то есть, это должен быть участок данных *RIFF*, начинающий *"RIFF..."*.

```
        /***********************************
                bitstream formats
        ***********************************/
    class StructuredAudioSpecificConfig {  // the bitstream header
            bit more_data = 1;
            while (more_data) {  // shall have at least one chunk
            bit(3) chunk_type;
            switch (chunk_type) {
            case 0b000 : orc_file orc; break;
            case 0b001 : score_file score; break;
            case 0b010 : midi_file SMF; break;
            case 0b011 : sample samp; break;
            case 0b100 : sbf sample_bank; break;
            case 0b101 : symtable sym; break;
                    }
            bit(1) more_data;
               }
    }
```

Конфигурация декодера потока битов содержит всю информацию, чтобы сконфигурировать и запустить структурированный аудиодекодер. Она содержит последовательность одного или более участков, где каждый участок является одним из следующих типов: файл оркестра, файл отсчета, файл *midi*, данные выборки, банк выборки, или таблица символов. Многие участки каждого из этих типов могут появляться в потоке битов (за исключением *midi_file*) со следующей семантикой:

1. *orc_file:* многочисленные файлы оркестра должны быть объединены. Если больше чем один глобальный блок появляется в объединенном оркестре, то это синтаксическая ошибка.

2. *score_file*: многочисленные файлы отсчета должны быть сортированы совместно по временам событий и объединены.

3. *midi_file*: только один элемент *midi_file* может появиться в потоке битов структурированного аудио.

4. *sample*: к *samples* может получить доступ в оркестр.

5. *sbf*: все многочисленные банки выборок доступны для процесса синтеза. Поведение не определено, если присутствует определенная комбинация *MIDI* и номер банка *MIDI* используется не один раз, либо в единственном банке выборок, либо в множественных банках выборок.

6. *symtable*: множественные таблицы символов, каждая из которых дает имена символам в оркестре. Имена N_0 в первой таблице символов в потоке битов применяются к символам 0.. N_0-1; имена N_1 во второй таблице символов к символам N_0.. N_0+N_1-1; и так далее.

```
    class SA_access_unit {        // the streaming data
    bit(1) more_data = 1;
    while (more_data) {
    bit(2) event_type;
    switch (event_type) {
    case 0b00 : score_line score_ev; break;
```

```
    case 0b01 : midi_event midi_ev; break;
    case 0b10 : sample samp; break;
        }
    bit(1) more_data;
      }
}
```

Блок доступа структурированного аудио содержит управляющую информацию потоковой передачи в режиме реального времени, которая будет обеспечена для выполнения процесса декодирования структурированного аудио. Он может содержать такое количество команд управления, какое требуется и разрешено допустимой пропускной способностью. Он не должен содержать новые инструментальные определения. Конфигурация оркестра фиксируется при запуске декодера. Он может содержать линии отсчета, события *MIDI* и новые данные выборки. Когда обеспечивается часть блока доступа, линия отсчета не обязана содержать метку времени. Когда *has_time* очищается в классе *score_line*, событие диспетчеризируется немедленно. Линии отсчета без меток времени не являются реагирующими к изменениям темпа оркестра.

6 Типы объектов

Есть четыре типа объектов, стандартизированных для структурированного аудио. Каждый из этих типов соответствует определенному набору эксплуатационных характеристик. Объектным типом по умолчанию является объектный тип 4. Когда ссылка дается на формат структурированного аудио *MPEG*-4 независимо от объектного типа, это нужно понимать как то, что ссылка сделана на объектный тип 4.

Терминалы, реализующие профили систем *MPEG*-4, содержащие узел *AudioFX,* должны также обеспечивать поддержку для объектного типа 3 или 4 структурированного аудио.

1. *MIDI only*. В этом объектном типе в заголовке информации о потоке должен появиться только участок *midi_file*, и только событие *midi_event* должно иметь место в данных потока битов. В этом объектном типе используются отображения вставки *General MIDI*. Этот объектный тип используется, чтобы задействовать обратную совместимость с существующим контентом *MIDI* и устройствами рендеринга. Нормативное и независимое от реализации качество звука не может быть обеспечено в этом объектном типе.

2. *Wavetable syntesis*. В этом объектном типе только файл *midi_file* и участки *sbf* должны появиться в заголовке информации о потоке, и только событие *midi_event* должно иметь место в данных потока битов. Этот объектный тип используется, чтобы описать контент музыки и звуковых эффектов в ситуациях, в которых не требуются полная гибкость и функциональность *SAOL*, включая 3–*D* аудио.

3. Алгоритмический синтез и *AudioFX*. В этом объектном типе участки *sbf* и *midi_file* не должны появляться в заголовке информации о потоке. Этот объектный тип используется, чтобы описать алгоритмический синтез и обеспечить обработку звуковых эффектов в узле *AudioFX*, когда использование формата банка выборок *SASBF* не требуется.

4. Основной синтетический. Могут появиться все элементы потока битов и элементы потоковой информации.

7 Процесс декодирования

7.1 Введение

Этот подпункт описывает алгоритмический структурированный процесс декодирования аудио, в котором поток битов соответствующий, объектному типу 3 или 4 преобразовывается в звук.

7.2 Заголовок конфигурации декодера

При создании элементарного потока структурированного аудио создается декодер структурированного аудио и для этого декодера обеспечивается объект потока битов класса *SA_decoder_config,* как информация о конфигурации. В это время декодер должен инициализировать планировщика времени выполнения, и затем разобрать заголовок конфигурации декодера на его составные части и использовать их следующим образом:

Файл оркестра: Файл оркестра должен быть проверен на синтаксическое соответствие грамматике *SAOL* и семантике уровня. Безотносительно предварительной обработки (то есть, компиляция, выделение статического хранения и т. д.) должна быть сделана подготовка к времени выполнения оркестра.

Файл отсчета: Каждое событие в файле отсчета должно быть зарегистрировано в планировщике. "Зарегистрироваться" означает сообщать планировщику о присутствии определенного параметризованного события в определенное будущее время, и о соответствующих действиях планировщика.

Файл *MIDI*: Каждое событие в файле *MIDI* должно быть преобразовано в соответствующее событие, и эти события зарегистрированы в планировщике.

Банк выборок: Данные должны храниться в банке выборок, и должны выполняться безотносительно предварительной обработки, необходимой чтобы подготовиться к использованию банка для синтеза.

Данные выборки: Данные в выборке должны храниться, и должны выполняться безотносительно предварительной обработки, необходимой чтобы подготовить данные из генератора звуковой таблицы *SAOL*. Если данные выборки представлены как 16-разрядные целые числа в потоке битов, то в это время они должны быть преобразованы в формат с плавающей точкой.

Таблица символов: Никакое нормативное поведение декодера не связывается с таблицей символов.

Если имеется больше одного файла оркестра в заголовке информации о потоке, различные файлы объединяются посредством сочленения и обрабатываются как один большой файл оркестра. Таким образом, каждый файл оркестра в рамках потока битов ссылается на одно и то же глобальное пространство имен, инструментальное пространство имен и пространство имен кода операции.

7.3 Данные потока битов и создание звука

7.3.1 Отношение с системным уровнем

На каждом шаге, в рамках системной работы, системный уровень может представить декодер структурированного аудио с блоком доступа, содержащим данные, соответствующие классу *SA_access_unit*. Декодер структурированного аудио отвечает за то, чтобы получить элементы данных *AU*, проанализировать и понять их как различные элементы данных потока битов структурированного аудио, выполнить продолжающийся оркестр *SAOL*, произвести одну единицу композиции выхода и передать системному уровню эту единицу композиции.

7.3.2 Элементы данных потока битов

Поскольку блоки доступа получают из системного демультиплексора, они анализируются и используются декодером структурированного аудио различными способами следующим образом:

Выборочные данные должны храниться, и независимо от того, что предварительная обработка необходима для ссылки предстоящими линиями отсчета, содержащими ссылки на эту выборку, будут выполняться. Если выборочные данные будут представлены как 16-разрядные целые числа в потоке битов, то они должны быть преобразованы в формат с плавающей точкой. Любые выборки в блоке доступа должны быть обработаны перед линиями отсчета, если линии отсчета ссылаются на эти выборки.

События линии отсчета должны быть зарегистрированы в планировщике, если у них будут отметки времени, или если их нет, то выполняться в следующем *k*-цикле.

События *MIDI* должны быть преобразованы в соответствующие события *SAOL*, и затем зарегистрированы в планировщике, если у них есть отметки времени, или если их нет, то выполняться следующем *k*-цикле.

7.3.3 Семантика планировщика

7.3.3.1 Назначение планировщика

Планировщик является центральным механизмом управления системы декодирования структурированного аудио. Он отвечает за обработку событий инструментами создания и прекращения, отслеживая то, какие инструментальные создания активны, давая различным инструментам команду выполнить синтез, направляя вывод инструментов на шины, и отправляя шины инструментам эффектов.

7.3.3.2 Инстанцирование инструмента

Инстанцировать (создать) инструмент означает создать пространство данных для его переменных и пространство данных, требующееся для любых кодов операции, вызванных этим инструментом. Когда инструмент инстанцируется, должны выполняться следующие задачи. Должно быть выделено место для любых полей параметра и установлены их значения согласно *p*-полям выражения или события инстанцирования. Затем должно быть выделено место для любых локально объявленных переменных, и значения этих переменных установлены в 0. Затем в локальное место для хранения должны быть скопированы текущие значения любых импортированных переменных *i-rate*. Затем должны быть созданы локально объявленные звуковые таблицы и заполнены данными согласно их объявлению.

7.3.3.3 Завершение инструмента

Завершить инстанцирование инструмента означает уничтожить пространство данных для этого экземпляра.

7.3.3.4 Выполнение инструмента

Выполнить инстанцирование инструмента на определенной скорости означает вычислить результаты команд, данных в это инструментальное определение. Когда экземпляр инструмента будет выполнен на определенной скорости, должны выполняться следующие шаги. Во-первых, значения любых глобальных переменных и звуковых таблиц импортированных этим инструментом на этой скорости должны быть скопированы в место для хранения инструмента. Кроме того, во время выполнения на *a*-скорости экземпляр инструмента, который является целью оператора *send*, текущее значение стандартного имени *input* в экземпляре должны быть установлены в текущее значение шины или шин, на которые ссылаются в операторе *send*. Затем, блок кода для этого инструмента должен выполняться на определенной скорости с учетом пространства данных инстанцирования инструмента. Затем, значения любых глобальных переменных и звуковых таблиц, экспортируемых этим инструментом на этой скорости, должны быть скопированы в глобальное место для хранения. Наконец, при выполнении инстанцирования инструмента на *a*-скорости значение вывода инструмента должно быть добавлено к шине, на которую направляется инструмент. Если инструмент не является целью выражения *send*, ссылающегося на специальную шину *output_bus*, вывод инструмента является выводом оркестра и может быть превращен в звук.

7.3.3.5 Запуск оркестра и конфигурация

7.3.3.5.1 Введение

Задачи определения инструмента и ширины шины, назначения глобальной переменной, выполнение *startup*, создания глобальной звуковой таблицы, инициализации шины и инструмента *send* должны быть выполнены в обозначенном ниже порядке.

7.3.3.5.2 Определение ширины выхода инструмента и ширины шины

Ширина выхода каждого инструмента определяется в порядке, определенном глобальными правилами упорядочивания. Ширина каждой шины обеспечивается оператором *send* или определяется суммой выходных ширин инструментов, направленных к этой шине в *route*. У любого инструмента, который не получает данных шины согласно правилам упорядочивания, должна быть ширина *inchannels* равная 1 (эта спецификация необходима, так как выходные ширины могут зависеть от значения *inchannels* или ширины *input*).

Примечание: если выходная ширина инструмента зависит от *outchannels* и ширина шины, куда этот инструмент направляется, не определяется посредством других операторов *send/route*, оркестр может быть недетерминированным. Программисты должны обратить особое внимание на проверку детерминированного поведения оркестра или использовать определенные ширины шины.

7.3.3.5.3 Варьируемое выделение, выполнение запуска и создание глобальной звуковой таблицы

Для изменений любых глобальных сигналов должно быть выделено место, и их значения установлены в нуль. Если в оркестре будет инструмент, называемый *startup*, этот инструмент нужно инстанцировать и выполнить в *i-rate*. После того, как это выполнение закончено, создаются все глобальные звуковые таблицы и заполняются данными согласно их определениям в глобальном блоке оркестра.

7.3.3.5.4 Инициализация шин и инструментов *send*

После создания глобальной звуковой таблицы создаются и инициализируются шины оркестра. Каждый канал каждой шины устанавливается в значение 0. После того, как шины созданы, все инструменты, которые являются целями операторов *send*, нужно инстанцировать и выполнить в *i-rate*, в порядке, определенном глобальными правилами упорядочивания. Глобальное абсолютное время оркестра должно быть установлено в 0.

Примечание — Время называют *absolute*, если оно определяется в секундах. Когда сначала декодируется команда темпа и значение темпа изменяется из его значения по умолчанию, время отсчета и абсолютное время больше не идентичны. Все времена в отсчете следующем за выполнением линии темпа, масштабируются согласно новому темпу и ставятся в очередь в абсолютные времена диспетчеризации и продолжительности.

7.3.3.6 Выполнение декодера во время потоковой передачи

В каждом цикле оркестра процессом синтеза в реальном времени производится один композиционный модуль выборок. Этот синтез выполняется согласно правилам приведенным ниже, и получающийся вывод оркестра представляется системному уровню как композиционный модуль. Чтобы выполнить один цикл оркестра, должны выполняться следующие задачи в обозначенном порядке:

1. Если есть событие конца, чье время диспетчеризации ранее или равно текущему абсолютному времени оркестра, или событие конца было получено без метки времени после последнего выполнения этого правила, никакой дальнейший вывод не производится, и на все будущие запросы от системного уровня даются ответы буферами как нуль.

2. Если есть какие-либо инструментальные события, чьи времена диспетчеризации ранее или равна текущему абсолютному времени оркестра, или какие-либо инструментальные события были получены без меток времени после последнего выполнения этого правила, для каждого такого инструментального события создается инстанцирование инструмента, и каждый из этих инстанцирований выполняется в *i-rate* порядке, предписанном глобальными правилами упорядочивания. Если инструментальное событие определяет продолжительность для инстанцирования инструмента, то инстанцирование инструмента должно быть запланировано для завершения во время, данное суммой текущего абсолютного времени оркестра и указанной продолжительности (масштабируемый к абсолютным единицам измерения времени согласно фактическому темпу, если имеется).

3. Если есть какие-либо активные инстанцирования инструмента, время завершения которых ранее или равно текущему абсолютному времени оркестра, то стандартное имя *released* должно быть установлено в 1 в пределах каждого экземпляра такого инструментального инстанцирования, и инструмент отмечается для завершения в шаге 12.

4. Если имеются какие-либо события управления, чьи времена диспетчеризации ранее или равны текущему абсолютному времени оркестра, или любые события управления были получены без меток времени после последнего выполнения этого правила, глобальным переменным или инструментальным переменным в пределах инстанцирований инструмента, маркированных каждым таким событием управления, нужно обновить их значения. Подразумевается, что оркестром может быть получено не больше, чем одно изменение управления на переменную за цикл управления . Если множественные изменения управления, которые ссылаются на одну и ту же переменную, получаются в единственном цикле управления, результирующее значение инструмента или глобальной переменной является неуказанным.

5. Если есть какие-либо табличные события, чьи времена диспетчеризации ранее или равны текущему абсолютному времени оркестра, или любые табличные события были получены без меток времени после последнего выполнения этого правила, глобальные звуковые таблицы должны создаваться или уничтожаться, как определено табличным событием.

6. Если имеются какие-либо события *MIDI*, метка времени которых ранее или равна текущему времени оркестра, или которые были получены без меток времени после последнего выполнения этого правила, они диспетчеризируются согласно их семантике.

7. Если есть какие-либо события темпа, чьи времена диспетчеризации ранее или равны текущему абсолютному времени оркестра, или любые события темпа были получены после последнего выполнения этого правила, то глобальная переменная темпа должна быть установлена в указанное значение, и времена диспетчеризации всех событий, ожидающихся для выполнения, должны масштабироваться к новым временам согласно новому значению темпа. Уже запланированные времена для завершений также масштабируются в их остающейся части, согласно отношению между старым и новым темпом. На существующие времена *extend* это не влияет, так как они определяются в абсолютном времени и таким образом "вне" отсчета. Значение стандартного имени *dur* должно быть изменено в каждом активном экземпляре инструмента, чтобы отразить новую продолжительность инструмента.

Если множественные события темпа обрабатываются согласно предыдущему абзацу в том же самом цикле управления, то глобальная переменная темпа должна изменяться только однажды к темпу, обозначенному в событии темпа, полученном последним или с последней меткой времени, а другие события темпа отбрасываются. Если есть множественные события темпа с той же самой меткой времени, или события без метки времени, события с меткой времени должны быть диспетчеризированы в том же самом цикле управления, и результирующее значение глобальной переменной темпа не указывается.

Примечание — Если текущее время оркестра отличается от времени диспетчеризации темпа, чтобы вычислить новые продолжительности, и будущее время диспетчеризации событий должно использоваться прежнее время.

8. Если поле *speed* узла сцены *AudioSource*, ответственное за инстанцирование этого декодера, было изменено в последнем *k-rate*, переменная стандарта темпа должна быть установлена в 60-кратное значение, и события в оркестре должны повторно масштабироваться.

9. Значение каждого канала каждой шины должно быть установлено в 0.

10. Каждый активный экземпляр инструмента должен выполняться однажды в *k-rate* и *n* раз в *a-rate*, где *n* является числом выборок в период управления. Каждое выполнение в *k-rate* должно быть выполнено в порядке, данном глобальными правилами упорядочивания, и каждое соответствующее выполнение

в *a-rate* (то есть, первое выполнение *a-rate* в *k-rate*, второе выполнение *a-rate* в *k-rate* и т.д.) должно быть в порядке, данном глобальными правилами упорядочивания.

Примечания

1 Если инструмент *a* упорядочен перед инструментом *b,* то выполнение *k-rate a* должно быть строго перед выполнением *k-rate b* и выполнение *k-rate a* должно быть строго перед первым выполнением *a-rate a*, и первое выполнение *a-rate a* должно быть строго перед первым выполнением *a-rate b*. Однако нет никакого нормативного упорядочивания между вторым выполнением *a-rate a* и первым выполнением *a-rate b* или между первым выполнение *a-rate a* и выполнением *k-rate b* в пределах определенного цикла оркестра.

2 В соответствии с правилами упорядочивание выполнения, описанное в этом подпункте, может быть перестроено или проигнорировано, когда исходя из исследования оркестра может быть решено, что если так сделать, то не будет оказано никакого влияния на вывод процесса декодирования. "Не имеет никакого влияния" должно быть принято как означающее, что вывод процесса декодирования в перестроенном порядке является повыборочно идентичным выводу процесса декодирования, выполняемого строго согласно правилам в этом подпункте.

3 Выполнения *k-rate* каждого экземпляра инструмента должны происходить как атомарная работа; то есть выполнение *k-rate* одного инструмента должно быть завершено прежде, чем начинается следующее. Не допустимо выполнять *k-rate* параллельно. Это не касается *a-rate.* Если у двух инструментов нет никакого отношения упорядочивания согласно глобальным правилам упорядочивания их, *a-rate* могут быть выполнены в любом порядке или параллельно.

11. Если специальная шина *output_bus* направляется к инструменту, вывод этого инструмента в каждом *a-rate* является выводом оркестра в этом *a-rate*. Иначе значение специальной шины *output_bus* после выполнения каждого инструмента для *a-rate* является выводом оркестра в этом *a-rate*. Если значение текущего вывода оркестра будет больше чем 1 или меньше чем -1, оно должно быть установлено в 1 или -1, соответственно.

12. Если инструмент, названный *extend* с параметром больше, чем количество времени в цикле управления, инструмент не завершается. Все другие инструменты, отмеченные для завершения в шаге 3, завершаются. В случае сообщения *MIDI "All Notes Off"* инструменты могут не расширять себя, и в это время уничтожаются.

13. Текущее глобальное абсолютное время оркестра продвигается на один период управления.

7.3.3.7 Приоритет событий

Определенные события могут быть определены как "приоритетные" события, устанавливая соответствующее поле в элементе потока битов или используя * маркер в текстовом отсчете. В случае, когда происходит превышение возможностей декодера, этот флаг позволяет автору контента минизировать ухудшение выполнения события.

Если флаг *high_priority* будет установлен, и если наступит критическое состояние, событие всегда должно выполняться без ухудшения, и никакие инстанцирования, создаваемые на низких приоритетных уровнях, не будут активными. Если флаг *high_priority* будет убран, то событие должно выполняться без ухудшения, если не наступят никакие критические состояния. Инструментальные события с убранным флагом *high_priority* могут быть преждевременно завершены, если не доступны ресурсы, чтобы диспетчеризировать событие с установкой флага *high_priority*.

Примечание — Ухудшение не планируется как дозволенный, нормативный метод, чтобы понизить вычислительную сложность. Соответствующие декодеры должны быть в состоянии декодировать в нормальных условиях потоки битов указанного уровня *Profile@Level* без ухудшения.

7.3.3.8 Запаздывающие события

В случае ошибки транспортировки или ошибки кодера определенные события могут поступить с метками времени, которые уже прошли. Поле *use_if_late* в элементе потока битов указывает надлежащее поведение в этом случае. Если это поле очищается, то событие игнорируется, и обработка должна продолжаться, как будто событие никогда не прибывало. Если это поле устанавливается, то событие немедленно диспетчеризуется, как если бы оно было получено без метки времени.

7.4 Соответствие

Относительно всего нормативного языка соответствие нормативному языку измеряется во время вывода оркестра. Любая оптимизация кода *SAOL* или перестановка последовательности обработки могут быть выполнены, пока их выполнение никак не влияет на вывод оркестра. "Не имеет никакого влияния", в этом смысле означает, что вывод перестроенного или оптимизированного оркестра является повыборочно идентичным выводу исходного оркестра.

8 Синтаксис и семантика *SAOL*

8.1 Соотношение с синтаксисом потока битов

Описание синтаксиса потока битов определяет представление инструментов и алгоритмов *SAOL*, которые должны быть предоставлены декодеру в потоке битов. Однако маркируемое описание, как представлено там, не достаточно, чтобы описать синтаксис и семантику языка *SAOL*. Чтобы обеспечить создание потока битов и обмен надежным способом, полезно иметь стандартное удобочитаемое текстовое представление кода *SAOL* в дополнение к маркируемому двоичному формату.

Грамматика формата *Backus-Naur (BNF)* обозначает язык или бесконечный набор программ. Допустимые программы, которые могут быть переданы в потоке битов, ограничиваются этим набором. Любая программа, которая не может быть проанализирована этой грамматикой, не является допустимой программой *SAOL* — у нее есть синтаксическая ошибка — и поток битов является недопустимым потоком. Хотя поток битов составляется из маркеров, грамматика будет описана с точки зрения лексических элементов. Синтаксические правила, выраженные грамматикой, которые ограничивают набор текстовых программ, также нормативно ограничивают синтаксис потока битов через отношение потока битов и текстового формата в нормативном процессе маркирования.

Этот подпункт таким образом описывает текстовое представление *SAOL*, которое стандартизируется, но стоит за пределами отношения декодера потока битов.

Приложение В содержит грамматику для текстового языка *SAOL*, представленного в форматах *'lex'* и *'yacc'*. Используя эти версии грамматики, синтаксические анализаторы могут быть созданы автоматически, используя инструменты *'lex'* и *'yacc'*. Эти версии годятся только в информативных целях и используются в создании декодера.

Нормативный язык синтаксиса в этом подпункте обеспечивает границы синтаксически верных программ *SAOL* и дополнительно, синтаксически верных последовательностей потока битов, которые могут появиться в классе потока битов *orchestra*. Таким образом есть конструкции, которые допустимы только после чтения грамматики *BNF*, но отвергаются в нормативном тексте, сопровождающем грамматику. Состояние таких конструкций является состоянием тех, которые лежат за пределами языка, определенного одной только грамматикой.

Процесс декодирования для потоков битов, содержащих синтаксически недопустимые программы *SAOL* (то есть, программы *SAOL*, которые не соответствуют грамматике *BNF*, или содержат синтаксические ошибки или ошибки несоответствия уровня), является необусловленным.

Нормативная в отношении языка семантика в этом подпункте описывает семантические границы поведения декодера структурированного аудио. Определенные конструкции описывают "ошибочные по времени выполнения" ситуации. Поведение декодера при таких обстоятельствах не нормировано, но реализации позволяют корректно восстановиться из таких ситуаций и продолжать декодирование.

8.2 Лексические элементы

8.2.1 Понятия

Текстовый оркестр *SAOL* содержит знаки препинания, которые синтаксически снимают неоднозначность оркестра, идентификаторы, которые обозначают символы оркестра, числа, которые обозначают постоянные величины, строковые константы, которые в настоящий момент не используются, комментарии, которые допускает внутренняя документация оркестра и пробел, который лексически разделяет различные текстовые элементы. Эти элементы не появляются в потоке битов так, как каждый представляется там маркером.

8.2.2 Идентификаторы

Идентификатор является серией одной или более букв, цифр и подчеркиваний, которая начинается с буквы или подчеркивания. Он обозначает символ оркестра. Идентификаторы являются чувствительными к регистру, что означает, что идентификаторы, которые отличаются регистром только одного или более символов, обозначают различные символы.

Строка символов, эквивалентная одному из зарезервированных слов, перечисленных в 8.9, одному из стандартных имен перечисленных в 8.6.8, имени одного из базовых кодов операции перечисленных в 9.3, или имени одного из базовых генераторов звуковой таблицы, перечисленных в 10, не обозначает символ, а обозначает что зарезервировано слово, стандартное имя, код операции, или генератор звуковой таблицы.

Идентификатор обозначается в грамматике *BNF* терминальным символом *<ident>*.

8.2.3 Числа

Существуют два вида символьных констант, которые содержат числовые значения в *SAOL*: целочисленные константы и константы с плавающей точкой.

Целочисленная константа обязана появляться в определенных контекстах, таких как определения массива. Целочисленный маркер является серией одной или более цифр. Строка символов, которая оказывается отрицательным целым числом, должна быть лексически проанализирована как константа с плавающей точкой. Никакая целочисленная константа, больше чем 2^{32} (4294967296), не должна иметь места в оркестре.

В *SAOL* нет разницы между числами, кодированными с маркером потока битов для целых чисел и кодированными с маркером потока битов для байтов. Последний является только средством сжатия потока битов.

Целочисленная константа обозначается в грамматике *BNF* терминальным символом <*int*>.

Константа с плавающей точкой присутствует в выражениях *SAOL* и обозначает постоянное числовое значение. Маркер с плавающей точкой состоит из основания, дополнительно сопровождаемого экспонентой. Основание является серией одной или более цифр, опционно сопровождаемых десятичной точкой и серией из ни одной или большего количества цифр, или десятичной точкой, сопровождаемой серией из одной или более цифр. Экспонента обозначается буквой *e*, дополнительно сопровождаемой символом + или –, сопровождаемым серией из одной или более цифр. Так как константа с плавающей точкой появляется в выражении *SAOL*, где унарный оператор отрицания всегда является доступным, константы с плавающей точкой не должны быть лексически отрицательными. Каждая константа с плавающей точкой в оркестре должна быть представимой 32-разрядным числом с плавающей точкой.

Константа с плавающей точкой обозначается в грамматике *BNF* терминальным символом <*number*>.

8.2.4 Строковые константы

Строковые константы не используются в нормативной спецификации *SAOL*, но описаны здесь, чтобы они могли обрабатываться разработчиками, которые хотят добавлять к своим реализациям функциональность свыше нормативных требований.

Строковая константа обозначает постоянное строковое значение, то есть, последовательность символов. Строковая константа является серией символов, заключенных в двойные кавычки (“). Символ двойных кавычек может быть включен в строковую константу, предшествуя ей с символом наклонной влево черты (\). Любой другой символ, включая символ разрыва линии (новая линия), должен быть заключен в кавычки.

8.2.5 Комментарии

Комментарии могут использоваться в текстовом представлении *SAOL* чтобы внутренне документировать оркестр. Однако они не включаются в поток битов и теряются в последовательности *tokenisation/detokenisation* (назначения/удаления маркеров).

Комментарий является любой серией символов, начинающихся с двух наклонных черт (*//*), и завершающихся с новой линией. Во время лексического анализа, всякий раз когда в линии находится элемент *//*, остальная часть линии игнорируется.

8.2.6 Пробел

Пробел служит для того, чтобы лексически разделить различные элементы текстового оркестра *SAOL*. Он не имеет никакой синтаксической функции в *SAOL*, и не представляется в потоке битов, таким образом точный пробел текстового оркестра теряется в последовательности *tokenisation/detokenisation*. Пробел не требуется в *SAOL*, кроме случаев когда надо снять неоднозначность маркеров и зарезервированных слов, которые появляются друг за другом (например, отделяет *“asig”* от объявленного имени переменной).

Пробел является любой серией одного или более пространств, таблицы, и/или символов новой строки.

8.3 Переменные и значения

Каждая сигнальная переменная в рамках оркестра *SAOL* содержит значение или упорядоченный набор значений для переменных массива, как промежуточное вычисление оркестра. В любом моменте времени значение переменной, выборка в звуковой таблице или единственный элемент переменной массива должны быть представлены 32-разрядным значением с плавающей точкой.

Реализации свободно использовать любое внутреннее представление для переменных значений, пока вычисленные результаты идентичны результатам вычислений, используя 32-разрядные значения с плавающей точкой.

Определенными чувствительным цифровым образом фильтрующими операциями, результаты использования большей точности в вычислении могут быть эквивалентно вредными для вывода оркестра, так же как и результаты использования меньшей точности, поскольку устойчивость фильтра может крити-

чески зависеть от ошибки квантования. Этому строго следуют для потоков битов, содержащих код, который генерирует совсем другие результаты, вычисленные с 32-разрядной и 64-разрядной арифметикой.

При выводе оркестра значения, вычисленные оркестром, должны находиться между минимальным значением –1 и максимальным значением 1. Эти значения при выводе оркестра представляют максимум негативно - и положительно оцененных аудиосэмплов, которые могут быть произведены терминалом. Если значения, вычисленные оркестром, падают вне того диапазона, они отсекаются к [-1,1]. Этот звук представляется циклом управления *MPEG*–4 системы для использования в составе *AudioBIFS*.

8.4 Оркестр

<orchestra> -> <orchestra element>
<orchestra> <orchestra> -> <orchestra element>

Оркестр является набором подпрограмм обработки сигналов и объявлений, которые составляют структурированное описание обработки аудиоданных. Это должно состоять из списка одного или более элементов оркестра.

<orchestra element> -> <global block>
<orchestra element> -> <instrument declaration>
<orchestra element> -> <opcode declaration>
<orchestra element> -> <template declaration>
<orchestra element> -> NULL

Есть четыре вида элементов оркестра:

1. Глобальный блок содержит инструкции для глобальных параметров оркестра, маршрутизаций шины, глобальных переменных объявлений, и инструментального упорядочивания. Недопустимо иметь больше, чем один глобальный блок в оркестре.

2. Инструментальные объявления описывают последовательности обработки инструкций, которыми можно параметрически управлять, используя *SASL* или файлы счета *MIDI*.

3. Объявления кода операции описывают последовательности обработки инструментов, которые обеспечивают инкапсулировавшую функциональность, используемую нулем или большим количеством инструментов в оркестре.

4. Шаблонные объявления описывают многократные инструменты, используя краткую параметрическую форму.

Элементы оркестра могут появиться в любом порядке в пределах оркестра. В частности определения кода операции могут произойти или синтаксически прежде или после того, как они используются в инструментах или других кодах операции.

8.5 Глобальный блок

8.5.1 Синтаксическая форма

<global block> -> global { <global list> }
<global list> -> <global statement> <global list>
<global list> -> NULL

Глобальный блок должен содержать глобальный список, который должен состоять из последовательности нулевых или более глобальных операторов.

<global statement> -> <global parameter>
<global statement> -> <global variable declaration>
<global statement> -> <route statement>
<global statement> -> <send statement>
<global statement> -> <sequence definition>
<global statement> -> <interpolation level>

Есть несколько видов глобального оператора.

1. Глобальные параметры устанавливают параметры оркестра, такие как частота дискретизации управляют уровнем и числом входных и выходных каналов звука.

2. Глобальные переменные определяют переменные, которые могут быть совместно использованы многократными инструментами.

3. Операторы маршрута описывают маршрутизацию инструментальных выводов на шины.

4. Определения последовательности описывают порядок работы инструментов планировщика времени.

5. Уровень интерполяции определяет качество интерполяции, выполняемой в процессе синтеза.

8.5.2 Глобальный параметр

8.5.2.1 Параметр *srate*

\<*global parameter*> -> *srate* <*int*>;

srate глобальный параметр определяет аудио частоту дискретизации оркестра. Процесс декодирования должен создать аудио на этой частоте дискретизации. Не допустимо упрощать сложность оркестра или учесть терминальную возможность, генерируя аудио на других частотах дискретизации. Это может иметь серьезное неблагоприятное воздействие на определенные элементы обработки оркестра.

srate параметр должен быть целочисленным значением между 4000 и 96000 Гц. Если *srate* параметр не будет обеспечен в оркестре, то значение по умолчанию должно быть самым быстрым из аудиосигналов, обеспеченных как входной. Если частота дискретизации не обеспечивается и нет никаких входных аудиосигналов, частота дискретизации по умолчанию должна быть 32000 Гц. Если больше чем одна *srate* инструкция параметра происходит в оркестре — то это синтаксическая ошибка.

В объектном терминале типа 3, или когда оркестр *SAOL* используется в *AudioFX* узла *AudioBIFS*, *srate* параметр должен быть одним из следующих значений: (4000, 8000, 11025, 12000, 16000, 22050, 24000, 32000, 44100, 48000, 88200, 96000).

8.5.2.2 Параметр *krate*

<global parameter> -> krate <int>;

krate глобальный параметр определяет уровень управления оркестра. *krate* параметр должен быть целочисленным значением между 1 и частотой дискретизации включительно. Если *krate* параметр не будет обеспечен в оркестре, то уровень управления по умолчанию должен составить 100 Гц. Если в оркестре больше чем одна *krate* инструкция параметра, то это синтаксическая ошибка.

Если уровень управления не равен делителю частоты дискретизации, то уровень управления является следующим большим целым числом, которое действительно равномерно делит частоту дискретизации. Период управления оркестра является числом выборок или количеством времени, представленным этими выборками, в одном цикле управления.

8.5.2.3 Параметр *inchannels*

<global parameter> -> inchannels <int>;

inchannels глобальный параметр определяет число каналов ввода. Если звуковые каналы обозначены, как входные источники, то дополнительные каналы должны быть установлены в непрерывные оцененные нули сигналы. Если звуковые каналы обозначены как входные источники, дополнительные каналы игнорируются.

Если *inchannels* параметр не будет обеспечен в оркестре, то значение по умолчанию должно быть суммой чисел каналов, обеспеченных входными источниками. Если нет никаких входных источников, значение должно быть 0. Если больше чем одна *inchannels* инструкция параметра происходит в оркестре, то это синтаксическая ошибка.

8.5.2.4 Параметр *outchannels*

<global parameter> -> outchannels <int>;

outchannels глобальный параметр определяет число каналов вывода звука. Процесс декодирования времени выполнения должен произвести и представить это число каналов внутренне. Не допустимо упростить сложность оркестра или учесть терминальную возможность, производя меньше каналов.

Если *outchannels* параметр не будет обеспечен в оркестре, то значение по умолчанию должно быть одним каналом. Если больше, чем одна *outchannels* инструкция параметра происходит в оркестре, то это синтаксическая ошибка.

5.8.5.2.5 Параметр *interp*

<global parameter> -> interp <int>;

interp глобальный параметр определяет качество интерполяции, выполняемой в процессе синтеза. Различные операции требуют доступа к звуковым таблицам в точках нецелого числа. Получить доступ к звуковой таблице в такой точке требует интерполяции среди доступных точек в звуковой таблице.

Если *interp* параметр 0, то "низкоуровневая" интерполяция выполняется. Каждая интерполяция должна выполняться, используя линейную интерполяцию. Таким образом, *i* и *j* могут быть двумя последовательными индексами звуковой таблицы и позволять *x* и *y* быть значениями в точках *i* и *j* соответственно.

Если *interp* параметр 1, то "высокоуровневая" интерполяция выполняется. Метод высокоуровневой интерполяции не нормирован, но это должен быть метод более высокого качества, чем линейная интерполяция.

Если параметр не 0 или 1, то это синтаксическая ошибка. Если больше чем одна *interp* инструкция параметра происходит в оркестре, то это синтаксическая ошибка.

Если *interp* параметр не определяется в оркестре, то качество интерполяции "низко" по умолчанию.

8.5.3 Глобальное переменное объявление

8.5.3.1 Синтаксическая форма

<global variable declaration> -> ivar <namelist> ;

<global variable declaration> -> ksig <namelist> ;

<global variable declaration> -> <table declaration> ;

Глобальные переменные объявления объявляют переменные, которые могут быть совместно использованы и к которым получен доступ всеми инструментами и счетом *SASL*. Только *ivar* и переменные типа *ksig*, так же как звуковые таблицы, могут быть объявлены глобально. Глобальное переменное объявление является табличным определением или вводит имя, сопровождаемое списком объявлений имени.

Глобальное объявление имени определяет, что маркер имени должен создавать пространство, равное значению сигнала, выделенному для временного хранения в глобальном контексте. Глобальное объявление массива определяет, что маркер имени должен создавать пространство, равное конкретному количеству сигнальных значений, выделенных в глобальном контексте.

8.5.3.2 Сигнальные переменные

<namelist> -> <name>, <namelist>

<namelist> -> <name>

namelist является последовательностью одного или более объявлений имени.

<name> -> <ident>

<name> -> <ident>[<array length>]

<array length> -> <int>

<array length> -> inchannels

<array length> -> outchannels

Объявление имени является идентификатором или объявлением массива. Для объявления массива параметр должен быть или целым числом, строго больше чем 0, или одним из маркеров *inchannels,* или *outchannels*. Если *outchannels,* то длина массива должна быть такой же, как число каналов ввода или каналов вывода оркестра, соответственно. Недопустимо использовать маркер *inchannels*, если число каналов ввода оркестра 0. Не каждый идентификатор может использоваться в качестве имени переменной. Зарезервированные слова, перечисленные в подпункте 8.8, стандартные имена, перечисленные в подпункте 8.6.8, именах базовых кодов операции, перечисленные в 9 и имена базовых генераторов звуковой таблицы, перечисленные в 10, не должны быть объявлены как имена переменной.

8.5.3.3 Объявления звуковой таблицы

<table declaration> -> table <ident> (<ident> , <expr> [, <expr list>]) ;

<expr> as defined in subclause

<expr>

<expr list>.

Звуковые таблицы являются структурами памяти и позволяют быстрое колебание, цикличное выполнение и воспроизведение. Объявление звуковой таблицы связывает имя (первый идентификатор) со звуковой таблицей, создаваемой базовым генератором звуковой таблицы, на который ссылается второй идентификатор. Если второй идентификатор не является одним из базовых генераторов звуковой таблицы, то это синтаксическая ошибка. Первое выражение в разграниченной запятой последовательности параметра называют выражением размера. Остающийся нуль или больше выражений включают в список параметра звуковой таблицы.

Семантика выражения размера и списка параметра звуковой таблицы определяется базовым генератором звуковой таблицы. Любое правильное выражение является законным, как часть табличного списка параметра. Позволяется ссылка на глобальные переменные (их значения могут быть установлены специальным инструментальным *startup*). Каждое выражение должно быть однозначным, кроме случая *concat* генератора, когда выражения должны быть табличными ссылками. Порядок создания звуковых таблиц не детерминирован, за исключением табличных параметров *concat* генератора, которые всегда сгенерированы перед *concat* генератором, который использует их. В этом случае таблицы, используемые в качестве параметров *concat* генератора, появляются перед таблицей, которая использует *concat* генератор, чтобы предотвратить циклы зависимости.

На глобальную звуковую таблицу может сослаться заполнитель звуковой таблицы в любом инструменте или коде операции. Глобальные звуковые таблицы должны быть созданы и инициализированы с данными во время инициализации оркестра, сразу после выполнения специального инструментального *startup*.

Чтобы создать звуковую таблицу, во-первых, поля выражения оцениваются в порядке, как они появляются в синтаксисе согласно правилам в 8.6.7. Затем выполняется определенный генератор звуковой таблицы, названный во втором идентификаторе, нормативная семантика каждого генератора звуковой таблицы детализирует, как звуковая таблица должна создаваться.

8.5.4 Оператор маршрута

<route statement> -> route (<ident> , <identlist>) ;
<identlist> -> <ident> , <identlist>
<identlist> -> <ident>
<identlist> -> <NULL>

Оператор *route* состоит из единственного идентификатора, который определяет шину и последовательность одного или более инструментальных имен, которые определяют инструменты. Оператор маршрута определяет, что перечисленные инструменты не производят звуковой вывод непосредственно и их результаты помещаются в данную шину. Каналы вывода от инструментов размещаются в отдельном канале шины. Многократные операторы *route* на ту же самую шину указывают, что данные инструментальные выводы должны быть суммированы на шине. Многократные операторы *route* с отличающимися числами каналов, ссылающихся на ту же самую шину, недопустимы.

Должно быть по крайней мере одно инструментальное имя в инструментальном списке.

Недопустимо, чтобы оператор *route* ссылался на шину, которая не является специальной шиной *output_bus*, и это не происходит в *send* операторе.

Недопустимо, чтобы оператор *route* обратился к специальной шине *input_*.

Все инструменты, которые не упоминаются в операторах *route*, помещают свой вывод в специальную шину *output_bus*, за исключением инструмента эффекта, которому был отправлен *output_bus*. Те же самые правила для допустимых комбинаций канала применяются к специальной шине *output_bus*, когда операторы маршрута были явными.

8.5.5 *send* оператор

<send statement>->send(<ident>;<expr list>;<namelist>); <namelist>
<namelist>
<expr list>

send оператор создает инструментальное инстанцирование, определяет шины, и определяет, что инструмент, на который ссылаются, используется в качестве процессора эффектов для этих шин.

Все шины в оркестре определяются при использовании *send* операторов. Недопустимо для оператора, ссылающегося на шину, чтобы можно было обратиться к шине, которая не определяется в *send* операторе. Исключением является специальная шина *output_bus*, которая всегда определена.

Идентификатор в *send* операторе ссылается на инструмент, который будет использоваться в качестве обрабатывающего шину инструмента, также названного инструментом эффекта. Нет никакого синтаксического различия между инструментами эффекта и другими инструментами. Список идентификаторов ссылается на одну или более шин, которые должны быть доступными для инструмента эффекта через его *input* стандартное имя, следующим образом:

Первые n_0 каналы *input*, каналы 0 через n_0-1 являются n_0 каналами первой шины, на которую ссылаются. Каналы n_0 через n_0+n_1-1 *input* являются n_1 каналами второй шины, и так далее, в общей сложности $n_0 + n_1 + \ldots + n_k$ каналы.

Кроме того, группировка шин в *input* массиве должна быть сделана доступной для инструмента эффекта через его стандартное имя *inGroup* следующим образом:

у первых n_0 значений *inGroup* есть значение 1;

у каналов n_0 через n_0+n_1-1 *inGroup* есть значение 2,

и так далее, через $n_0 + n_1 + \ldots + n_k$, с последним n_k и значением k.

Список выражения является списком нулевых или более *i-rate* выражений, которые обеспечиваются для инструмента эффекта как его поля параметра. Любое выражение *i-rate* является законным как часть этого списка; в частности позволяется ссылка на *i-rate* глобальные переменные. Число обеспеченных выражений должно соответствовать числу полей параметра, определенных в инструментальном объявлении; иначе, это — синтаксическая ошибка.

Инструмент эффекта, упомянутый в *send* операторе, нужно инстанцировать при запуске оркестра. Эти инструментальные инстанцирования должны остаться в силе, пока процесс синтеза оркестра не завершится. Одно инструментальное инстанцирование должно создаваться для каждого *send* оператора в оркестре. Если такое инструментальное инстанцирование использует *turnoff* оператор, инстанцирование уничтожается. Никакие другие изменения в оркестре не производятся.

Любая шина, за исключением специальной шины *output_bus*, может быть отправлена больше чем одному инструменту эффекта и/или инстанцированиям. В случае, когда простой идентификатор не используется, недопустимо обратиться к шине с более одной длиной. Специальная шина *output_bus* представляет обработку *second-to-finalmost* звукового потока. Это может быть отправлено только одному инструменту эффекта. Если инструмент самостоятельно направляется или используется *outbus* оператором, то это синтаксическая ошибка. Если *output_bus* не отправляется инструменту, он превращается в звук в конце цикла оркестра. Если *output_bus* отправляется инструменту, вывод этого инструмента превращается в звук в конце передачи оркестра. Этому инструменту не разрешают использовать *turnoff* оператор.

В случае если число каналов ввода, полученных инструментальным экземпляром, отличается от ширины шины *input* и *inGroup,* используют прежнее значение *inchan*. В случае если *inchan* меньше, чем число каналов на шинах, обеспечивающих ввод, используются только первые каналы *inchan.* В этом случае "дополнительные" каналы являются нулевыми и для *input* и для *inGroup*.

По крайней мере одно имя шины должно быть обеспечено в *send* инструкции.

8.5.6 Спецификация последовательности

<sequence specification> -> sequence (<identlist>) ;

<identlist>.

Оператор *sequence* позволяет выполнить спецификацию упорядочивания выполнения инструментальных инстанцирований планировщиком времени. Оператор *identlist* описывает частичное упорядочивание при наборе инструментов. Если на инструмент *a* и инструмент *b* ссылаются в том же самом операторе *sequence* с предыдущим *b*, то инстанцирования инструмента выполняются строго перед инстанцированиями инструмента *b*.

Есть несколько правил последовательности по умолчанию:

1. Специальный инструментальный *startup* инстанцируют, и инстанцирование *i-rate* выполняется в самом начале оркестра.

2. Любые инструментальные экземпляры, соответствующие инструменту *startup*, выполняются сначала в определенном цикле оркестра.

3. Если *output_bus* отправляется инструменту, инструментальное инстанцирование, соответствующее *send* оператору является последним инстанцированием, выполняемым в цикле оркестра.

4. Для каждого инструмента, направленного к шине, которая отправляется инструментом эффекта, инстанцирования направленного инструмента выполняются перед инстанцированиями инструмента эффекта. Если циклы создаются, используя *route* и *send* операторы, упорядочивание разрешается синтаксически. Если *send* оператор последний, то инструментальное инстанцирование выполняется последним.

Правила 2, 3 по умолчанию и 4 могут быть переопределены при помощи оператора *sequence*. Правило 1 не может быть переопределено.

Если операторы *sequence* создают циклы в упорядочивании, то это синтаксическая ошибка. Любые *send* операторы, которые являются "обратной" частью неявного *send* цикла не имеют никакого эффекта.

Если последовательность двух инструментов не определяется правилами последовательности по умолчанию, их инстанцирования могут быть выполнены в любом порядке или параллельно.

Невозможно определить упорядочивание многократных инстанцирований того же самого инструмента. Эти инстанцирования могут быть выполнены в любом порядке или параллельно.

8.6 Инструментальное определение

8.6.1 Синтаксическая форма

instrument definition> -> instr <ident> (<identlist>) [preset <int> [<int> ...]] {
<instr variable declarations> <block> }

У инструментального определения есть несколько элементов.

1. Идентификатор, который определяет имя инструмента,

2. Список нуля или большего количества идентификаторов, которые определяют имена для полей параметра, также названных *pfields* инструмента,

3. Дополнительный список предварительно установленных значений для того, чтобы определить *MIDI* отображения,

4. Список нуля или большего количества инструментальных объявлений переменной,

5. Блок операторов, определяющих исполнимую функциональность инструмента.

8.6.2 Инструментальное имя

Любой идентификатор может служить инструментальным именем за исключением того, что инструментальное имя не должно быть зарезервированным словом, именем базового кода операции или именем

базового генератора звуковой таблицы. Инструментальное имя может быть переменной в локальном или глобальном контексте.

Ни у каких двух инструментов или кодов операции в оркестре не должно быть одинакового имени.

8.6.3 Поля параметра

<identlist>->

Поля параметра инструмента, также названные *pfields*, являются интерфейсом, через который инстанцируют инструмент. В инструментальном коде у *pfields* есть семантика уровня *i-rate* локальных переменных. Их значения должны быть установлены на инструментальном инстанцировании, перед созданием локальных переменных, с соответствующими значениями как дано в строке счета, событии счета, событии *MIDI*, *send* операторе или *instr* операторе, соответствующих инструментальному инстанцированию.

8.6.4 Предварительно установленный тег

Тег *preset* определяет предварительно установленное число (ла) инструмента. Если изменения программы *MIDI* прибывают в потоке *MIDI* или файле *MIDI,* управляющем оркестром, то программы обращаются к данным *preset* тега различных инструментов. Только у одного инструмента может быть предварительно установленное одно и тоже число. Если многократные инструменты в оркестре определяют тот же самый *preset* тег, то синтаксически последниму присваивается предварительно установленное число. Если *preset* тег не связывается с определенным инструментом, то инструмент не имеет никакого предварительно установленного числа и не может быть сослан с изменением программы. Если дается больше чем один тег, то инструмент отвечает на все перечисленные предварительно установленные значения.

Предварительно установленные теги в *SAOL* соответствуют и предварительной установке и значению банка программы в управлении *MIDI.* Программа на предварительной установке *x*, банк *y* в синтаксисе *MIDI* должна быть обозначена как предварительная установка $(y - 1) * 128 + (x - 1)$ в *SAOL* (так как предварительные установки и банки нумеруются, запускаясь с 1 в *MIDI*).

8.6.5 Инструментальные объявления переменной

8.6.5.1 Синтаксическая форма

<instr variable declarations> -> *<instr variable declarations> <instr variable declaration> <instr variable declarations>*

<instr variable declarations> -> *<NULL>*

<instr variable declaration> -> *[<sharing tag>] ivar <namelist> ;*

<instr variable declaration> -> *[<sharing tag>] ksig <namelist> ;*

<instr variable declaration> -> *asig <namelist> ;*

<instr variable declaration> -> *<table declaration> ;*

<instr variable declaration> -> *<sharing tag> table <identlist> ;*

<instr variable declaration> -> *oparray <ident> [<array length>] ;*

<instr variable declaration> -> *<tablemap declaration> ;*

<sharing tag> -> *imports*

<sharing tag> -> *exports*

<sharing tag> -> *imports exports*

<tablemap declaration> -> *tablemap <ident> (<identlist>) ;*

Инструментальными объявлениями переменной объявляют переменные, которые могут использоваться в рамках инструмента. Любая переменная уровня, так же как звуковые таблицы *tablemaps* и заполнители звуковой таблицы, могут быть объявлены в инструменте. Инструментальное объявление переменной является или объявлением звуковой таблицы или именем типа, которому возможно предшествует тег совместного использования, сопровождаемый списком объявлений имени, или тегом совместного использования, сопровождаемым маркерной *table*, сопровождаемой списком идентификаторов, ссылающихся на глобальные или будущие звуковые таблицы, или объявление массива кода операции, или определение табличной карты.

8.6.5.2 Объявление звуковой таблицы

Синтаксис и семантика 8.5.3.3 содержат для инструмента локальные звуковые таблицы со следующими исключениями и дополнениями:

Инструмент, локальная звуковая таблица, доступен только в пределах локального контекста единственного инструментального инстанцирования. Это должно быть создано и инициализировано с данными в инструментальное время инстанцирования, сразу после того, как от параметров вызова будут присвоены значения *pfield* . Это может быть удалено и освобождено, когда инструментальное инстанцирование завершается.

Не каждое выражение *i-rate*, является законным, как часть табличного списка параметра. Ссылка на константы, *pfields*, импортированные *i-rate* переменные и *i-rate* стандартные имена разрешена. Инструментальная инициализация звуковой таблицы должна произойти через инструментальный код, и ссылка на локальные *i-rate* переменные запрещается.

8.6.5.3 Сигнальные переменные

Синтаксис и семантика 8.5.3.2 содержат для инструмента локальные сигнальные переменные со следующими исключениями и дополнениями:

Локальное объявление имени определяет, что должны создаваться маркер имени и пространство, равное значению единицы сигнала, выделенное для временного хранения в каждом инструментальном инстанцировании, связанном с инструментальным определением. Локальное объявление массива определяет, что должен создаваться маркер имени и пространство, равное конкретному количеству сигнальных значений, выделенных в каждом инструментальном инстанцировании, связанном с инструментальным определением.

Теги *imports* и/или *exports* могут использоваться совместно с локальным *i-rate* или переменной *k-rate*. Они не должны использоваться с *a-rate* переменными. Если тег *imports* будет использоваться, то переменное значение должно быть заменено значением глобальной переменной того же самого имени в инструментальное время инициализации (для *i-rate* сигнальных переменных) или в начале каждой передачи управления (для *k-rate* переменных). Тег *imports* может использоваться для локального *k-rate* переменной сигнала уровня, даже если нет никакой глобальной переменной того же самого имени. В этом случае показывают, что *k-rate* переменная уровня может быть изменена строками *control* в счете *SASL*. Тег *imports* не должен использоваться для локальных *i-rate* сигнальных переменных, если нет глобальной переменной того же самого имени.

Если тег *exports* будет использоваться, то значение глобальной переменной того же самого имени должно быть заменено значением локальной сигнальной переменной после инструментальной инициализации (для *i-rate* сигнальных переменных) или в конце каждой передачи управления (для *k-rate* переменных сигнала). Тег *exports* не должен использоваться, если нет никакой глобальной переменной того же самого имени.

Если для определенной сигнальной переменной теги *imports* и/или *exports* будут использоваться, и есть глобальная переменная с тем же самым именем, то ширина массива локальных и глобальных переменных должна быть одинакова.

Если для определенной локальной переменной тег *imports* не используется, то его значение устанавливается в 0 перед инструментальной инициализацией.

Если для определенного локального переменного объявления теги *imports* и *exports* не используются, даже если есть глобальная переменная того же самого имени, между этими двумя переменными нет никакого семантического отношения. Конструкция является синтаксически законной.

8.6.5.4 Заполнитель звуковой таблицы

Совместно теги *imports* и *exports* могут использоваться к ссылочным глобальным и будущим звуковым таблицам. В этом случае, локальное объявление табличной ссылки называют заполнителем звуковой таблицы. Определение заполнителя звуковой таблицы не содержит полное определение звуковой таблицы, а только ссылку на глобальное или будущее имя звуковой таблицы.

Если используется только тег *imports*, и есть глобальная звуковая таблица с тем же самым именем, то в инструментальное время инстанцирования текущее содержание глобальной звуковой таблицы копируется в локальную звуковую таблицу с тем же именем.

Если содержание глобальной звуковой таблицы изменяется после того как определенное инструментальное инстанцирование, ссылающееся на глобальную звуковую таблицу, создано, новое содержание глобальной звуковой таблицы не должно быть скопировано в инструментальное инстанцирование. Если содержание локальной звуковой таблицы будет изменено, то эти изменения не должны быть отражены в глобальной звуковой таблице.

Если теги *imports* и *exports* используются и есть глобальная звуковая таблица с тем же самым именем, то в инструментальное время инстанцирования и в начале каждой передачи управления текущее содержание глобальной звуковой таблицы делается доступным для локальной звуковой таблицы с тем же именем. "Делается доступным" в предыдущем предложении означает, что доступ может быть в форме копирования данных из одной звуковой таблицы в другую или ссылкой указателя на пространство памяти, или любой эквивалентной реализацией. В конце инструментального инстанцирования и в конце каждой передачи управления текущее содержание локальной звуковой таблицы так же делается доступным для

глобальной звуковой таблицы с тем же самым именем. В случае, если звуковая таблица изменяется в одном инструментальном экземпляре и не указано точно когда, эти изменения будут видимы в других инструментальных экземплярах, то это должно быть не позже, чем следующий цикл оркестра (разрешается быть в том же самом цикле оркестра).

Не допустимо использовать один только тег *exports* для заполнителя звуковой таблицы.

Если тег *imports* используется и нет никакой глобальной звуковой таблицы с тем же именем, то ссылаются на будущую звуковую таблицу, которая будет в потоке битов. Когда инструмент инстанцируют, содержание новой звуковой таблицы в потоке битов с тем же именем должно быть скопировано в локальную звуковую таблицу. Если никакая звуковая таблица в потоке битов с тем же самым именем во время инструментального инстанцирования не была заполнена, то поток битов недопустим. Если звуковая таблица изменяется, обеспечивая новую звуковую таблицу с тем же самым именем при использовании *table* строки в потоке битов, то ссылка сразу изменяется на новую звуковую таблицу.

Не допустимо использовать тег *exports*, если нет никакой глобальной звуковой таблицы с тем же самым именем.

8.6.5.5 Объявление массива кода операции

Массив кода операции или *oparray* объявление объявляет несколько состояний кода операции, которые могут использоваться текущим инструментом или кодом операции. Объявляя состояния этим способом, доступ к ним доступен через *oparray* выражение. Идентификатор в объявлении должен быть именем базового кода операции или определен пользователем кода операции в другом месте в оркестре. Длина массива объявляет сколько состояний имеется для доступа к *oparray* в локальном блоке кода. Это должно быть целочисленное значение или специальные теги *inchannels* или *outchannels*.

Если больше чем одно *oparray* объявление ссылается на то же самое имя кода операции в инструменте или коде операции, то это — синтаксическая ошибка.

8.6.5.6 Табличное определение карты

<table map definition> -> tablemap <ident> (<identlist>)

Табличная карта является структурой данных, позволяющей косвенную ссылку звуковых таблиц через нотацию массива. Идентификатор называет табличную карту, и это не должно быть именем любой другой сигнальной переменной или другого ограниченного слова в локальном контексте. Список идентификаторов дает много имен звуковой таблицы для использования с табличной картой. Каждое из этих имен должно соответствовать определенному заполнителю или звуковой таблицы в пределах текущего контекста. *Tablemap* объявление может прибыть прежде, после, или посреди объявлений звуковой таблицы и заполнителей звуковой таблицы в инструменте. На все звуковые таблицы в пределах инструмента можно сослаться в *tablemap* независимо от синтаксического размещения *tablemap*.

Когда имя *tablemap* используется в ссылочном массивом выражении, индекс выражения определяет к какой из звуковых таблиц в списке обращается выражение. Первая звуковая таблица в списке является номером 0, вторым номером 1 и так далее.

8.6.6 Операторы блока программы

8.6.6.1 Синтаксическая форма

<block> -> <statement> [<block>]
<block> -> <NULL>
<statement> -> <lvalue> = <expr> ;
<statement> -> <expr> ;
<statement> -> if (<expr>) { <block> }
<statement> -> if (<expr>) { <block> } else { <block> }
<statement> -> while (<expr>) { <block> }
statement> -> instr <ident> (<expr list>) ;
<statement> -> output (<expr list>) ;
<statement> -> spatialize (<expr list >) ;
<statement> -> outbus (<ident> , <expr list>) ;
<statement> -> extend (<expr>) ;
<statement> -> turnoff ;
<expr list> -> <expr> [, <expr list>]
<expr list> -> <NULL>

Блок является последовательностью нуля или большего количества операторов. Оператор должен принять одну из нескольких форм, которые перечисляются и описываются в последующих подпунктах.

У каждого оператора есть правила уровней семантики, управляющих уровнем оператора, правила контекстов уровня, в которых это допустимо и времена, в которые должны выполняться различные субкомпоненты.

Чтобы выполнить блок операторов на определенном уровне, каждый оператор в пределах блока должен выполняться в таком порядке, чтобы это привело к выполнению операторов последовательно в линейном порядке.

8.6.6.2 Присвоение

<lvalue> -> *<ident>*

<lvalue> -> *<ident> [<expr>]*

lvalue или значение левой стороны обозначает сигнальную переменную или переменные, значения которых должны быть изменены. *lvalue* может быть локальным именем переменной, когда обозначаются места хранения, связанные с этим именем. *lvalue* может также быть локальным именем массива, когда обозначается место для хранения массива. *lvalue* может также быть единственным элементом локального массива, обозначенным индексом имени локального массива. *lvalue* не должен быть табличной ссылкой или *tablemap* выражением. *lvalue* не должен быть стандартным именем кроме случаев *MIDIctrl* или *params*.

Если *lvalue* обозначит весь массив, то выражение правой стороны присвоения должно обозначаться массивом с той же длиной или единичным значением, иначе конструкция синтаксически недопустима. В случае, если *lvalue* зависит от вычисления выражения правой стороны (например, если весь массив умножается с одним из его элементов), *lvalue* неопределен.

Если *lvalue* обозначает единичное значение, то выражение правой стороны присвоения должно обозначить единичное значение, иначе конструкция синтаксически недопустима.

Величина *lvalue* является величиной сигнальной переменной, если нет индексации, или величиной сигнального массива, обозначенного сигнальной переменной и уровнем индексации, если есть индексация.

Уровень оператора является уровнем *lvalue*, однако, оператор недопустим, если уровень правой стороны быстрее, чем уровень *lvalue*.

Присвоение должно выполняться следующим образом:

В каждой передаче через оператор, происходящей на равных уровнях присвоения, должно быть оценено правое выражение стороны. Затем место для хранения, обозначенное *lvalue*, должно быть обновлено, чтобы быть равным значению правого выражения. Если *lvalue* обозначит весь массив, и выражение правой стороны единичное значение, то каждое из значений каждого из элементов массива должно быть изменено на единичное значение правой стороны.

8.6.6.3 Нулевое присвоение

<statement> -> *<expr>*

Нулевое присвоение содержит только выражение. Это обеспечивают коды операции, у которых нет полезных возвращаемых значений, и они не использовались в контексте присвоения фиктивной переменной.

Уровень оператора является уровнем выражения. Выражение может быть однозначным или оценено массивом. Это не должна быть табличная ссылка.

Нулевое присвоение должно выполняться следующим образом:

В каждой передаче через оператор, происходящей на равных уровнях присвоения, должно быть оценено выражение.

8.6.6.4 *If*

<statement> -> *if (<expr>) { <block> }*

If оператор позволяет условную оценку блока программы. Выражение, которое тестируется в *if* операторе называют выражением защиты.

Уровень оператора является уровнем выражения защиты или уровнем самого быстрого оператора в защищенном блоке кода.

Не допустимо для блока программы, которым управляет *if* оператор, содержать операторы медленнее, чем выражение защиты. Недопустимо для любого из операторов в блоке программы, которым управляют, содержать коды операции, которые были бы выполнены медленнее, чем выражение защиты. Выражение защиты должно быть однозначным выражением.

If оператор должен выполняться следующим образом:

В каждой передаче через оператор, происходящей на равных уровнях присвоения, должно быть оценено выражение защиты. Если оператор защиты в определенной передаче оценен ненулевым значением, то блок программы должен быть оценен на уровне, соответствующем той передаче.

Если в блоке, выполняющийся в *a-rate* или *k-rate,* есть *i-rate* операторы, эти операторы должны быть выполнены только в первый раз.

Если в блоке, выполняющейся в *a-rate,* есть операторы *k-rate*, эти операторы должны быть выполнены только в первый раз, когда блок выполняется в *kcycle*.

8.6.6.5 *Else*

<statement> -> if (<expr>) { <block> } else { <block> }

Else оператор позволяет дизъюнктивую оценку двух блоков программы. Выражение, которое тестируется в операторе, называют выражением защиты.

Уровень оператора является уровнем выражения защиты, или уровнем самого быстрого оператора в первом защищенном блоке программы, или уровнем самого быстрого оператора во втором защищенном блоке программы.

Else недопустимо для блоков программы, которым оператор *else* приказывает содержать более медленные операторы, чем выражение защиты. Недопустимо для любого из операторов в блоках программы, которыми управляют, содержать коды операций, которые были бы выполнены медленнее, чем выражение защиты. Выражение защиты должно быть однозначным выражением.

Else оператор должен выполняться следующим образом:

В каждой передаче через оператор, происходящей на равных уровнях присвоения, должно быть оценено выражение защиты. Если выражение защиты в определенной передаче оценено ненулевым значением, то первый защищенный блок программы должен быть на уровне соответствующем этой передаче. Если оператор защиты в определенной передаче обнулить, то каждый оператор во втором защищенном блоке программы должен быть обнулен.

Если в блоке программы, выполняющейся в *a-rate* или *k-rate,* есть *i-rate* операторы, эти операторы должны быть выполнены только в первый раз, когда блок выполняется .

Если в блоке программы, выполняющейся в *a-rate,* есть *k-rate* операторы, эти операторы должны быть выполнены только в первый раз, когда блок выполняется в *kcycle*.

8.6.6.6 *While*

<statement> -> while (<expr>) { <block> }

While оператор позволяет блоку программы быть условно оцененным несколько раз в одном уровне передачи. Выражение, которое тестируется в *while* операторе называют выражением защиты.

Уровень *while* оператора является уровнем выражения защиты.

Не допустимо для блока программы, которым управляет *while* оператор, содержать операторы, которые работают на уровне, кроме уровня выражения защиты. Не допустимо для любого из операторов в блоке программы, которым управляют, содержать коды операции, которые выполняются на уровне, кроме уровня выражения защиты. Выражение защиты должно быть однозначным выражением. Не допустимо для выражения защиты содержать базовые коды операции *specialop*.

While оператор должен выполняться следующим образом:

В каждой передаче через оператор, происходящей на равных уровнях присвоения, должно быть оценено выражение защиты. Если выражение защиты в определенной передаче оценено в ненулевое значение, то каждый оператор в защищенном блоке программы должен быть оценен согласно определенным правилам для этого оператора, и затем выражение защиты переоценивают, выполняя итерации, пока выражение защиты не обнулится.

8.6.6.7 *Instr*

<statement> > instr <ident> (<expr list>);

Instr оператор позволяет инструментальному инстанцированию динамически создавать другие инструментальные инстанцирования для методов синтетической производительности или иерархического представления. Он состоит из идентификатора, обращающегося к инструменту, определенному в текущем оркестре, задержки, продолжительности и списка выражений, определяющих параметры инструмента на который ссылаются.

Если число выражений в списке выражения не два или больше чем число *pfields*, принятого инструментом на который ссылаются, то это синтаксическая ошибка, Каждое выражение в списке выражения должно быть однозначным выражением.

Уровень *instr* оператора является уровнем самого быстрого выражения в списке выражения или уровнем выражения защиты, или уровенем кода операции.

Instr оператор должен выполняться следующим образом:

В каждой передаче через оператор, происходящей на равных уровнях присвоения, оценивается каждое из выражений в списке выражения. Новое инструментальное событие регистрируется в планировщике. Время нового инструментального события является суммой текущего времени оркестра и величины первого выражения в списке выражения масштабируемого текущим глобальным темпом. Продолжительность нового инструментального события является значением второго выражения в списке выражения. Значения *p* полей для нового инструментального события являются значениями остающихся выражений в списке выражения. У значений первых и вторых выражений есть модули ударов счета, и они масштабируются согласно фактическому темпу оркестра.

Исключение происходит, когда задержка (первое выражение в списке выражения) меньше, чем продолжительность периода управлением оркестра. В этом случае, инструментальное событие не создается, но новое инструментальное инстанцирование было сразу создано. Продолжительность нового инстанцирования является значением второго выражения в списке выражения, и значения инструментальных *p*-полей в новом инстанцировании устанавливаются в значения остающихся выражений.

В этом случае передача *i-rate,* посредством нового инстанцирования инструмента должна выполняться непосредственно после его создания, прежде, чем выполняются операторы от блока программы, содержащего *instr* оператор. Любые изменения в глобальные переменные *i-rate*, сделанные в новом экземпляре во время передачи *i-rate*, не признаются в этом инструменте ("*caller*") (переменные *i-rate*, импортированные из глобального контекста, устанавливаются только во время передачи инициализации каждого экземпляра и никогда не изменяются позже).

У динамически создаваемого инструмента есть доступ к *MIDIctrl* (8.6.8.9), *MIDItouch* (8.6.8.10), *MIDIbend* (8.6.8.11), *channel* (8.6.8.12) и *preset* (8.6.8.13) стандартному имени его предшественника. Динамически создаваемый инструмент не планируется для завершения, когда предшественник будет завершен под управлением *MIDI*.

8.6.8.8 *Output*

<statement> -> output (<expr list>);

Output оператор создает аудиовыход из инструмента. Этот вывод не становится превращенным непосредственно в звук, а скорее буферизуется на одной или более шинах, основанных по инструкциям данных в операторах *route* или на специальной шине *output_bus* по умолчанию. Если текущее инструментальное инстанцирование является *send* оператором, ссылающимся на специальную шину *output_bus*, то вывод текущего инстанцирования суммирует его так, чтобы *output* мог быть превращен непосредственно в звук.

Список выражения должен содержать по крайней мере одно выражение.

Уровнем *output* оператора является *a-rate*.

Все операторы оркестра, которые ссылаются на шину *outbus*, или маршрутизируются по умолчанию к специальной шине *output_bus*, будут иметь совместимое число параметров выражения, представляющего каналы вывода. "Совместимый" означает что, если *output* для определенной шины больше, чем один параметр выражения, то у всех других *output* на эту шину должно быть то же самое число параметров выражения или только один параметр выражения. Число каналов специальной шины *output_bus* должно быть тем же самым, как параметр у глобальной переменной *outchannels* и *output,* используемое инструментами, и будет совместимым с этим числом каналов.

Output оператор выполняется следующим образом:

В каждом *k-rate* проходят через выходной буфер с числом каналов, определенных правилами в 7.3.3.5.2, чтобы обнулить значения. Каждый уровень проходит через оператор, и параметры выражения должны быть оценены. Значения параметра выражения должны быть помещены в выходной буфер. Если значение у *output* оператора будет больше, чем одно выражение параметра, то значение каждого параметра должно быть добавлено к текущему значению выходного буфера в соответствующем канале. Если у *output* оператора будет только одно выражение параметра, то значение этого выражения должно быть добавлено к текущему значению выходного буфера в каждом канале.

Параметры выражения *output* оператора могут быть оценены массивом, в котором отображение, описанное в предыдущем абзаце, не от выражений, чтобы буферизовать каналы, а от каналов значения массива, чтобы буферизовать каналы.

Каждый уровень проходит через инструментальное инстанцирование во время, определенное оркестром. Значения в выходном буфере должны быть добавлены каналом к текущему значению шины или шин,

на которые ссылается выражение *route* или выражения, на которые также ссылается этот инструмент. Если не будет операторов *route*, то значения в выходном буфере должны быть добавленны каналом к текущему значению специальной шины *output_bus*. Если это инструментальное инстанцирование, создаваемое ссылкой на специальную шину *output_bus* в *send* операторе, то значения в выходном буфере являются выводом оркестра.

8.6.6.9 *Spatialize*

<statement>-> spatialize (<expr list>);

Spatialize оператор позволяет инструментам производить звук *spatialised*, используя ненормативные методы, которые зависят от реализации.

Список выражения должен содержать четыре выражения. Второе, третье и четвертое не должны быть уровенем выражения. Первое выражение представляет аудиосигнал *spatialised*. Второе, азимут (угол), из которого исходит звук, измеряется в радианах по часовой стрелке от 0 непосредственно перед слушателем. Третье, угол возвышения, откуда исходит звук, измеряется в радианах вверх от 0 на горизонтальном месте слушателя. Четвертое, расстояние от источника звука, измеряется в метрах от позиции слушателя. Каждое из этих четырех выражений должно быть однозначным.

Уровень *spatialize* оператора является *a-rate*.

Spatialize оператор должен выполняться следующим образом:

В каждом уровень проходит через инструмент, выражения в списке выражения должны быть оценены. Звуковой сигнал в первом выражении должен быть представлен слушателю, как если бы он исходил от азимута, повышения, и расстояния, данных во вторых, третьих и четвертых выражениях. Никакие нормативные требования не распространяются на эту *spatialisation* возможность.

Звук, произведенный через *spatialize* оператор, превращается непосредственно в вывод оркестра. На это не должны влиять маршрутизации шины или дальнейшее манипулирование в пределах оркестра. Если *spatialize* многократно повторяются в пределах оркестра, то различные звуки должны быть смешаны через простое суммирование после *spatialisation*. Если звук *spatialised* и *non-spatialised* будет произведен в пределах оркестра, то вывод всего звука *non-spatialised* должен быть смешан через простое суммирование с различными звуками *spatialised*. Чтобы включить это смешивание звука, произведенного через каждый *spatialize* оператор, количество каналов должно быть равно глобальному числу каналов вывода оркестра.

8.6.6.10 *Outbus*

<statement>-> outbus (<ident>, <expr list>);

Outbus оператор позволяет инструментам помещать динамически расчетные сигналы в шины. Параметр идентификатора должен обратиться к имени шины, определенной с помощью *send* оператора в глобальном блоке.

Если нет никаких выражений в списке выражения, или если идентификатор не обращается к шине, определенной в глобальном блоке с помощью *send* оператора, то это синтаксическая ошибка. Число выражений в списке выражения должно быть совместимо с другими операторами, ссылающимися на эту же шину.

Уровень *outbus* оператора является *a-rate*.

Outbus оператор должен выполняться следующим образом:

Каждый уровень проходит через оператор, список выражения должен быть оценен. Значения выражения должны быть добавлены к текущему значению шины, на которую ссылаются. Если будет больше, чем одно выражение в списке выражения, то каждое значение выражения должно быть добавлено к соответствующему каналу шины, на которую ссылаются. Если будет только одно выражение в списке выражения, то значение выражения должно быть добавлено к каждому каналу шины, на которую ссылаются.

Выражения в списке выражения могут быть оценены массивом.

Outbus оператор не должен использоваться в инструменте, который является целью *send* оператора, ссылающегося на специальную шину *output_bus*.

8.6.6.11 *Extend*

<statement>-> extend (<expr>);

Extend оператор позволяет инструментальному инстанцированию динамически удлинять свою продолжительность.

Параметр выражения не должен быть уровнем. Выражение должно быть однозначным.

Уровень *extend* оператора является уровнем параметра выражения.

Extend оператор должен выполняться следующим образом.

Каждый проходит через оператор на одном уровне с уровнем оператора, выражение должно быть оценено. Продолжительность инструментального инстанцирования должна быть расширена количеством времени, в секундах, данных значением выражения.

Extend оператор не должен выполняться в инструменте, который создается как результат *send* оператора, ссылающегося на специальную шину *output_bus*.

Когда e*xtend* оператор вызовут, стандартное имя *dur* должно быть обновлено, чтобы отразить новую продолжительность. То есть *dur*: = *dur* + *x*, где *x* является значением выражения параметра.

8.6.6.12 *Turnoff*

<statement> -> turnoff;

Turnoff оператор позволяет инструментальному инстанцированию динамически завершить себя.

Уровень *turnoff* оператора является *k-rate*.

Turnoff оператор должен выполняться следующим образом.

Когда *turnoff* оператор достигнет *k-rate*, инструментальный экземпляр завершится после следующего *k-rate*. То есть, если текущее время оркестра будет *T* и *k* продолжительность передачи, то инструментальное инстанцирование завершится во время *T+k*.

Turnoff оператор не должен обновлять *dur* стандартное имя.

Turnoff оператор не должен выполняться в инструментальном экземпляре, который создается как результат *send* оператора, ссылающегося на специальную шину *output_bus*.

Примечание — *turnoff* сразу не уничтожает инстанцирование. Инстанцирование выполняется для еще одной передачи оркестра, чтобы позволить инструментальному времени исследовать *released* переменную.

8.6.7 Выражения

8.6.7.1 Синтаксическая форма

<expr>-> <ident>
<expr>-> <number>
<expr>-> <int>
<expr>-> <ident> [<expr>]
<expr>-> <ident> (<expr list>)
<expr>-> <ident> [<expr>] (<expr list >)
<expr>-> <expr>? <expr>: <expr>
<expr>-> <expr> <binop> <expr>
<expr>->! <expr>
<expr>-> - <expr>
<expr>-> (<expr>)
<expr>-> sasbf (<expr list >);
<binop>-> +
<binop>-> -
*<binop>-> **
<binop>-> /
<binop>-> ==
<binop>->> =
<binop>-> <=
<binop>->! =
<binop>->>
<binop>-> <
<binop>-> &&
<binop>-> ||

Выражение может принять одну из нескольких форм. У каждой формы есть два уровня семантики, которые описывают уровень выражения с точки зрения уровней подвыражений, и оценивают семантику, которую описывают значения выражения с точки зрения значений подвыражений.

8.6.7.2 Свойства выражений

Каждое выражение концептуально маркируется двумя свойствами: его уровем и шириной. Уровень выражения определяет, как быстро значение этого выражения могло бы измениться. Ширина выражения определяет, сколько каналов звука или других данных представляется выражением. В каждом типе выражения уровень и ширина выражения зависят от типа выражения и возможно уровня и ширины компонентных подвыражений.

8.6.7.3 Идентификатор

<expr> -> *<ident>*

Выражение идентификатора обозначает место хранения или место, которое содержит значения, сохраненные в памяти. Нельзя ссылаться на идентификатор, который не появляется в локальном инструменте или контексте кода операции.

Уровень выражения идентификатора является типом уровня, в котором идентификатор был объявлен, или неявно объявляется в случае стандартных имен.

Уровень *table* идентификатора *i-rate*.

Если идентификатор обозначен единичным именем (то есть не является типом массива), то значение выражения идентификатора является значением, сохраненным в памяти, связанной с этим идентификатором в текущем контексте, и ширина выражения равна 1.

Если идентификатор обозначен именем массива, то значение выражения идентификатора является упорядоченной последовательностью значений, сохраненных в памяти и связанных с идентификатором в текущем контексте, и ширина выражения является шириной массива.

Если идентификатор обозначен таблицей, то значение выражения идентификатора является ссылкой на таблицу с именем. У табличной ссылки ширина 1.

8.6.7.4 Постоянная величина

<expr>-> <number>

<expr>-> <int>

Выражение постоянной величины обозначается одним числом.

Уровень выражения постоянной величины *i-rate*.

Ширина выражения постоянной величины равна 1.

Значение постоянного выражения является значением числа, обозначенного константой. Значение постоянного выражения всегда является значением с плавающей точкой.

8.6.7.5 Ссылка массива

<expr> -><*ident*> [<*expr*>]

Ссылочное выражение массива позволяет сделать выбор одного значения из нескольких значений массива. Идентификатор массива в синтаксисе называют именем массива, а выражение — индексным выражением. Недопустимо использовать идентификатор в ссылке массива, которая не объявлена в локальном инструменте или контексте кода операции массива, или явно не определены как оцененное массивом стандартное имя или табличная карта.

У индексного выражения ширина равна 1.

Уровень ссылочного выражения массива является уровнем имени массива, или уровенем индексного выражения.

Ширина ссылочного выражения массива равна 1.

Если массив, на который ссылаются, является массивом, оцененным сигнальной переменной, то значение ссылочного выражения массива является значением того элемента последовательности значений в памяти массива, соответствующего значению выражения индексации, где элемент 0 соответствует первому значению в последовательности. Это — ошибка времени выполнения, если значение выражения индексации меньше 0, равно или больше, чем объявленный размер массива. Если выражение индексации не является целым числом, то оно округляется к самому близкому целому числу.

Если массив, на который ссылаются, является табличной картой, то значение ссылочного выражения массива является ссылкой на тот элемент последовательности таблиц, соответствующих значению индексного выражения, где элемент 0 соответствует первой таблице в последовательности. Это — ошибка времени выполнения, если значение выражения индексации — меньше чем 0, равно или больше, чем объявленный размер табличной карты. Если выражение индексации не является целым числом, то оно округляется к самому близкому целому числу. Табличные ссылки могут появиться только в кодах операции.

8.6.7.6 Вызов кода операции

<expr>-> <ident> (<expr list>)

Выражение вызова кода операции позволяет использование обработки функциональности, инкапсулировавшей в пределах кода операции.

Идентификатор называют именем кода операции, и выражение перечисляет фактические параметры выражения вызова кода операции. Недопустимо использовать идентификатор, который не является именем базового кода операции и не является именем, определенным пользователем кода операции. Для определяемых пользователем кодов операции число фактических параметров должно быть таким же, как

число формальных параметров в определении кода операции. Для базовых кодов операции без переменных списков параметров, число фактически требуемых параметров изменяется от кода операции до кода операции. Если определенный формальный параметр в определении кода операции будет массивом, то соответствующий фактический параметр должен быть введенным массивом выражением равной ширины. Если определенный формальный параметр в определении кода операции будет таблицей, то соответствующий фактический параметр должен быть табличной ссылкой.

Если определенный формальный параметр в определении кода операции будет на определенном уровне, то соответствующее фактическое выражение параметра не должно быть на более высоком уровне.

Ширина выражения вызова кода операции является числом каналов, обеспеченных операторами *return* в блоке кода операции.

Для вызова базовых кодов операции, это — синтаксическая ошибка, если какой-либо из следующих операторов применяется:

- фактических параметров в обращении кода операции меньше, чем необходимых формальные параметры;

- фактических параметров в обращении кода операции больше, чем необходимые и дополнительные формальные параметры, и определение кода операции не включает *varargs* "..." пункт;

- определенное фактическое выражение параметра имеет более высокий уровень, чем соответствующий формальный параметр, или чем *varargs* формальный параметр, если это – корреспонденция;

- определенное фактическое выражение параметра не является единичным, или не является табличным, когда соответствующий формальный параметр определяет таблицу.

Контекст вызова кода операции ограничивается больше, чем другие выражения. В пределах блока с оператором защиты (*if*, *else* или в *while*) у вызовов кода операции не должно быть уровня ниже чем уровень выражения защиты. Вызов кода операции с определенным именем не должен происходить ни в пределах блока кода определения этого кода операции, ни в пределах блоков кода ни одного из кодов операции, вызванных этим кодом операции, ни одним из кодов операции, вызванных ими, и т.д. Таким образом, запрещаются рекурсивные и взаимно рекурсивные коды операции.

Если *kopcode* вызов кода операции происходит в выражении, которое работает в *a-rate*, в первый раз, когда это выражение выполняется в *k* цикле кода операции, *kopcode* вызывают после семантики, описанной в этом подпункте. Для всех последующих оценок выражения в том же самом *k* цикле не выполняется *kopcode*. Вместо этого исходное значение от первого выполнения используется в оценке выражения.

Если *iopcode* вызов кода операции происходит в выражении, которое работает в *a-rate* или *k-rate*, в первый раз, когда код операции выполняет это выражение, *iopcode* вызывают после семантики, описанной в этом подпункте. Для всех последующих оценок выражения не выполняется *iopcode,* а вместо этого исходное значение от первого выполнения используется в оценке выражения.

Если *specialop* вызов кода операции происходит в выражении, которое работает в *a-rate*, семантика *k-rate specialop* вызова кода операции следует по правилам для вызовов *kopcode*, в то время когда семантика *a-rate specialop* вызова кода операции происходит в каждом цикле.

Чтобы вычислить значение выражения вызова кода операции, ссылающегося на определяемый пользователем код операции на определенном уровне, значения фактического выражения параметра должны быть вычислены в порядке их появления в списке выражения. Значения формальных параметров в пределах контекста кода операции должны быть установлены в значения соответствующих фактических выражений параметра. Если это будет первым выражением вызова кода операции, ссылающимся на этот контекст кода операции, то место для хранения кода операции должно создаваться так, чтобы сохранить локальные сигнальные переменные и звуковые таблицы. Любые глобальные переменные, импортированные кодом операции на этом уровне, должны быть скопированы в место для хранения кода операции. Блок оператора кода операции должен выполняться по действующему значению. Значение выражения вызова кода операции является значением первого оператора *return,* с которым встречаются выполняя код операции. Значение выражения вызова кода операции может быть оценено массивом. После выполнения кода операции любые глобальные переменные, экспортируемые кодом операции, должны быть скопированы в глобальное место для хранения.

Если определенное фактическое выражение параметра в выражении вызова кода операции является идентификатором или ссылочным массивом выражения, то этот параметр является ссылочным параметром в этом обращении кода операции. Когда блок оператора кода операции будет выполнен, заключительное значение формального параметра, связанного с этим фактическим параметром, должно быть скопировано в переменное значение, обозначенное идентификатором или ссылкой массива, если параметр не является стандартным именем, которое не может использоваться в качестве *lvalue*. Эта модификация

должна произойти сразу после завершение блока оператора, и прежде чем делаются другие вычисления. Единичное значение и выражение значения массива могут быть ссылочными параметрами. Если оцененное массивом выражение будет использоваться, то связанный формальный параметр должен быть массивом той же самой длины.

Чтобы вычислить значение выражения вызова кода операции, ссылающегося на код операции на определенном уровне, значения фактических выражений параметра должны быть вычислены в порядке их появления в списке выражения. Исходное значение базового кода операции должно быть вычислено согласно правилам для определенного кода операции. Если уровень формального параметра ниже, чем уровень кода операции, то применяют следующие правила:

- в коде операции заходят в *k-rate*, фактические переменные, связанные с *ivar* формальными параметрами, обновляются только в первый раз, когда выполняется код операции;

- в коде операции заходят в *a-rate*, фактические переменные, связанные с *ivar* формальными параметрами обновляются только в первый раз, когда выполняется код операции. Фактические переменные, связанные с *ksig* формальными параметрами, обновляются только в первый раз, когда этот код операции выполняется в *k* цикле кода операции.

Примечание — Переменные, объявленные в рамках определяемого пользователем кода операции, оцениваются статически. То есть, они сохраняют свои значения от вызова до вызова. Значения переменных в рамках определяемого пользователем кода операции устанавливаются в 0 прежде, чем код операции вызовут в первый раз. Каждый синтаксически отличный вызов кода операции создает только один контекст кода операции.

8.6.7.7 Вызов *oparray*

<expr>-> <ident> [<expr>] (<expr list>)

Выражение вызова *oparray* позволяет динамически делать выбор состояния кода операции из нескольких.

Идентификатор называют именем кода операции, выражение в скобках называют индексным выражением, а выражения в списке параметра называют фактическими параметрами. Недопустимо использовать идентификатор, который не является именем базового кода операции и также не является именем, определенным пользователем кода операции. Недопустимо использовать идентификатор, для которого *oparray* хранение не выделяется в локальном контексте. Для определяемых пользователем кодов операции число фактических параметров должно быть тем же самым, как число формальных параметров в определении кода операции. Для ядра число фактически требуемых параметров изменяется от кода операции до кода операции.

Индексное выражение должно быть единичным выражением.

Уровень выражения вызова *oparray* является уровнем кода операции, на который ссылаются. Уровень индексного выражения не должен быть выше, чем уровень кода операции, на который ссылаются.

Ширина выражения вызова *oparray* является числом каналов, возвращенных операторами *return* в пределах блока кода операции.

Контекст выражения вызова *oparray* ограничивается таким же образом, как описано для выражения вызова кода операции в 8.6.7.6. Семантика уровня для выполнения вызова *oparray* по тем же правилам, описанным для выражения вызова кода операции в 8.6.7.6.

Значение выражения вызова *oparray* определяется таким же образом, как описано для вызова кода операции в 8.6.7.6, со следующими исключениями и дополнениями.

Прежде, чем значения фактических выражений параметра вычисляются, вычисляется значение индексного выражения. Это — ошибка времени выполнения, если значение индексного выражения не находится в диапазоне [0.. *n*-1], где *n* является размером выделения в *oparray* определении для этого *oparray*. Если индексное выражение не является целым числом, оно округляется к самому близкому целому числу. Хранение контекста, связанного с именем кода операции и значением индексного выражения, выбирается из набора контекстов *oparray* в локальном контексте. В пределах каждого контекста *oparray* локальные переменные сохраняют свои значения от вызова до вызова.

8.6.7.8 Комбинация векторных и скалярных элементов в математических выражениях

Для каждого математического выражения ширина выражения является максимальной шириной любого из ее подвыражений. У каждого подвыражения, в пределах выражения, должна быть та же самая ширина, или равна 1.

8.6.7.9 Переключатель

<expr>-> <expr>? <expr>: <expr>

Выражение переключателя комбинирует значения из двух подвыражений, основанных на значении третьего.

Уровень выражения переключателя является уровнем самого первого из этих трех подвыражений.

Значение выражения переключателя вычисляется следующим образом. Три подвыражения оцениваются. Для каждого значения первого подвыражения, если это значение является ненулевым, соответствующее значение выражения переключателя является соответствующим значением второго подвыражения. Если это значение является нулем, соответствующее значение выражения переключателя является соответствующим значением третьего подвыражения.

В особом случае, когда у всех подвыражений ширина 1, тогда выражение переключателя "закорачивает": первое подвыражение оценивается, и если его значение является не нулевым, то оценивается второе подвыражение, и его значение является значением выражения переключателя. Если значение первого подвыражения является нулем, то оценивается третье подвыражение, и его значение является значением выражения переключателя.

8.6.7.10 *Not*

<expr> ->! <expr>

Выражение *not* выполняет логическое отрицание в подвыражении.

Уровень выражения *not* является уровнем подвыражения.

Значение выражения *not* вычисляется следующим образом: подвыражение оценивается. Для каждого не нулевого значения в подвыражении соответствующее значение выражения *not* является нулем. Для каждого нулевого значения в подвыражении соответствующее значение выражения *not* равно 1.

8.6.7.11 Отрицание

<expr>-> <expr>

Выражение отрицания выполняет арифметическое отрицание в подвыражении.

Уровень выражения отрицания является уровнем подвыражения.

Значение выражения отрицания должно быть вычислено следующим образом: подвыражение оценивается. Для каждого значения в подвыражении соответствующее значение выражения отрицания является арифметическим значением отрицания.

8.6.7.12 Бинарные операторы

<expr> -> <expr> <binop> <expr>

Существуют 12 бинарных операторов. Каждый из них вычисляет различную функцию в двоичном подвыражении.

Значение выражения должно быть вычислено следующим образом. Два подвыражения должны быть оценены. Соответствующее значение двоичного выражения должно быть вычислено согласно таблицы 1, где x_1 и x_2 являются значениями первых и вторых подвыражений:

Таблица 1 — Бинарные операторы

Оператор	Значение выражения
+	$x_1 + x_2$
-	$x_1 - x_2$
*	$x_1 x_2$
/	x_1 / x_2
= =	если $x_1 = x_2$, то 1, иначе 0
>	если $x_1 > x_2$, то 1, иначе 0
<	если $x_1 < x_2$, то 1, иначе 0
<=	если $x_1 \le x_2$, то 1, иначе 0
> =	если $x_1 \ge x_2$, то 1, иначе 0
! =	если $x_1 \ne x_2$, то 1, иначе 0

Для "логического и" оператора && в особом случае, когда у обоих подвыражений ширина 1, выражение вычисляется способом "закорачивания". Первое подвыражение должно быть оценено. Если его значение 0, то значение выражения 0. Если его значение будет не нулевым, то второе подвыражение должно быть оценено, и если его значение 0, то значение выражения 0, иначе значение выражения 1.

Для "логического или" оператора || в особом случае, когда у обоих подвыражений ширина 1, выражение вычисляется способом "закорачивания". Первое подвыражение должно быть оценено. Если его значение является не нулевым, то значение выражения 1. Если его значение будет 0, то второе подвыражение должно быть оценено, и если его значение является ненулевым, то значение выражения 1, иначе значение выражения 0.

8.6.7.13 Круглая скобка

<expr> -> (<expr>)

Оператор круглая скобка не выполняет нового вычисления, но позволяет создать спецификацию арифметической группировки.

Уровень выражения круглая скобка является уровнем подвыражения.

Ширина выражения круглая скобка является шириной подвыражения.

Значение выражения круглая скобка является значением подвыражения.

8.6.7.14 Порядок операций

Выражения связывают в порядке предписанном в таблице 2. Операции, перечисленные в таблице 1, выполняются перед операциями в таблице 2 всякий раз, когда упорядочивание синтаксически неоднозначно. Операции, перечисленные в первой и последней строках, выполняются справа налево. Операции, перечисленные в оставшихся строках, слева направо.

Таблица 2 — Порядок операций

Оператор	Функции
!, -	отрицание, унарные операции
*, /	умножение, деление
+, -	сложение, вычитание
<, >, <=, >=	относительно
==, !=	равно
&&	логическое и
||	логическое или
?:	переключение

8.6.7.15 Синтез *SASBF*

<expr> -> sasbf (<expr list>);

Выражение *sasbf* позволяет использование *DLS,* процедуру синтеза банка совместно с инструментом *SAOL*. Это не должно использоваться в объектном потоке битов типа 3, и возможности выполнения не должны обеспечиваться объектным декодером типа 3. Список параметра должен иметь два, три, или четыре выражения. Они должны быть единичными выражениями *i- rate*.

1. Первое требуемое для синтеза выражение должно соответствовать *MIDI*. Если это значение не целое число, то оно должно быть округлено к самому близкому целому числу. Если это значение меньше чем 1 или больше чем 128, то это ошибка времени выполнения.

2. Второе выражение должно соответствовать требуемой для синтеза скорости *MIDI*. Если это значение не будет целым числом, то оно должно быть округлено к самому близкому целому числу. Если это значение меньше чем 0 или больше чем 128, то это ошибка времени выполнения.

3. Третье выражение, если дано, соответствует предварительно установленному числу *MIDI*.

4. Четвертое выражение, если дано, соответствует числу банка *MIDI*. Если есть меньше чем четыре выражения, и никакое инструментальное предварительно установленное число не обеспечивается, то это синтаксическая ошибка.

Выражение *sasbf* является выражением уровня. У выражения *sasbf* есть два канала, определенные процедурой синтеза.

Значение *sasbf* выражения вычисляется следующим образом:

На первом выполнении выражения должно быть оценено каждое выражение в списке выражения. Используя эти значения, одно примечание синтеза должно быть диспетчеризировано к процедуре синтеза банка звуковой таблицы.

На первом и каждом последующем уровне проходят через выражение *sasbf*. Значение выражения должно быть следующей парой аудиосэмплов от процесса синтеза банка звуковой таблицы стерео.

Во время процесса синтеза банка звуковой таблицы значения *MIDI*-контроллеров и другие непрерывные значения в текущем канале и примечании должны быть предпочтительны. Эти значения не передают в список выражения *sasbf*, но делаются доступными для синтезатора *sasbf* в зависимости от способа реализации. Если значения глобального *MIDIctrl[]*, стандартное имя будет изменено каким-либо инструментом, то новые значения должны быть предпочтительны для всех процессов синтеза *sasbf*.

Если инструмент, содержащий определенное выражение *sasbf*, не инстанцировали в ответ на событие *MIDI*, то его нет ни в одном канале, и таким образом, синтезом *sasbf* для этого выражения не могут управлять основанные на *MIDI* непрерывные контроллеры.

Синтаксис выражения *sasbf* приводится к одному примечанию процедуры синтеза *sasbf*. Не существует механизма чередования выборки единичного выражения *sasbf* в многократных строках для того, чтобы инстанцировать многократные процедуры синтеза банка с одним синтаксическим выражением. Выражение *sasbf* не является кодом операции, и не разрешается использовать его в качестве *oparray* конструкции.

Значение стандартного имени *released* в инструменте, содержащем выражение *sasbf*, должно быть сделано доступным для каждого процесса *sasbf* в зависимости от способа реализации. Процесс *sasbf* должен использовать этот флаг, так чтобы определить, когда начать синтез части выпуска данного указания. Если определенный экземпляр *sasbf* должен расширить продолжительность времени выпуска, то это должно расширить примечание на один *k* цикл. Если на следующем *k* цикле экземпляр *sasbf* все еще не заканчивается, то это может расширить примечание на дальнейший *k* цикл, и так далее.

Если в инструменте кратное число *sasbf* и каждый требует расширенной продолжительности, то вместе они должны расширить продолжительность на один *k* цикл.

Процесс синтеза *sasbf* для каждого примечания завершается, когда инструмент, содержащий выражение *sasbf*, уничтожается.

8.6.8 Стандартные имена

8.6.8.1 Определение

Не все идентификаторы, которые будут созданы в инструменте или коде операции, обязаны быть объявлены как переменные. Несколько идентификаторов, названных стандартными именами, не должны использоваться в качестве переменных и фиксировать семантику, которая должна быть реализована в совместимом декодере *SAOL*. Стандартные имена могут использоваться в качестве переменных, встроенных в выражения любым инструментом *SAOL* или кодом операции. Семантика использования стандартного имени *lvalue* не определена.

8.6.8.2 *k_rate*

lvar k_rate

Стандартное имя *k_rate* должно содержать уровень управления оркестра, Гц.

8.6.8.3 *s_rate*

lvar s_rate

Стандартное имя *s_rate* должно содержать частоту дискретизации оркестра, Гц.

8.6.8.4 *inchan*

lvar inchan

Стандартное имя *inchan* в каждом контексте должно содержать число каналов ввода, обеспечивающих инструментальное инстанцирование, с которым связывается этот контекст. Для кода операции инструментальное инстанцирование указывается кодом операции.

У различных экземпляров того же самого инструмента могут быть различные числа каналов ввода. Инструкции для того, чтобы вычислить значение стандартного имени, в 7.3.3.5.2.

8.6.8.5 *outchan*

lvar outchan

Стандартное имя *outchan* в каждом контексте должно содержать число каналов ввода, обеспечивающих инструментальное инстанцирование, с которым этот контекст имеет смысловую связку.

8.6.8.6 *time*

Ivar time

Стандартное имя *time* в каждом контексте должно содержать время, в которое создавалось инструментальное инстанцирование, связанное с этим контекстом.

8.6.8.7 *dur*

Ivar dur

Стандартное имя *dur* в каждом контексте должно содержать продолжительность инструментального инстанцирования, как первоначально создающееся, и как потенциально пересмотренное при помощи *extend* оператора, или – 1, если продолжительность не была известна при инстанцировании.

Хотя *dur* является *i-rate* переменной, оно может быть изменено во время продолжительности инструментального экземпляра через *extend* оператор или через изменение темпа. В этом случае значения выражений в *ivar* не изменяются.

8.6.8.8 *itime*

ksig itime

Стандартное имя *itime* в каждом контексте должно содержать прошедшее время инструментального экземпляра. Таким образом на первом *k* цикле *itime* должно быть 0, и после этого должно быть постепенно увеличено на 1/*KR*, где *KR* является частотой дискретизации оркестра в начале каждого *k* цикла передачи уровня.

8.6.8.9 *MIDIctrl*

ksig MIDIctrl [128]

Стандартная переменная *MIDIctrl* должна содержать для каждого контекста текущее количество *MIDI*-контроллеров на канал, соответствующее каналу, которому присваивается инструментальное инстанцирование, связанное с тем контекстом.

Инструменты могут использовать *MIDIctrl* в качестве *Ivalue*, то есть, чтобы присвоить новые значения, использующие = оператор.

MIDIctrl [64] является специальным контроллером. Нормативно определено, как выдержать контроллер, и для этого есть *MIDI noteoff* инструкции.

8.6.8.10 *MIDItouch*

ksig MIDItouch

Стандартная переменная *MIDItouch* должна содержать для каждого контекста текущую величину ретуши *MIDI* на ноте, которая заставляет связанное инструментальное инстанцирование создаваться.

8.6.8.11 *MIDIbend*

ksig MIDIbend

Стандартная переменная *MIDIbend* должна содержать для каждого контекста текущую величину *MIDI* изменения высоты тона на канале, соответствующем каналу, которому присваивается инструментальное инстанцирование, связанное с этим контекстом.

8.6.8.12 *channel*

Ivar channel

Стандартное имя *channel* содержит расширенный канал *MIDI* примечания, ответственного за создание текущего инструментального экземпляра.

8.6.8.13 *preset*

ivar preset

Стандартное имя *preset* содержит предварительно установленное число *SAOL* (который составляет меньше чем предварительно установленное число *MIDI*) примечания, ответственного за создание текущего инструментального экземпляра. Стандартное имя *preset* не содержит все предварительно установленные числа для текущего инструмента, а только те, которые привели к инстанцированию текущего экземпляра.

8.6.8.14 *input*

asig input [inchannels]

Стандартная переменная *input* должна содержать для каждого контекста входной сигнал или сигналы, обеспечиваемые для инструментального инстанцирования через *send* инструкцию.

8.6.8.15 *inGroup*

ivar inGroup [inchannels]

Стандартная переменная *inGroup* должна содержать для каждого контекста группировки входных сигналов, обеспечиваемых для инструментального инстанцирования.

8.6.8.16 *released*

ksig released

Стандартное имя *released* должно содержать для каждого контекста 1, если инструментальное инстанцирование связанное с контекстом будет уничтожено в конце текущей передачи оркестра. Иначе, *released* буду содержать 0.

8.6.8.17 *cpuload*

ksig cpuload

Стандартное имя *cpuload* должно содержать для каждого контекста способы загрузки памяти центрального процессора, наиболее строго связанные с инструментальным инстанцированием, связанным с контекстом. Если инструментальное инстанцирование будет работать полностью на одном центральном процессоре, то этот центральный процессор должен быть оценен. Если инструментальное инстанцирование работает на нескольких процессорах, то оценочная процедура ненормирована.

Величина загрузки ЦП должна быть, как процент возможности в реальном времени. Если ЦП будет полностью загружен и не сможет больше выполнять вычисления, то значение *cpuload* должно быть 1 на этом ЦП, в этом *k* цикле. Если ЦП будет полностью разгружен и не выполнит вычислений, то значение *cpuload* должно быть 0 на этом ЦП в том *k* цикле. Если ЦП будет полузагружен и сможет выполнить вдвое больше вычислений в реальном времени, чем выполняет в настоящий момент, то значение *cpuload* должно быть 0,5 на этом ЦП, в том *k* цикле.

8.6.8.18 *position*

imports ksig position [3]

Стандартное имя *position* содержит абсолютную позицию узла, ответственного за создание текущего оркестра. Позиция дается текущей величиной поля *position sound* узла, который является первым узлом в графике. Значение является глобальной переменной и совместно используемой всеми инструментами.

8.6.8.19 *direction*

ksig direction [3]

Стандартное имя *direction* содержит ориентацию узла, ответственного за создание текущего оркестра. Направление дается текущей величиной поля *direction sound* узла, который является первым узлом в графике. Значение является глобальной переменной и совместно используемой всеми инструментами.

8.6.8.20 *listenerPosition*

ksig listenerPosition [3]

Стандартное имя *listenerPosition* содержит абсолютную позицию слушателя. Позиция дается текущей величиной поля *position* активного узла *ListeningPoint*.

8.6.8.21 *listenerDirection*

ksig listenerDirection [3]

Стандартное имя *listenerDirection* содержит ориентацию слушателя. Направление дается текущей величиной поля *direction* активного узла *ListeningPoint*.

8.6.8.22 *minFront*

ksig minFront

Стандартное имя *minFront* дает один параметр звуковой диаграммы излучения звука текущего узла. Этот параметр и его семантика определяются *minFront* полем первого *sound* узла.

8.6.8.23 *maxFront*

ksig maxFront

Стандартное имя *maxFront* дает один параметр звуковой диаграммы излучения звука текущего узла. Этот параметр и его семантика определяются *maxFront* полем первого *sound* узла.

8.6.8.24 *minBack*

ksig minBack

Стандартное имя *minBack* дает один параметр звуковой диаграммы излучения звука текущего узла. Этот параметр и его семантика определяются *minBack* полем первого *sound* узла.

8.6.8.25 *maxBack*

ksig maxBack

Стандартное имя *maxBack* дает один параметр звуковой диаграммы излучения звука текущего узла. Этот параметр, и его семантика, определяются *maxBack* полем первого *sound* узла.

8.6.8.26 *params*

Imports exports ksig params [128]

Стандартное имя *params* используется совместно со всеми инструментами. В каждом *k* цикле оркестра должна содержаться текущая величина поля *params* узла *AudioFX*, ответственного за инстанци-

рование текущего оркестра. Если оркестр будет создаваться узлом *AudioSource*, а не узлом *AudioFX*, то значение *params* должно быть 0 в каждом канале.

Инструменты могут использовать *params* в качестве *lvalue*, то есть, чтобы присвоить новые значения используют = оператор. В этом случае, когда инструменту присваивается стандартное имя *params*, значение контроллера на канале, которому присваивается инструментальный экземпляр, связанный с тем контекстом, должно быть изменено. Значение *params* изменяется во всех других инструментальных экземплярах, связанных с этим каналом, и это изменение должно вступить в силу тогда, когда каждый из этих инструментальных экземпляров выполняется в *k* уровне.

8.7 Определение кода операции

8.7.1 Синтаксическая форма

Авторы потока битов могут создать свои собственные коды операции согласно этим правилам, чтобы инкапсулировать функциональность и упростить инструменты и процесс авторской разработки контента.

<opcode definition> -> *<opcode rate> <ident> (<formal param list>) {*
<opcode var declarations>
<opcode statement block>
}

<opcode rate> -> *aopcode*
<opcode rate> -> *kopcode*
<opcode rate> -> *iopcode*
<opcode rate> -> *opcode*

У определения кода операции есть несколько элементов:

1. Тег уровня, который определяет уровень, на котором выполняется код операции, или указывает, что код операции является полиморфным уровнем,
2. Идентификатор, который определяет имя кода операции,
3. Список нулевых или более формальных параметров кода операции,
4. Список нуля или большего количества объявлений переменной кода операции,
5. Блок операторов, определяющих исполнимую функциональность кода операции.

8.7.2 Тег уровня

Тег уровня описывает уровень, на котором код операции должен работать, или указывает, что код операции является полиморфным уровнем. Существуют четыре тега уровня:

1. *iopcode*, указывая, что код операции работает в *i-rate*,
2. *kopcode*, указывая, что код операции работает в *k -rate*,
3. *aopcode*, указывая, что код операции работает в *i-rate*,
4. *opcode*, указывая, что код операции является полиморфным уровнем.

8.7.3 Имя кода операции

Любой идентификатор может служить именем кода операции за исключением того, что имя кода операции не должно быть зарезервированным словом, именем одного из базовых кодов операции, перечисленных в разделе 9, или именем одного из базовых генераторов звуковой таблицы, перечисленных в разделе 10. Имя кода операции может быть именем переменной в локальном или глобальном счете.

Ни у каких двух инструментов или кодов операции в оркестре не должно быть того же самого имени.

8.7.4 Формальный список параметра

8.7.4.1 Синтаксическая форма

<formal param list> -> *<formal param> [, <formal param list>]*
<formal param list> -> *<NULL>*
<formal param> -> *<opcode variable rate> <name>*
<formal param> -> *table <ident>*
<opcode variable rate> -> *asig <opcode variable rate>* -> *ksig <opcode variable rate>* -> *ivar <opcode variable rate>* -> *xsig*

Формальный список параметра определяет интерфейс вызова к коду операции. Каждый формальный параметр в списке имеет имя, тип уровня, и может иметь ширину массива. Если ширина массива будет специальным маркером *inchannels*, то ширина массива должна быть тем же самым числом каналов ввода к связанному инструментальному инстанцированию. Если ширина массива будет специальным маркером *outchannels*, то ширина массива должна быть тем же самым числом каналов вывода к связанному инструментальному инстанцированию.

В пределах блока оператора кода операции формальные параметры могут использоваться, как любая другая переменная. Тег уровня каждого формального параметра определяет уровень переменной. Если код операции будет на определенном уровне, то никакой формальный параметр не должен быть объявлен раньше, чем этот уровень.

Существует специальный тег *xsig*, который позволяет формальным параметрам быть полиморфным уровнем. *xsig* не должен быть тегом уровня любого формального параметра, если тип кода операции не будет *opcode*.

8.7.5 Объявления переменной кода операции

8.7.5.1 Синтаксическая форма

<opcode var declarations> -> <opcode var declaration> [<opcode var declarations>]

<opcode var declarations> -> <NULL>

<opcode var declaration> -> <instr variable declaration>

<opcode var declaration> -> xsig <namelist> ;

В контексте кода операции маркеры *inchannels* и *outchannels* обращаются к каналам ввода и каналам вывода связанного инструмента.

Синтаксис и семантика объявлений переменной кода операции являются теми же, как и в инструментальных объявлениях переменной, со следующими исключениями и дополнениями.

Имена переменной кода операции доступны только в рамках кода операции, содержащего их. Инструментальные объявления переменной для инструментального инстанцирования, связанного с вызовом кода операции, не в рамках кода операции, и ссылки на эти имена не должны быть сделаны, если имена не будут явно объявлены в пределах кода операции. Стандартные имена в рамках каждого кода операции должны дать семантику в применении к инструментальному инстанцированию, связанному с вызовом кода операции. У переменной не должно быть уровня выше, чем уровень кода операции. Стандартные имена выше, чем уровень кода операции, не определяются в коде операции. Тег *imports* не должен использоваться для локального сигнала *k* уровня, когда нет никакой глобальной переменной того же самого имени.

Значения переменных кода операции статичны и сохраняются от вызова до вызова, ссылающегося на определенное состояние кода операции. Значения переменных кода операции должны быть установлены в 0 в состояние кода операции прежде, чем первый вызов, ссылающийся на это состояние, будет выполнен.

tablemap объявление может сослаться на любые таблицы, объявленные в локальном контексте, так же как на любые формальные параметры, которые являются таблицами.

Значения переменных кода операции в различных состояниях того же самого кода операции (из-за различного синтаксического использования выражений кода операции или различных выражений индексации в *oparray* выражениях) являются отдельными и не имеют никакого отношения друг к другу.

Существует специальный тег *xsig*, который может использоваться, чтобы объявить переменные кода операции с полиморфным уровнем кода операции, если тип кода операции не будет *opcode*.

8.7.6 Блок оператора кода операции

8.7.6.1 Синтаксическая форма

<opcode statement block> -> <opcode statement> [<opcode statement block>]

<opcode statement block> -> <NULL>

<opcode statement> -> <statement>

<opcode statement> -> return (<expr list>) ;

Синтаксис и семантика операторов в кодах операции являются тем же, как синтаксис и семантика операторов в инструментах, со следующими исключениями и дополнениями.

Никакой оператор в коде операции не должен быть выше, чем уровень кода операции.

Оператор присваивания и значения всех переменных обращается к состоянию кода операции, или к связанному с определенным выражением индексации в обращении оператору *oparray*.

Существует специальный оператор *return*, который используется в кодах операции. Этот оператор позволяет возвращать значения кодов операции назад.

8.7.6.2 Оператор *return*

Оператор *return* позволяет возвращать значения кодов операции назад.

Список параметра выражения может содержать и однозначные и оцененные массивом выражения.

Уровень оператора *return* является уровнем кода операции. Никакое выражение в списке параметра выражения не должно быть выше, чем уровень кода операции.

Оператор *return* должен быть оценен следующим образом. Каждое выражение в списке параметра выражения оценивается в порядке, как они происходят в списке. Исходное значение кода операции явля-

ется значением массива, сформированным, упорядочивая значения параметров выражения. В случае, когда есть только один параметр выражения, который является однозначным выражением, тогда исходное значение кода операции является единственным значением этого выражения. У обратного значения, обозначенного оператором *return* в пределах кода операции, должна быть та же самая ширина.

После того, как встречаются с оператором *return*, никакие дальнейшие операторы в коде операции не оцениваются и сразу управляют возвратами к инструменту вызова или коду операции.

8.7.7 Уровень кода операции

8.7.7.1 Введение

Уровень вызова кода операции зависит от типа кода операции, следующим образом:

1. Если тип кода операции является *aopcode*, говорят о коде операции *a-rate*.
2. Если тип кода операции является *kopcode*, говорят о коде операции, *k-rate*.
3. Если тип кода операции является *iopcode*, говорят о коде операции *i-rate*.
4. Если тип кода операции является *opcode*, код операции является полиморфным уровнем.

8.7.7.2 Коды операции полиморфного уровня

Коды операции полиморфного уровня берут из контекста, в котором их создают. Это позволяет тому же самому блоку оператора кода операции применяться к многократным контекстам. Без такой конструкции три версии каждого кода операции этого вида должны быть созданы и использоваться, в зависимости от контекста.

Уровень кода операции *opcode* для определенного вызова является уровнем самого первого фактического выражения параметра в этом обращении или уровнем самого первого формального параметра в определении кода операции, или уровнем самой первой защиты *if, while,* или *else* выражения, окружающей вызов кода операции, или уровенем кода операции, включающего вызов кода операции. Если код операции *opcode* не имеет никаких *non-table* параметров, и не включается в защищенный блок или вызов кода операции, то это *kopcode* по умолчанию.

Коды операции полиморфного уровня могут содержать переменные объявления, и формальные объявления параметра, используя тег *xsig*. У формального параметра типа *xsig* тот же самый уровень, как фактическое выражение параметра в выражении вызова, которому он соответствует. У переменной типа *xsig* тот же самый уровень, как код операции.

Коды операции полиморфного уровня не должны содержать переменные объявления и операторы выше, чем самый первый формальный параметр в объявлении кода операции. Код операции со всеми *xsig* формальными параметрами не должен содержать переменные объявления кроме *xsig* и *ivar*, и это не должно содержать операторы на определенном уровне раньше, чем уровень инициализации.

8.7.7.3 Совместно используемые переменные и операторы ниже, чем уровень кода операции

В *kopcode* операторы на уровне инициализации выполняются при первом обращении в код операции. В этом обращении *imports*, *exports, imports exports ivars* и таблицы обновляются, как описано в 8.6.5.3 и 8.6.5.4, и любые поколения звуковой таблицы выполняются, как описано в 8.6.5.2.

В *aopcode* операторы на уровне инициализации выполняются при первом обращении в код операции. В этом обращении *imports*, *exports, imports exports ivars* и таблицы обновляются, как описано в 8.6.5.3 и 8.6.5.4, и любые поколения звуковой таблицы выполняются, как описано в 8.6.5.2. Операторы на уровне управления выполняются при первом обращении каждого *k* цикла к коду операции. В этом обращении *imports*, *exports*, *imports exports ksigs* и таблицы обновляются, как описано в 8.6.5.3 и 8.6.5.4. Операторы, которые содержат выражения типа *specialop*, выполняются в *k* уровне.

8.8 Шаблонное объявление

8.8.1 Синтаксическая форма

<template declaration> -> template < <identlist> > [preset <maplist>] (<identlist>)
map { <identlist> } with { <maplist> }
{ <instr variable declarations> <block> }
<maplist> -> < <expr list> > , <maplist>
<maplist> -> < <expr list> >

Шаблонное объявление позволяет краткое объявление многократных инструментов, которые одинаковы в обработке структуры и синтаксисе, но отличаются только по нескольким ключевым выражениям или именам звуковой таблицы.

8.8.2 Семантика

Первый список идентификаторов содержит имена для инструментов, объявленных с шаблоном. В этом списке должен быть по крайней мере один идентификатор. Первый дополнительный *maplist* содержит

список предварительно установленных списков чисел, которые будут связаны с каждым инструментом в шаблоне. Этот *maplist* может быть опущен, когда нет никаких предварительно установленных чисел, связанных с шаблонными инструментами. Если *maplist* будет присутствовать, то он должен содержать столько списков сколько инструментальных имен в первом идентифицированном списке. Второй список идентификаторов содержит *pfields* для шаблонного объявления. У каждого инструмента объявленного с шаблоном, есть тот же самый список *pfields*. Третий список идентификаторов содержит список шаблонных переменных, которые должны быть заменены в последующем блоке кода с выражениями от второго *maplist*. В этом списке не может быть никаких идентификаторов, когда каждый инструмент, объявленный шаблоном, одинаковый.

Список карты принимает форму списка списков. У этого списка должно быть столько элементов, сколько шаблонных переменных объявлено в третьем списке идентификаторов. Каждый подсписок является списком выражений и должен иметь столько элементов, сколько инструментальных имен в первом списке идентификаторов. Первый (дополнительный) *maplist* не должен содержать идентификаторы, только числовые значения.

8.8.3 Шаблонные инструментальные определения

Многие инструменты определяются шаблонным определением, так как их имена есть в первом списке идентификаторов. Чтобы описать каждый из инструментов, идентификаторы, описанные в третьем списке, поочередно заменяются выражениями из списка карты.

Чтобы создать код для первого инструмента, данный блок кода обрабатывается, заменяя первую шаблонную переменную первым выражением из первого подсписка списка карты, вторую шаблонную переменную первым выражением из второго подсписка списка карты, третью шаблонную переменную первым выражением из третьего подсписка списка карты, и так далее. Чтобы создать код для второго инструмента, данный блок кода обрабатывается заменяя, первую шаблонную переменную вторым выражением из первого подсписка списка карты, вторую шаблонную переменную вторым выражением из второго подсписка списка карты, третью шаблонную переменную вторым выражением из третьего подсписка списка карты, и так далее.

Эта обработка блока кода происходит перед любой проверкой синтаксиса или проверкой уровня элементов инструментов. Шаблонные переменные не являются истинными сигнальными переменными и не должны быть объявлены в переменном блоке объявления.

8.9 Зарезервированные слова

Следующие слова резервируются и не должны использоваться в качестве идентификаторов в оркестре *SAOL*.

Aopcode, asig, else, exports, extend, global, if, inports, inchannels, inst,r interp, iopcode, ivar, kopcode, krate, ksig, map, oparray, opcode, outbus, outchannels, output, preset, return, route, sasbf, send, sequence, spatialize, srate, table, tablemap, template, turnoff, while, with, xsig.

Кроме того, имена переменной, запускающиеся *_sym_*, резервируются для специфичного использования (например, поток битов *detokenisation*), и не могут быть именем инструмента, сигнальной переменной, звуковой таблицей или пользователем кода операции в оркестре.

9 Определения кода операции ядра *SAOL* и семантика

9.1 Введение

Все базовые коды операции должны быть реализованы в каждом терминале, который может декодировать объекты типа 3 или 4.

Для каждого базового кода операции описывается следующее:

прототип показывающий уровень кода операции, параметры, которые требуются при вызове кода операции и уровни этих параметров;

нормативная семантика обратного значения. Семантика описывает, как вычислить исходное значение для каждого вызова кода операции;

нормативная семантика любых побочных эффектов базового кода операции.

9.2 Тип *specialop*

Тег уровня *specialop* не является фактическим лексическим элементом языка *SAOL* и не должен появляться в оркестре *SAOL*, но используется в качестве сокращения для базовых кодов операции с ними.

Базовые коды операции с уровнем вводят *specialop* и описывают функции, которые отображаются от одного или более сигналов *a-rate* в сигнале *k-rate*. Таким образом, у них есть один или более параметров, которые изменяются в *a-rate*, и они имеют нормативную семантику *a-rate*, но они только возвращают значения и/или имеют побочные эффекты в *a-rate*. При использовании этих кодов операции в выражениях обрабатываются, как *kopcode* коды, и используются для определения ошибок уровня.

Выражение, запрещенное правилами в 8.6.6 и 8.6.7 для *a-rate* и *k-rate*, не должно иметь *specialop*. Выражение *specialop* должно быть только в защищенном контексте, если выражение защиты также является *specialop* выражением. Выражение *specialop* не должно быть в выражении защиты или кодовом блоке для *while* оператора. Обращения к *specialop* кодам операции не возможны в *iopcodes, kopcodes* и *opcodes*. Если обращения к *specialop* кодам операции присутствуют в *aopcode* и *specialop* кодах то операции для *a-rate* и *k-rate* выполняются по правилам, определенным в 8.7.7.3.

9.3 Список базовых кодов операции

Несколько базовых кодов операции описываются в последующих подпунктах. Они делятся по категориям, но в этом нет никакого нормативного значения, а только ясность представления.

Математические функции	*int, frac, dbamp, ampdb, abs, sgn, exp, log, sqrt, sin, cos, atan, pow, log*10, *asin, acos, floor, ceil, min, max*
Конверторы	*gettune, settune, octpch, pchoct, cpspch, pchcps, cpsoct, octcps, midipch, pchmidi, midioct, octmidi, midicps, cpsmidi*
Табличные операции	*ftlen, ftloop, ftloopend, ftsr, ftbasecps, ftsetloop, ftsetend, ftsetbase, ftsetsr, tableread, tablewrite, oscil, loscil, doscil, koscil*
Сигнальные генераторы	*kline, aline, kexpon, aexpon, kphasor, aphasor, pluck, buzz, grain*
Звуковые генераторы	*irand, krand, arand, ilinrand, klinrand, alinrand, iexprand, kexprand, aexprand, kpoissonrand, apoissonrand, igaussrand, kgaussrand, agaussrand*
Фильтры	*port, hipass, lopass, bandpass, bandstop, biquad, allpass, comb, fir, iir, firt, iirt*
Спектральный анализ	*fft, ifft*
Регулировка усиления	*rms, gain, balance, compressor*
Демонстрационное преобразование	*decimate, upsamp, downsamp, samphold, sblock*
Задержки	*delay, delay*1, *fracdelay*
Эффекты	*reverb, chorus, flange, speedt, fx_speedc*
Изменение темпа	*gettempo, settempo*

Для каждого базового кода операции дается прототип кода операции. Это показывает уровень кода операции, число необходимых и дополнительных формальных параметров и уровень каждого из формальных параметров. Определенные параметры к определенным базовым кодам операции представляются в скобках, когда ' этот формальный параметр является дополнительным. Определенные коды операции используют в кавычках, что означает, что код операции может обработать произвольное число параметров. Кавычки тегируются с уровнем для таких кодов операции, которые являются уровнем всех параметров. Если нет нормативного языка для определенного кода операции, который определен иначе, то это синтаксическая ошибка, если применяется какой-либо из следующих операторов:

- в обращении кода операции меньше фактических параметров, чем необходимых формальных параметров;
- в обращении кода операции больше фактических параметров, чем необходимо, и дополнительные формальные параметры и определения кода операции не включают переменные, разделенные кавычками;
- фактическое выражение параметра имеет более быстрый уровень, чем соответствующий формальный параметр или переменный формальный параметр, если это — корреспонденция;
- фактическое выражение параметра не является однозначным или не является табличным, когда соответствующий формальный параметр определяет таблицу.

Имена, связанные с формальными параметрами в базовых прототипах кода операции, не имеют никакого нормативного значения, но используются для ясности при обращении к значениям, которые передают, как соответствующие фактические параметры, и описывают, как вычислить значение базового кода операции.

9.4 Математические функции

9.4.1 Введение

Каждый из кодов операции вычисляет математическую функцию.

9.4.2 *int*

opcode int (xsig x)

Код операции *int* вычисляет целочисленную часть своего параметра.

Исходное значение должно быть целочисленной частью *x*.

9.4.3 *frac*

opcode frac (xsig x)

Код операции *frac* вычисляет дробную часть своего параметра.

Исходное значение должно быть дробной частью *x*, то есть, *x* — интервал (*x*). Если *x* отрицателен, то *frac (x)* также отрицателен.

9.4.4 *dbamp*

opcode dbamp (xsig x)

Код операции *dbamp* вычисляет параметр, эквивалентный амплитуде, где амплитуда 1 соответствует уровню 90 дБ. Если *x* не строго положителен, то это ошибка расчета.

Исходное значение должно быть $90 + 20 \log_{10} x$.

9.4.5 *ampdb*

opcode ampdb (xsig x)

Код операции *ampdb* вычисляет оцененый параметр, эквивалентный амплитуде, где амплитуда 1 соответствует уровню 90 дБ.

Исходное значение должно быть $10^{(x - 90)/20}$.

9.4.6 *abs*

opcode abs (xsig x)

Код операции *abs* вычисляет абсолютное значение параметра. Исходное значение должно быть $-x$, если $x < 0$, или x, в других случаях.

9.4.7 *sgn*

opcode sgn (xsig x)

Код операции *sgn* вычисляет знак (функцию знака) параметра. Исходное значение должно быть –1, если $x < 0$, 0 если $x = 0$, или 1 если $x > 0$.

9.4.8 *exp*

opcode (xsig x)

Код операции *exp* вычисляет экспоненциальную функцию.

Исходное значение должно быть e^x.

9.4.9 *log*

opcode log (xsig x)

Код операции log вычисляет логарифм параметра.

Если *x* не положителен, то это ошибка расчета.

9.4.10 *sqrt*

opcode sqrt (xsig x)

Код операции *sqrt* вычисляет квадратный корень параметра. Исходное значение должно быть *sqrt* (*x*).

9.4.11 *sin*

opcode sin (xsig x)

Код операции sin вычисляет синус параметра, данного в радианах. Исходное значение должно быть sin *x*.

9.4.12 *cos*

Opcode cos (xsig x)

Код операции cos вычисляет косинус параметра, данного в радианах. Исходное значение должно быть cos *x*.

9.4.13 *atan*

opcode atan (xsig x)

Код операции *atan* вычисляет арктангенс параметра, данного в радианах. Исходное значение должно быть $\tan^{-1} x$, в диапазоне $[-\pi/2, \pi/2)$.

9.4.14 *pow*

opcode pow (xsig x, xsig y)

Код операции *pow* вычисляет мощность.

Исходное значение должно быть x^y.

9.4.15 log10

opcode log10 (xsig x)

Код операции log10 вычисляет логарифм параметра с основанием 10. Исходное значение должно быть $\log_{10} x$.

9.4.16 *asin*

opcode asin (xsig x)

Код операции *asin* вычисляет арксинус параметра, в радианах.

Исходное значение должно быть $\sin^{-1} x$, в диапазоне $[-\pi/2, \pi/2)$.

9.4.17 *acos*

opcode acos (xsig x)

Код операции acos вычисляет арккосинус параметра, в радианах.

Исходное значение должно быть $\cos^{-1} x$, в диапазоне $[0, \pi)$.

9.4.18 *ceil*

opcode ceil (xsig x)

Код операции *ceil* вычисляет нижнее значение параметра. Исходное значение должно быть самым маленьким целым числом *y* так, чтобы $x \leq y$.

9.4.19 *floor*

opcode floor (xsig x)

Код операции *floor* вычисляет верхнее значение параметра. Исходное значение должно быть самым большим целым числом *y* так, чтобы $y \leq x$.

9.4.20 *min*

opcode min (xsig x1 [xsig...])

Код операции min находит минимум из значений нескольких параметров. Исходное значение должно быть минимальным значением из значений параметра.

9.4.21 *max*

opcode max (xsig x1 [xsig...])

Код операции max находит максимум из значений нескольких параметров. Исходное значение должно быть максимальным значением из значений параметра.

9.5 Преобразователи подачи

9.5.1 Введение

Существуют четыре представления для подачи в *SAOL* оркестра. Эти четыре представления следующие:

- класс подачи, или *pch* представление. Подача представляется как целочисленная часть номера октавы, где 8 будет октава содержащая середину *C* (*C*4); плюс дробная часть, которая представляет класс подачи, где ,00 будет *C*, ,01 будет *C#*, ,02 будет *D* и т.д. Дробные части между шагами класса подачи округляются к самому близкому классу подачи;

- часть октавы или *oct* представление. Подача представляется, как целочисленная часть номера октавы, где 8 будет октава, содержащая середину *C* (*C*4). Плюс дробная часть, которая представляет часть октавы, где каждый шаг равный 1/12 октавы, представляет полутон;

- *MIDI* представление числа подачи. Подача представляется как целочисленное число полутонов, выше или ниже середины *C*, представленной как 60;

- частота или *cps* представление. Подача, представленная как некоторое число циклов в секунду.

Каждый из преобразователей подачи представляет преобразование, которое сделает его имя в новом представлении первым и оригинальным параметром во втором представление. Таким образом *cpsmidi* является преобразователем, который возвращает частоту, соответствующую определенной подаче *MIDI*.

9.5.2 *gettune*

opcode gettune ([xsig dummy])

Код операции *gettune* возвращает значение текущей глобальной настройки оркестра. Глобальная настройка должна быть установлена по умолчанию в 440, но может быть изменена, используя *settune* код операции.

Dummy параметр используется, чтобы определить уровень обращения кода операции.

9.5.3 *settune*

kopcode settune (ksig x)

Код операции *settune* устанавливает и возвращает текущее значение глобальной настройки оркестра.

Исходное значение должно быть *x*.

9.5.4 octpch

opcode octpch (xsig x)

Код операции *octpch* преобразовывает класс октавы в соответствии с масштабом шкалы.

9.5.5 *pchoct*

opcode pchoct (xsig x)

Код операции *pchoct* преобразовывает представление октавы представлению класса подачи.

Если *x* не строго положителен, то это ошибка вычисления.

9.5.6 *cpspch*

opcode cpspch (xsig x)

Код операции *cpspch* преобразовывает представление класса подачи в циклах в секунду в соответствии с масштабом шкалы и глобальными настройками.

Если *x* не строго положителен,то это ошибка.

9.5.7 *pchcps*

opcode pchcps (xsig x)

Код операции *pchcps* преобразовывает представление в циклах в секунду в представление класса подачи в соответствии с глобальной настройкой.

Если *x* не строго положителен, то это ошибка.

9.5.8 *cpsoct*

opcode cpsoct (xsig x)

Код операции *cpsoct* преобразовывает представление октавы в циклах в секунду в представление в соответствии с глобальной настройкой.

Если *x* не строго положителен, то это ошибка.

9.5.9 *octcps*

opcode octcps (xsig x)

Код операции *octcps* преобразовывает представление циклов в секунду в представление октавы в соответствии с глобальной настройкой.

Если *x* не строго положителен, то это ошибка.

9.5.10 *midipch*

opcode midipch (xsig x)

Код операции *midipch* преобразовывает представление класса подачи в *MIDI* представление.

Если *x* <= 3, то это ошибка.

9.5.11 *pchmidi*

opcode pchmidi (xsig x)

Код операции *pchmidi* преобразовывает *MIDI* представление в представление класса подачи.

Если *x* не строго положителен, то это ошибка.

9.5.12 *midioct*

opcode midioct (xsig x)

Код операции *midioct* преобразовывает представление октавы в *MIDI* представление.

Если *x* <= 3, то это ошибка.

9.5.13 *octmidi*

opcode octmidi (xsig x)

Код операции *octmidi* преобразовывает *MIDI* представление в представление октавы.

Если *x* не строго положителен, то это ошибка.

9.5.14 *midicps*

opcode midicps (xsig x)

Код операции *midicps* преобразовывает представление циклов в секунду в *MIDI* представление в соответствии с глобальной настройкой.

Если *x* не строго положителен, то это ошибка.

9.5.15 *cpsmidi*

opcode cpsmidi (xsig x)

Код операции *cpsmidi* преобразовывает *MIDI* представление в представление циклов в секунду в соответствии с глобальной настройкой.

Если *x* не строго положителен, то это ошибка.

9.6 Табличные операции

9.6.1 *ftlen*

opcode ftlen (table t)

Код операции *ftlen* обратный длине таблицы. Длина таблицы является значением, вычисленным на основании размера параметра в звуковой таблице генератора.

Исходное значение должно быть длиной таблицы, на которую ссылается *t*.

9.6.2 *ftloop*

opcode ftloop (table t)

Код операции *ftloop,* обратный стартовой точке цикла звуковой таблицы. Точка цикла устанавливается или в звуковом демонстрационном блоке данных в потоке битов или *ftsetloop* кодом операции, в других случаях это 0.

Исходное значение должно быть стартовой точкой цикла звуковой таблицы на которую ссылается *t*.

9.6.3 *ftloopend*

opcode ftloopend (table t)

Код операции *ftloopend,* обратный конечной точке цикла звуковой таблицы. Точка цикла устанавливается или в звуковом демонстрационном блоке данных в потоке битов или *ftsetend* базовым кодом операции, в других случаях это 0.

Исходное значение должно быть конечной точкой цикла звуковой таблицы на которую ссылается *t*.

9.6.4 *ftsr*

opcode ftsr (table t)

Код операции *ftsr,* обратный частоте дискретизации звуковой таблицы. Частота дискретизации устанавливается в звуковом демонстрационном блоке данных в потоке битов, в других случаях это 0.

Исходное значение должно быть частотой дискретизации звуковой таблицы, Гц, на которую ссылается *t*.

9.6.5 *ftbasecps*

opcode ftbasecps (table t)

Код операции *ftbasecps* обратный основной частоте звуковой таблицы, в циклах в секунду (Гц). Основная частота устанавливается или в звуковом демонстрационном блоке данных в потоке битов, или в базовой звуковой таблице *sample* генератора, или базовым кодом операции *ftsetbase,* в других случаях это 0.

Исходное значение должно быть основной частотой звуковой таблицы, в Гц, на которую ссылается *t*.

9.6.6 *ftsetloop*

kopcode ftsetloop (table t, ksig x)

Код операции *ftsetloop* устанавливает стартовую точку цикла звуковой таблицы в новое значение.

Если *x* <0, или если *x* больше, чем размер звуковой таблицы, на которую ссылается *t,* то это ошибка вычисления.

У этого кода операции есть побочные эффекты. Стартовая точка цикла звуковой таблицы *t* должна быть установлена в *x*.

Исходное значение должно быть *x*.

9.6.7 *ftsetend*

kopcode ftsetend (table t, ksig x)

Код операции *ftsetend* устанавливает конечную точку цикла звуковой таблицы в новое значение.

Если *x* <0, или если *x* больше, чем размер звуковой таблицы, на которую ссылается *t,* то это ошибка вычисления.

У этого кода операции есть побочные эффекты. Конечная точка цикла звуковой таблицы *t* должна быть установлена в *x*.

Исходное значение должно быть *x*.

9.6.8 *ftsetbase*

kopcode ftsetbase (table t, ksig x)

Код операции *ftsetbase* устанавливает основную частоту звуковой таблицы в новое значение.

Если *x* не строго положителен, то это ошибка вычисления.

У этого базового кода операции есть побочные эффекты. Основная частота звуковой таблицы *t* должна быть установлена в *x*, где *x* является значением в Гц.

Исходное значение должно быть *x*.

9.6.9 *ftsetsr*

kopcode ftsetsr (table t, ksig x)

Код операции *ftsetsr* устанавливает параметры частоты дискретизации звуковой таблицы к новому значению *x*.

Если *x* не строго положителен, то это ошибка вычисления.

У этого базового кода операции есть побочные эффекты. Частота дискретизации звуковой таблицы *t* должна быть установлена в *x*, где *x* является значением в Гц.

Исходное значение должно быть *x*.

9.6.10 *tableread*

kopcode tableread (table t, xsig index)

Код операции *tableread* возвращает единственное значение из звуковой таблицы.

Исходное значение должно быть значением звуковой таблицы *t* по числу *index*, где номер 0 является первой выборкой в звуковой таблице. Если *index* не будет целым числом, то исходное значение должно быть интерполировано от соседних точек звуковой таблицы.

9.6.11 *tablewrite*

opcode tablewrite (table t, xsig index, xsig val)

Код операции *tablewrite* устанавливает единственное значение в звуковой таблице.

Если *index* <0, или если *index* больше, чем размер звуковой таблицы, на которую ссылается *t*, то это ошибка вычисления.

У этого кода операции есть побочные эффекты. *index* должен быть округлен к самому близкому целому числу, и значение *index* числа в звуковой таблице *t* должно быть установлено в новое значение *val*, где номер 0 является первой выборкой в звуковой таблице.

Исходное значение должно быть *val*.

Если глобальные таблицы написаны в уровне одного инструмена, это не указывает, когда новые значения становятся доступными другим инструментам. В частности это не указывает, доступны ли изменения в том же самом цикле оркестра. Изменения должны быть доступными всем инструментам в следующем цикле оркестра, и разрешаются быть доступными инструментам упорядоченным позже в том же самом цикле оркестра.

9.6.12 *oscil*

aopcode oscil (table t, asig freq [ivar val])

Код операции *oscil* описывает несколько раз циклы вокруг звуковой таблицы *t* на уровне *freq* циклов в секунду. Когда код операции будет оценен, *loops* должны быть округлены к самому близкому целому числу. Если *loops* не будует оценены, то их значения должно быть –1.

Если *loops* не строго положительны и также не –1, то это ошибка вычисления.

Исходное значение вычисляется согласно следующей процедуре.

При первом обращении уровня *oscil* внутренняя фаза должна быть установлена в 0, и внутреннее число набора циклов к *loops*. При последующих обращениях внутренняя фаза должна быть постепенно увеличена на *freq/SR*, где *SR* является частотой дискретизации оркестра. Если после приращения внутренняя фаза не будет в интервале [0,1] и внутреннее количество цикла строго положительно, то фаза должна быть установлена в дробную часть ее значения (*p*: = *p* – пол (*p*)), и количество цикла постепенно уменьшено.

Если внутреннее количество цикла будет 0, то исходное значение должно быть 0. Иначе исходное значение должно быть значением номера *x* в звуковой таблице, где *x* = *p* * *l*, где *p* является текущей внутренней фазой и *l* является длиной таблицы *t*. Если *x* не будет целым числом, то значение должно быть интерполировано от соседних табличных значений.

9.6.13 *loscil*

aopcode loscil (table t, asig freq [ivar basefreq, ivar loopstart, ivar loopend])

Код операции *loscil* описывает циклы вокруг звуковой таблицы *t*. Цикличное выполнение продолжается, пока код операции является активным и выполняется по сценарию, который зависит от основной частоты *basefreq* и частоты дискретизации таблицы.

Если *basefreq* не будет известен, то он должен быть установлен в основную частоту таблицы *t* по умолчанию. Если у таблицы *t* есть основная частота 0 и *basefreq* не известен, то это ошибка вычисления. Если *basefreq* не строго положителен, то это ошибка времени вычисления. Параметр *basefreq* должен быть определен в Гц.

Если *loopstart* и *loopend* не известны, то они должны быть установлены в стартовую точку и конечную точку цикла таблицы *t* соответственно. Если *loopend* не известен и конечная точка цикла *t* равна 0, то *loopend* должен быть установлен до конца таблицы в (*l* – 1), где *l* — длина таблицы в демонстрационных точках. Если *loopstart* не меньше, чем *loopend* или отрицателен,то это ошибка вычисления.

Исходное значение вычисляется согласно следующей процедуре.

Если *l* длина таблицы, то *m* — *loopstart*/*l*, и *n* — *loopend*/*l*. При первом обращении к уровню *loscil*, внутренняя фаза должна быть установлена в 0. При последующих обращениях внутренняя фаза должна быть постепенно увеличена на *freq*TSR/(basefreq*SR)*, где *TSR* является частотой дискретизации таблицы и *SR* является частотой дискретизации оркестра. Если приращение заставило внутреннюю фазу выйти за интервал [*m*, *n*] или становиться меньше, чем 0, фаза должна быть установлена в *m*+*p*–*kn*, где *p* является внутренней фазой, и *k* является нижним значением (*p*/*n*). Однако, если фазовый указатель еще не передал значение *loopstart*, и если приращение заставило фазовый указатель становиться меньше, чем нуль, фазовый указатель берет нулевое значение.

Исходное значение должно быть значением номера *x* в звуковой таблице, где *x* = *p* * *l*, *p* — текущая внутренняя фаза и *l* — длина таблицы *t*. Если *x* не будет целым числом, то значение должно быть интерполировано от соседних табличных значений.

9.6.14 *doscil*

aopcode doscil (table t)

Код операции *doscil* воспроизводит выборку без подстройки частоты или цикла.

Исходное значение вычисляется согласно следующей процедуре.

При первом обращении к уровню *doscil* внутренняя фаза должна быть установлена в 0. На последующих обращениях внутренняя фаза должна быть постепенно увеличена на *TSR*/*SR*, где *TSR* — частота дискретизации таблицы *t* и *SR* является частотой дискретизации оркестра. Если после приращения внутренняя фаза больше, чем 1, то делается код операции.

Если код операции будет сделан, то исходное значение должно быть 0. Иначе исходное значение должно быть значением демонстрационного номера *x* в звуковой таблице, где *x* = *p***l*, где *p* — текущая внутренняя фаза, и *l* — длина таблицы *t*. Если *x* не будет целым числом, то значение должно быть интерполировано от соседних табличных значений.

9.6.15 *koscil*

kopcode koscil (table t, ksig frq [ivar loops])

Код операции *koscil* описывает несколько раз циклы вокруг звуковой таблицы *t* на уровне *freq* циклов в секунду, возвращая значения на уровне управления. Когда код операции будет оценен, *loops* должны быть округлены к самому близкому целому числу. Если *loops* не известен, то его значение должно быть установлено в –1.

Если *loops* не строго положительны и также не –1, то это ошибка вычисления.

Исходное значение вычисляется согласно следующей процедуре.

При первом обращении к уровню *koscil* внутренняя фаза должна быть установлена в 0, и внутреннее число набора циклов в *loops*. При последующих обращениях внутренняя фаза должна быть постепенно увеличена на *freq/KP*, где *KP* — уровень управления оркестра. Если после приращения фаза не будет в интервале [0,1] и внутреннее количество цикла положительно, то фаза должна быть установлена в дробную часть ее значения (*p*: = *p* – пол (*p*)), и количество цикла постепенно уменьшено.

Если внутреннее количество цикла будет 0, то исходное значение должно быть 0. Иначе исходное значение должно быть значением номера *x* в звуковой таблице, где *x* = *p* * *l*, *p* — текущее значение внутренней фазы и *l* — длина таблицы *t*. Если *x* не будет целым числом, то его значение должно быть интерполировано из соседних табличных значений.

9.7 Сигнальные генераторы

9.7.1 *kline*

*kopcode kline (ivar x*1*, ivar dur*1*, ivar x*2 *[ivar dur*2*, ivar x*3*, ...])*

Код операции *kline* производит процедуру сегментированный строкой или функцию "отката" со значениями, изменяющимися на *k* уровне. Эта функция занимает *dur*1 секунды, чтобы пойти от *x*1 до *x*2, *dur*2 секунды, чтобы пойти от *x*2 до *x*3, и так далее.

Если появляются какое-либо из следующих условий, то это ошибка вычисления:

- число параметров четное;
- любое значений *dur* отрицательно.

Исходное значение должно быть вычислено следующим образом.

При первом обращении к *kline* внутреннее время должно быть установлено в 0, текущая левая точка в *x*1, текущая правая точка в *x*2, и текущая продолжительность в *dur*1. При последующих обращениях внутреннее время должно быть постепенно увеличено на 1/*KP*, где *KP* является уровнем управления оркестра. Пока внутреннее время больше, чем текущая продолжительность, внутреннее время должно быть постепенно уменьшено, текущая продолжительность должна быть установлена в следующий параметр продолжительности, текущую левую точку в текущую правую точку, и текущая правая точка в следующее контрольное значение. Если нет никакого дополнительного параметра продолжительности, то включается генератор.

Если включается генератор, то исходное значение равно 0. Иначе, исходное значение — $l + (r - l)\ t/d$, где l — текущая левая точка, r — текущая правая точка, t — внутреннее время, d — текущая продолжительность.

9.7.2 *aline*

*aopcode aline (ivar x*1, *ivar dur*1, *ivar x*2 *[ivar dur*2, *ivar x*3, *...])*

Код операции *aline* производит процедуру сегментированный строкой или функцию "отката" со значениями, изменяющимися на *a* уровне. Эта функция занимает *dur*1 секунды, чтобы пройти от *x*1 до *x*2, *dur*2 секунды, чтобы пойти от *x*2 до *x*3, и так далее.

Если появляются какое-либо из следующих условий, то это ошибка вычисления:

- число параметров четное;
- любое значений *dur* отрицательно.

Исходное значение должно быть вычислено следующим образом.

При первом обращении к *aline* внутреннее время должно быть установлено в 0, текущая левая точка в *x*1, текущая правая точка в *x*2 и текущая продолжительность в *dur*1. При последующих обращениях внутреннее время должно быть постепенно увеличено на 1/*SR*, где *SR* — частота дискретизации оркестра. Пока внутреннее время больше, чем текущая продолжительность и существует другой параметр продолжительности, внутреннее время должно быть постепенно уменьшено до текущей продолжительности, текущая продолжительность должна быть установлена в следующий параметр продолжительности, текущая левая точка установлена в текущую правую точку, и текущей правой точке присвоено значение следующей контрольной точки (*x*-значение) (эти шаги повторения в случае необходимости, пока внутреннее время больше, чем текущая продолжительность). Если нет никакого дополнительного параметра продолжительности, то включается генератор.

Если генератор включен, то исходное значение равно 0. Иначе, исходное значение равно $l + (r - l)\ t/d$, где l — текущая левая точка, r — текущая правая точка, t — внутреннее время, d — текущая продолжительность.

9.7.3 *kexpon*

kopcode kexpon (ivar x1, ivar dur1, ivar x2 [ivar dur2, ivar x3, ...])

Код операции *kexpon* выполняет сегментированную функцию, сделанную из экспоненциальных кривых со значениями, изменяющимися в *k* уровне. Эта функция занимает *dur*1 секунды, чтобы пойти от *x*1 до *x*2, *dur*2 секунды, чтобы пойти от *x*2 до *x*3 и так далее.

Если появляются какое-либо из следующих условий, то это ошибка вычисления:

- число параметров четное;
- значений *dur* отрицательно;
- все значения *x* не имеют одинаковый знак;
- все *x* равны 0.

Исходное значение должно быть вычислено следующим образом:

При первом обращении к *kexpon* , внутреннее время должно быть установлено в 0, текущая левая точка в *x*1, текущая правая точка в *x*2 и текущая продолжительность в *dur*1. При последующих обращениях внутреннее время должно быть постепенно увеличено на 1/*KP*, где *KP* — уровень управления оркестром. Пока внутреннее время больше, чем текущая продолжительность и есть другой параметр продолжительности, внутреннее время должно быть постепенно уменьшено до текущей продолжительности, текущая продолжительность должна быть установлена в следующий параметр продолжительности, текущая левая точка в текущую правую точку и текущая правая точка в следующую контрольную. Если нет никакого дополнительного параметра продолжительности, включается генератор.

Если генератор включен, то исходное значение равно 0. Иначе, исходное значение равно $l\,(r/l)^{t/d}$, где l — текущая левая точка, r — текущая правая точка, t — внутреннее время и d — текущая продолжительность.

9.7.4 *aexpon*

aopcode aexpon (ivar x1, ivar dur1, ivar x2 [ivar dur2, ivar x3, …])

Код операции *aexpon* выполняет сегментированную функцию, сделанную из экспоненциальных кривых со значениями, изменяющимися в *a* уровне. Эта функция занимает *dur*1 секунды, чтобы пойти от *x*1 до *x*2, *dur*2 секунды, чтобы пойти от *x*2 до *x*3 и так далее.

Если появляются какое-либо из следующих условий, то это ошибка вычисления:

- число параметров четное;
- значений *dur* отрицательно;
- все значения *x* не имеют одинаковый знак;
- все *x* равны 0.

Исходное значение должно быть вычислено следующим образом:

При первом обращении к *aexpon* внутреннее время должно быть установлено в 0, текущая левая точка в *x*1, текущая правая точка в *x*2 и текущая продолжительность в *dur*1. При последующих обращениях внутреннее время должно быть постепенно увеличено на 1/*SR*, где *SR* — частота дискретизации оркестра, пока внутреннее время больше, чем текущая продолжительность и есть другой параметр продолжительности, внутреннее время должно быть постепенно уменьшено до текущей продолжительности, текущая продолжительность должна быть установлена в следующий параметр продолжительности, текущая левая точка в текущую правую точку и текущая правая точка в следующую контрольную точку. Если нет никакого дополнительного параметра продолжительности, то включается генератор.

Если генератор включен, то исходное значение равно 0. Иначе, исходное значение равно $l\,(r/l)^{t/d}$, где l — текущая левая точка, r — текущая правая точка, t — внутреннее время и d — текущая продолжительность.

9.7.5 *kphasor*

kopcode kphasor (ksig cps)

Код операции *kphasor* производит неоднократное цикличное передвижение фазового значения от 0 до 1, время *cps* в секундах.

Исходное значение должно быть вычислено следующим образом:

При первом обращении к *kphasor* внутренняя фаза должна быть установлена в 0. При последующих обращениях внутренняя фаза должна быть постепенно увеличена на *cps/KR*, где *KR* является уровнем управления оркестра. Если внутренняя фаза будет не в интервале [0,1], то внутренняя фаза должна быть установлена в дробную часть ее значения (*p* = *frac* (*p*)). Исходное значение является внутренней фазой.

9.7.6 *aphasor*

aopcode aphasor (asig симв./с)

Код операции *aphasor* производит неоднократное цикличное передвижение фазового значения от 0 до 1, время *cps* в секундах.

Исходное значение должно быть вычислено следующим образом:

При первом обращении к *aphasor* внутренняя фаза должна быть установлена в 0. На последующих вызовах внутренняя фаза должна быть постепенно увеличена на *cps/SR*, где *SR* — частота дискретизации оркестра. Если внутренняя фаза будет не в интервале [0,1], то внутренняя фаза должна быть установлена в дробную часть ее значения *(p = frac (p))*. Исходное значение является внутренней фазой.

9.7.7 *pluck*

aopcode pluck (asig cps/c, ivar buflen, table init, ksig atten, ksig smoothrate)

Код операции *pluck* использует простую форму *Karplus-Strong* алгоритма, чтобы генерировать звуки строки повторной выборкой и сглаживанием буфера.

Если *buflen* не строго положителен, то это ошибка вычисления.

Исходное значение вычисляется следующим образом:

При первом обращении к *pluck* кода операции, буфер длины *buflen* должен быть создан и заполнен значениями от таблицы *init*, следующим образом. Если *x* — длина таблицы *init* будет меньше, чем *buflen*, то значения буфера должны быть установлены в первые *buflen* значения таблицы *init*. Если *x* будет больше или равным *buflen*, то первые *buflen* значения буфера должны быть установлены в значения в таблице *init*, а остаток от буфера заполнен, как описано в этом абзаце для целой таблицы. Чтобы заполнить буфер многократно используются полнее и частичные циклы.

Также на первом обращении *pluck* внутренняя фаза должна быть установлена в 0, и *smooth count* должен быть установлен в 0.

При последующих обращениях к *pluck* относительно состояния *smooth count* постепенно увеличивается. Если *smooth count* равно *smoothrate*, то *smoothrate* устанавливается в 0, и буфер должен быть сглажен следующим образом. Создается новый буфер длины *buflen,* и ее значения устанавливаются, равными среднему по текущему буферу. Каждое значение в новом буфере должно быть установлено в значение минимальное среднее значение пяти выборок текущего буфера. Таким образом, для каждой выборки *x* нового буфера его значение должно быть установлено в *atten * (b [x-2] + b [x-1] + b [x] + + b [x+1] + b [x+2])/*5, где b[.]относится к значениям текущего буфера и индексы вычисляются по модулю *buflen*. Затем значения текущего буфера должны быть установлены в значения нового буфера.

Внутренняя фаза должна быть постепенно увеличена на *cps/SR*, где *SR* является частотой дискретизации оркестра, и если получающееся значение не находится в интервале [0,1], то внутренняя фаза должна быть установлена в дробную часть внутренней фазы (*p* = *p* – пол (*p*)).

Исходное значение должно быть значением буфера в точке *p * buflen*, где *p* — внутренняя фаза. Если этот индекс не будет целым числом, то значение должно быть интерполировано из соседних значений буфера.

9.7.8 buzz

aopcode buzz (asig cps, ksig nharm, ksig lowharm, ksig rolloff)

Код операции *buzz* производит ограничение полосы последовательности импульсов, сформированную добавлением в подтекст косинуса основной частоты *cps*, Гц.

lowharm дает самую низкую используемую гармонику, где 0 — низ для *cps* частоты. Если *lowharm* отрицателен, то это ошибка вычисления.

nharm дает число гармоник, используемых с *lowharm*. Если *nharm* не строго положителен, то каждый обертон для частоты Найквиста оркестра используется (*nharm* будет установлен в *SR / 2 / cps – lowharm*).

rolloff дает мультипликативный *rolloff*, который определяет спектральную форму. Если *rolloff* отрицателен, то *partials* чередуются синфазно. Если |*rolloff* |> 1, то *partials* увеличиваются в амплитуде вместо ослабления.

Исходное значение вычисляется следующим образом. При первом обращении *buzz* относительно определенного контекста, внутренняя фаза должна быть установлена в 0. На последующих обращениях внутренняя фаза должна быть постепенно увеличена *cps/SR*, где *SR* частота дискретизации оркестра. Если после этого приращения внутренняя фаза будет больше чем 1, то внутренняя фаза должна быть установлена в дробную часть ее значения (*p: = frac (p)*).

Исходное значение должно быть

$$scale * \sum_{f=lowharm}^{lowharm+\ nharm} rolloff^{\,f-lowharm} \cos 2\pi(f+1)p,$$

где *p* является внутренней фазой.

Если |*rolloff* = 1 |, *scale* = (1 / (*nharm*+1)), то *scale* = (1–*abs* (*rolloff*)) / (1–*abs* (*rolloff* $^{(nharm\ +\ 1)}$)).

9.7.9 *grain*

aopcode grain (table wave, table evn, ksig density, ksig freq, ksig amp, ksig dur, ksig time, ksig phase)

Код операции *grain* используется для синтеза мелких частиц *[GRAN]*, чтобы синтезировать периодические, квазипериодические, шумные и текстурированные звуки. Звук при этом синтезе представляется как сумма многих коротких звуковых выборок или "мелких частиц", распределенных в частотно-временном пространстве.

wave является формой волны для мелкой частицы. *env* является конвертом для мелкой частице. *density* является интервалом времени триггерных зон в Гц. *freq* является частотой, в Гц, в которую можно поместить каждую новую мелкую частицу. *amp* является амплитудой каждой новой мелкой частицы, как масштабный коэффициент. *dur* является продолжительностью каждой мелкой частицы в секундах. *time* является смещением в секундах от триггера, в котором запускаются мелкие частицы (дрожат). *phase* является запускающейся фазой каждой мелкой частицы в диапазоне [0,1].

Если какое-либо из следующих условий выполняется: *density* не положительна, *dur* отрицателен, *time* отрицательно или *phase* не находится в диапазоне [0,1], то это ошибка вычисления.

При первом обращении к *grain* кода операции *number of grains* определяется в 0 и *density clock* и *trigger clock* устанавливаются в 0.

На первом и каждом последующем обращении к *grain*, выполняются следующие шаги:

Время постепенно увеличивается на 1/*SR*, где *SR* — частота дискретизации оркестра. Если время больше или равно 1/*density*, то время обнуляется, и затем, если *time* <1/ *density*, триггерные часы устанавливаются в *time*. Если *time*> = 1/*density*, триггерное время остается 0 и никакая мелкая частица не создается.

Если триггерное время положительно, то оно постепенно уменьшается на 1/*SR*. Если триггерные часы меньше или равны нулю, новая мелкая частица диспетчеризируется. Чтобы диспетчеризировать новую мелкую частицу, число активных мелких частиц постепенно увеличивается и обозначается как *i*. Должно быть выделено место, чтобы разместить текущую фазу *phase [i],* частоту *freq [i],* амплитуду *amp [i],* продолжительность *dur [i],* и время *gtime [i]* новой мелкой частицы. Эти значения должны быть установлены в текущее значение *phase*, *freq*, *dur*, и 0 соответственно.

Для каждой активной мелкой частицы *i* текущее значение мелкой частицы *x [i]* вычисляют следующим образом. В зависимости от формата *wave* существуют три варианта звуковой таблицы:

1. Если *wave* имеет параметры частоты дискретизации и основной частоты, то она имеет определенную выборку и является переходом к *freq* в *loscil* (). *loopstart* будет установлен в начальные параметры из таблицы *wave. loopend* будет установлен в параметры конца таблицы или *l*-1, где *l* является табличной длиной *wave*, если параметр конца цикла 0. Пусть *m* будет значением *loopstart*/l и *n* будет значением *loopend*/l. Тогда каждый раз после первого вычисления значение фазы *phase [i]* будет увеличено на *freq [i]*TSR/ (basefreq*SR)*, где *TSR* является табличной частотой дискретизации и *basefreq* — основная частота из таблицы *wave*. Если после этого приращения фаза не находится в диапазоне [0,1], то фаза должна быть установлена в *phase* [*i*]-*floor* (*phase* [*i*]). Текущее значение мелкой частицы *x[i]* — значение номера *q* из звуковой таблицы, где *q* = *phase* [*i*] * (*m*-*n*) + *m*. Если значение *q* не будет целым числом, то значение *x* [*i*] будет интерполировано из соседних табличных значений.

2. Если *wave* имеет параметр частоты дискретизации, но не имеет параметр основной частоты, то она не имеет определенной выборки и является соответствующей частотой дискретизации оркестра и таким образом *doscil ()* и *phase [i]* игнорируются. Каждый раз после первого вычисляения значения фаза *phase* [*i*] будет увеличена на *TSR/SR*, где *TSR* является табличной частотой дискретизации, и *SR* является частотой дискретизации оркестра. Если после этого приращения фаза больше чем 1, то все будущие значения этой мелкой частицы равны 0. Иначе текущее значение мелкой частицы *x[i]* — это значение номера *q* из звуковой таблицы, где *q* = *phase [i]* * *l* и *l* являются длиной *wave* звуковой таблицы. Если значение *q* не будет целым числом, то значение *x [i]* будет интерполировано из соседних табличных значений.

3. Если *wave* не имеет ни параметров частоты дискретизации, ни основной частоты, то она будет *waveshaped* и колеблется в *oscil().*Каждый раз после первого вычисления значения фаза *phase* [*i*] будет увеличена на *freq [i]/SR*. Если после этого приращения фазовое значение будет вне диапазона [0,1], то фаза должна быть установлена в дробную часть ее значения *phase [i] — floor (phase [i]).* Текущее значение мелкой частицы *x [i]* будет значением номера *q* из звуковой таблицы, где *q* = *phase* [*i*] * *l* и *l* является длиной *wave* звуковой таблицы. Если значение *q* не будет целым числом, то значение *x* [*i*] будет интерполировано из соседних табличных значений.

Выходное значение мелкой частицы *x* [*i*] вычисленное одним из этих трех способов, модулируется звуковой таблицей согласно гранулярной продолжительности. Время *gtime* [*i*] постепенно увеличивается на 1/*SR*, где *SR* является частотой дискретизации оркестра. Если *gtime* [*i*]> *dur*, то мелкая частица закончена. Это действие должно быть отмечено, как неактивное, и его пространство может быть освобождено. Иначе, значение модулятора *m* [*i*] это значение номера *q* звуковой таблицы *env*, где *q* = *floor (gtime [i] / dur [i]* * *l)* и *l* является длиной звуковой таблицы *env*. Значение окончательного результата мелкой частицы *x* [*i*] повторно масштабируется значением модулятора в *x* [*i*] = *amp** *x* [*i*] * *m* [*i*].

Выходное значение из кода операции является суммой выходных значений для всех активных мелких частиц.

9.8 Шумовые генераторы

9.8.1 Примечание по шумовым генераторам и псевдослучайным последовательностям

Следующие коды операции генерируют шум, то есть псевдослучайные последовательности различных статистических параметров. Чтобы обеспечить максимальную декорреляцию среди многократных шумовых генераторов, важно, чтобы все ссылки на псевдослучайную генерацию совместно использовали единую обратную связь. Таким образом, все случайные значения, требуемые различными состояниями

различных шумовых генераторов, должны использовать последовательные значения от одной "основной" псевдослучайной последовательности.

Строго запрещается поддержать многократные псевдослучайные последовательности при использование того же самого алгоритма для различных состояний шумовых кодов операции генерации, потому что это может привести к сильным корреляциям между многократными шумовыми генераторами.

Применяется стандартное математическое описание функций плотности вероятности. Это означает это если *pdf* случайной переменной *x* является *f(x)*, тогда вероятность этого значения в диапазоне *[y, z]* является $\int_{y}^{z} f(x)\,dx$.

9.8.2 *irand*

iopcode irand (ivar p)

Код операции *irand* генерирует случайное число от линейного распределения. Исходное значение должно быть случайным числом *x*, выбранным согласно *pdf*

$$p\,(x) = \begin{cases} 1/2p, \text{ если } x \in [-p, p] \\ 0 \text{ в других случаях} \end{cases}$$

9.8.3 *krand*

kopcode krand (ksig p)

Код операции *krand* генерирует случайные числа линейного распределения. Исходное значение должно быть случайным числом *x*, выбранным согласно *pdf*

$$p\,(x) = \begin{cases} 1/2p, \text{ если } x \in [-p, p] \\ 0 \text{ в других случаях} \end{cases}$$

9.8.4 *arand*

aopcode arand (asig p)

Код операции *arand* генерирует случайные помехи согласно линейному распределению. Исходное значение должно быть случайным числом *x*, выбранным согласно *pdf*

$$p\,(x) = \begin{cases} 1/2p, \text{ если } x \in [-p, p] \\ 0 \text{ в других случаях} \end{cases}$$

9.8.5 *ilinrand*

iopcode ilinrand (ivar p1, ivar p2)

Код операции *ilinrand* генерирует случайное число от линейно уменьшающегося распределения. Исходное значение должно быть случайным числом *x* выбранный согласно *pdf*

$$p\,(x) = \begin{cases} \text{abs } (2 / (p2 - p1)^* [(x - p1) / (p2 - p1)]) \text{ если } x \in [-p1, p2] \\ 0 \text{ в других случаях} \end{cases}$$

9.8.6 *klinrand*

kopcode klinrand (ksig p1, ksig p2)

Код операции *klinrand* генерирует случайные числа от линейно уменьшающегося распределения. Исходное значение должно быть случайным числом *x*, выбранным согласно *pdf*

$$p\,(x) = \begin{cases} \text{abs } (2 / (p2 - p1)^* [(x - p1) / (p2 - p1)]) \text{ если } x \in [-p1, p2] \\ 0 \text{ в других случаях} \end{cases}$$

9.8.7 *alinrand*

aopcode alinrand (asig p1, asig p2)

Код операции *alinrand* генерирует случайные помехи от линейно уменьшающегося распределения. Исходное значение должно быть случайным числом x, выбранным согласно *pdf*

$$p\,(x) = \begin{cases} \text{abs } (2 / (p2 - p1)^* [(x - p1) / (p2 - p1)]) \text{ если } x \in [-p1, p2] \\ 0 \text{ в других случаях} \end{cases}$$

9.8.8 *iexprand*

iopcode iexprand (ivar p1)

Код операции *iexprand* генерирует случайное число из экспоненциального распределения со средним значением *p*1.

Исходное значение должно быть случайным числом *x,* выбранным согласно *pdf*

$$p(x) = \begin{cases} 0 \text{ если } x \le 0 \\ ke^{-kx}, \text{ где } k = 1/p1 \text{ в других случаях} \end{cases}$$

9.8.9 *kexprand*

kopcode kexprand (ksig p1)

Код операции *kexprand* генерирует случайные числа из экспоненциального распределения со средним значением *p*1.

Исходное значение должно быть случайным числом *x,* выбранным согласно *pdf*

$$p(x) = \begin{cases} 0 \text{ если } x \le 0 \\ ke^{-kx}, \text{ где } k = 1/p1 \text{ в других случаях} \end{cases}$$

9.8.10 *aexprand*

aopcode aexprand (asig p1)

Код операции *aexprand* генерирует случайные помехи согласно экспоненциальному распределению со средним значением *p*1.

Исходное значение должно быть случайным числом *x,* выбранным согласно *pdf*

$$p(x) = \begin{cases} 0 \text{ если } x \le 0 \\ ke^{-kx}, \text{ где } k = 1/p1 \text{ в других случаях} \end{cases}$$

9.8.11 *kpoissonrand*

kopcode kpoissonrand (ksig p1)

Код операции *kpoissonrand* генерирует последовательность чисел, как случайный двоичный файл так, чтобы среднее время между *1's* являлось 1*p* секундами. Если 1*p* не строго положителен, то это ошибка вычисления.

При первом обращении к *kpoissonrand*, случайное число *x,* должно быть выбрано согласно *pdf*

$$p(x) = \begin{cases} 0 \text{ если } x \le 0 \\ ke^{-kx}, \text{ где } k = 1/(p1 * KR) \text{ в других случаях} \end{cases},$$

где *KR* является уровнем управления оркестра.

Исходное значение должно быть равно 0, и уровень этого случайного значения должен быть сохранен.

При последующих обращениях сохраняемая сумма должна быть постепенно уменьшена на 1. Если сумма будет уменьшена на 1, то исходное значение должно быть 1, и новое случайное значение должно быть сгенерировано и сохранено, как описано выше. Иначе исходное значение должно быть 0.

9.8.12 *apoissonrand*

aopcode apoissonrand (asig p1)

Код операции *apoissonrand* генерирует шум, как случайный двоичный файл так, что среднее время между 1'*s* является *p*1 секундами. Если 1*p* не строго положителен, то это ошибка вычисления.

При первом обращении к *apoissonrand* кода операции, случайное число *x* должно быть выбрано согласно *pdf*

$$p(x) = \begin{cases} 0 \text{ если } x \le 0 \\ ke^{-kx}, \text{ где } k = 1/(p1 * SR) \text{ в других случаях} \end{cases},$$

где *SR* является частотой дискретизации оркестра.

Исходное значение должно быть равно 0, и уровень этого случайного значения должен быть сохранен.

При последующих обращениях сохраняемая сумма должна быть постепенно уменьшена на 1. Если сумма будет уменьшенна на 1, то исходное значение должно быть 1, и новое случайное значение должно быть сгенерировано и сохранено как описано выше. Иначе исходное значение должно быть 0.

9.8.13 *igaussrand*

iopcode igaussrand (ivar mean, ivar var)

Код операции *igaussrand* генерирует случайное число из Гауссового (нормального) распределения со средним *mean* и различными *var*.

Если *var* не строго положителен, то это ошибка вычисления

Исходное значение должно быть случайным числом *x*, выбранным согласно *pdf*

$$p\,(x) = \frac{e^{-(mean-x)^2(2var)}}{\sqrt{2\pi x\,var}},$$

то есть, *p (x)* пропорционально *N* (*mean*, *var*), где *mean* является средним значением и *var*- различными нормальными распределениями.

9.8.14 *kgaussrand*

kopcode kgaussrand (ksig среднее значение, ksig var)

Код операции *kgaussrand* генерирует случайные числа из Гауссового (нормального) распределения со средним *mean* и различными *var*.

Если *var* не строго положителен, то это ошибка вычисления.

Исходное значение должно быть случайным числом *x*, выбранным согласно *pdf*

$$p\,(x) = \frac{e^{-(mean-x)^2(2var)}}{\sqrt{2\pi x\,var}},$$

то есть, *p (x)* пропорционально *N* (*mean*, *var*), где *mean* является средним значением и *var* — различными нормальными распределениями.

9.8.15 *agaussrand*

aopcode agaussrand (asig среднее значение, asig var)

Код операции *agaussrand* генерирует случайные помехи из Гауссового (нормального) распределения со средним *mean* и различными *var*.

Если *var* не строго положителен, то это ошибка вычисления.

Исходное значение должно быть случайным числом *x*, выбранным согласно *pdf*

$$p\,(x) = \frac{e^{-(mean-x)^2(2var)}}{\sqrt{2\pi x\,var}},$$

то есть, *p (x)* пропорционально *N* (*mean*, *var*), где *mean* является средним значением и *var* — различными нормальными распределениями.

9.9 Фильтры

9.9.1 *port*

kopcode port (ksig ctrl, ksig htime)

Код операции *port* преобразовывает оцененный управляющий сигнал в портаментный сигнал. *ctrl* является входящим управляющим сигналом, и *htime* является полупериодом перехода в секундах, для перехода от одного значения до следующего.

Исходное значение вычисляется следующим образом. При первом обращении к *port* текущее значение и старое значение устанавливаются в *ctrl*. При последующих вызовах, если *ctrl* не равен новому значению, то старое значение устанавливается в текущее значение, новое значение устанавливается в *ctrl*, и текущее время устанавливается в 0.

Если *htime* равно 0, текущее значение устанавливается в новое значение.

Исходное значение вычисляется следующим образом. Если текущая значение и новое значение равны, то исходное значение является новым значением. Иначе текущее время постепенно увеличивается на 1/*KR*, где *KR* является уровнем управления оркестра. Затем текущее значение должно быть установле-

но в $o + (n - o)(1 - 2^{-t/\text{htime}})$, где *t* является текущим временем, *n* является новым значением, и *o* является старым значением.

9.9.2 *hipass*

aopcode hipass (asig ввод, ksig сокращение)

Код операции *hipass* входного сигнала фильтра высоких частот. *cut* является фильтром с пределом – 6 дБ.

Метод фильтрации высоких частот не нормирован. Любой фильтр с указанной характеристикой может использоваться.

Исходное значение должно быть результатом фильтрации *input* с фильтром высоких частот *cut*.

9.9.3 *lopass*

aopcode lopass (asig ввод, ksig сокращение)

Код операции *lopass* входного сигнала фильтра низких частот. *cut* является фильтром с пределом – 6 дБ.

Метод фильтрации низких частот не нормирован. Любой фильтр с указанной характеристикой может использоваться.

Исходное значение должно быть результатом фильтрации *input* с фильтром низких частот *cut*.

9.9.4 *handpass*

aopcode bandpass (asig input, ksig cf, ksig bw)

Код операции *bandpass* входного сигнала полосового фильтра. *cf* является центральной частотой полосы пропускания, Гц. *bw* является пропускной способностью фильтра, измеряющегося от -6 дБ ниже центральной частоты к точке –6 дБ выше, и определяется в Гц. Если *cf* и *bw* строго положительны, то это ошибка вычисления.

Метод полосовой фильтрации не нормирован. Любой фильтр с указанной характеристикой может использоваться.

Исходное значение должно быть результатом фильтрации *input* с полосовым фильтром с центральной частотой *cf* и пропускной способностью *bw*.

9.9.5 *bandstop*

aopcode bandstop (asig input, ksig cf, ksig bw)

Код операции *bandstop* определяет полосы фильтрации входного сигнала. *cf* является центральной частотой полосы фильтрации, Гц. *bw* является пропускной способностью фильтра, измеряющегося а пределах от –6 дБ ниже центральной частоты к точке на –6 дБ выше, в Гц. Если *cf* и *bw* строго положительны, то это ошибка вычисления.

Метод полосы фильтрации входного сигнала не нормирован. Любой фильтр с указанной характеристикой может использоваться.

Исходное значение должно быть результатом фильтрации *input* с заграждающим фильтром с центральной частотой *cf* и пропускной способностью *bw*.

9.9.6 *biquad*

aopcode biquad (asig input, ivar b0, ivar b1, ivar b2, ivar a1, ivar a2)

Код операции *biquad* выполняет фильтрацию, используя канонический фильтр второго порядка. Конструкция фильтров с точно нормируемыми характеристиками использует переходы кодов операции *biquad*.

Исходное значение вычисляется следующим образом. При первом обращении к *biquad* промежуточные переменные *ti, to, w*1, и *w*2 устанавливаются в 0. Затем при первом обращении и каждом последующем, следующий псевдокод определяет функциональность:

ti = input;

*к = w*2 + *b*0**ti;*

*w*2 = *w*1 – *a1***to* + *b*1**ti;*

*w*1 = –*a*2**to* + *b*2**ti;*

и Исходное значение *to*.

Код операции *biquad* не должен иметь распределения по времени, но должен взять его из числа обращений. Если на состояние *biquad* ссылаются дважды в одном и том же цикле, то эффективная частота дискретизации фильтра в два раза выше чем частота дискретизации оркестра.

9.9.7 *allpass*

aopcode allpass (asig input, ivar time, ivar gain)

Код операции *allpass* выполняет фильтрацию входного сигнала. Величина времени задержки обратной связи в секундах.

Исходное значение должно быть *output*.

9.9.8 *comb*

aopcode comb (asig input, ivar time, ivar gain)

Код операции *comb* выполняет гребенчатую фильтрацию входного сигнала. Величина времени задержки обратной связи в секундах.

Исходное значение должно быть *x*.

9.9.9 *fir*

aopcode fir (asig input, ksig b0 [ksig b1, ksig b2, ksig ...])

Код операции *fir* применяет фильтр *FIR* произвольного порядка к входному сигналу. Метод реализации фильтров FIR не нормирован.

9.9.10 *iir*

aopcode iir (asig input, ksig b0 [ksig a1, ksig b1, ksig a2, ksig b2, ksig ...])

Код операции *iir* применяет фильтр произвольного порядка *IIR* к входному сигналу. Метод реализации фильтров *IIR* не нормирован и оставляется открытый для разработчиков.

Параметры *b*0, *b*1, *b*2, ... и *a*1, *a*2, ... определяют фильтр *IIR*

$H(z) = (b0 + b1z^{-1} + b2z^{-2} + ...) / (1 + a1z^{-1} + a2z^{-2} + ...)$.

Исходное значение должно быть последовательными значениями сигнала из данного фильтра к сигналу *input* в последовательных обращениях *iir*.

9.9.11 *firt*

aopcode firt (asig input, table t [, ksig order])

Код операции *firt* применяет фильтр произвольного порядка *FIR*, данного в таблице, к входному сигналу. Метод реализации фильтров *FIR* не нормирован и оставляется открытым для разработчиков.

Значения, сохраненные в выборках 0, 1, 2, ... *order* таблицы *t*, определяют фильтр *FIR*

$H(z) = t[0] + t[1]\,z^{-1} + t[2]\,z^{-2} + \ldots\ t[order-1]\,z^{-order+1}$

Если *order* не будет дан или будет больше, чем размер звуковой таблицы *t*, то *order* будет установлен в размере звуковой таблицы. Если *order* является нулем или отрицательный, то это ошибка вычисления.

Обратные значения должны быть последовательными значениями сигнала фильтра к сигналу со значением *input* в последовательных обращениях *firt*.

9.9.12 *iirt*

aopcode iirt (asig input, table a, table b [ksig order])

Код операции *iirt* применяет фильтр произвольного порядка *IIR* с данными из двух таблиц к входному сигналу. Метод реализации фильтров *IIR* не нормирован и оставляется открытым для разработчиков.

Значения, сохраненные в выборках 1, 2, ... *order* таблицы *a* и выборках 0, 1, 2, ..., *order* звуковой таблицы *b*, определяют фильтр *IIR*

$H(z) = (b[0] + b[1]\,z^{-1} + b[2]\,z^{-2} + ...) / (1 + [1]\,z^{-1} + [2]\,z^{-2} + ...)$,

где массив используется, чтобы указать на выборки звуковой таблицы. Если *order* не будет дан или будет больше, чем размер из этих двух звуковых таблиц, то *order* будет установлен в больший размер из этих двух звуковых таблиц. Если одна звуковая таблица будет меньшей, чем *order*, то дополнительные значения должны быть приняты как нулевые коэффициенты. Если *order* является нулем или отрицательный, то это ошибка вычисления.

Обратные значения должны быть последовательными значениями сигнала фильтра к сигналу со значением *input* в последовательных звонках *iirt*.

9.10 Спектральный анализ

9.10.1 *fft*

specialop fft (asig input, table re, table im [ivar len, ivar len, ivar shift, table win])

Код операции *fft* вычисляет оконные и перекрытые фреймы *DFT* и помещает результат в две таблицы. Это — "специальный код операции". Он принимает значения на звуковом уровне, и возвращается только на уровне управления.

Существуют несколько дополнительных параметров. Параметр *len* определяет длину фрейма (чтобы использовать входные выборки). Если параметр *len* является нулем или не оценен, то *len* устанавливается в следующее большее значение *SR/KR*, где *SR* — частота дискретизации оркестра, *KR* — уровень управления оркестра. Параметр *shift* определяет число выборок, на которые можно сместить аналитическое окно. Если *shift* является нулем или не оценен, то *shift* устанавливается в *len*. Параметр *size* — длина *DFT*

вычисленного кодом операции. Если *size* является нулем или не оценен, то *size* устанавливается в *len*. Параметр *win* — аналитическое окно, применимое к анализу. Если *win* не оценен, то окно серии длинных импульсов использует длины *len*.

Вычисления кода операции *fft* следующие. При первом обращении к коду операции *fft* создается буфер, содержащий длины *len*, и первый (*len* — *shift*) буфер выборки инициализируется с нулями. На каждом *a* уровне при обращении к коду операции *input,* выборка вставляется в следующую неинициализированную буферную позицию. Когда все *len* выборки в буфере инициализируются, выполняются следующие шаги:

1. В новом буфере создаются длины *size*, для которого каждое значение *new*[*i*] устанавливается в значение *buf* [*i*]**win* [*i*], где *new* [*i*] — значение нового буфера, *buf* [*i*] — значение содержания буфера, и *win* [*i*] — значение выборки в звуковой таблице аналитического окна. Если *size*>*len*, то значения *new* [*i*] для *i*>*len* обнуляются. Если *size* <*len*, то используются только первые значения *size* буфера.

2. Первые выборки *shift* удаляются из буфера, и оставшиеся выборки *len- shift* перемещаются к передней стороне буфера. После этого сдвига выборки *shift* в конце буфера обнуляются. Если *shift* > *len*, буфер содержания очищается.

3. Вычисляется реальный *DFT* нового буфера, приведенного к длине *size* значения частотной области. Реальные компоненты *DFT* помещаются в первые *size* выборки *table re*; воображаемые компоненты *DFT* помещаются в первые *size* выборки *table im*. *DFT* располагается так, что самые низкие частоты, запускающиеся с *DC* в нулевой точке выходных таблиц, подходят к частоте Найквиста, как *size*/2. Отражение спектра от частоты Найквиста до частоты дискретизации помещается во вторую половину таблиц.

DFT определяется как

$$d[i] = \frac{\sum_{k=0}^{size-1} e^{ijk2\pi/size} new[k]}{\sqrt{size}},$$

где *d[i]* — получающиеся компоненты DFT, 0 <*i* <*size*;

new [k] входные выборки, 0 <*k* <*size*;

и *j* является квадратным корнем -1.

Исходное значение на определенном *k* цикле равно 1, если *DFT* был вычислен, начиная с последнего *k* цикла, или 0, если его не было.

9.10.2 *ifft*

aopcode ifft (table re, table im [ivar len, ivar shift, ivar size, table win])

Код операции *ifft* вычисляет оконный и перекрытый *IDFTs* и передает результат потоком как звук. *re* и *im* являются звуковыми таблицами, которые содержат реальные и воображаемые части *DFT* соответственно. Существуют несколько дополнительных параметров, которые управляют процедурой синтеза. Параметр *len* является числом выходных выборок, используемых в качестве звука. Если параметр *len* является нулем или не оценен, то *len* устанавливается в следующее большее значение *SR/KR*, где *SR* — частота дискретизации оркестра и *KR* — уровень управления оркестра. Параметр *shift* является числом выборок, на которые смещается аналитическое окно между фреймами. Если *shift* не дается или является нулем, он берется как *len*. Параметр *size* имеет размер *IDFT*. Если *size* не оценен или является нулем, он берется как *len*. Параметр *win* — окно синтеза. Если параметр *win* не оценен, то окно серии длинных импульсов использует длины *len*.

Вычисления кода операции *ifft* следующие. При первом обращении к коду операции *ifft* вычисляется *size* IDFT *table re, table im*. Если *re* и/или *im* будут более длинными чем выборки *size*, то должны использоваться только первые выборки *size* этих таблиц. Результатом этого *IDFT* является последовательность значений *size*. Реальные компоненты первых *len* элементов этой последовательности умножаются на соответствующие выборкам *win* окна и помещаются в выходной буфер длины *len*. (*out*[*i*] = *seq* [*i*] * *win* [*i*] для 0 <*i* <*len*).

IDFT вычисляется, учитывая, что элементы с самым низким номером табличного *re* и *im* являются самыми низкими частотами аудиосигнала, начинаясь с *DC* в выборке 0, продолжаясь до частоты Найквиста в объеме *size*/2, и затем в отраженном спектре от *size*/2 до *size* -1.

IDFT определяется как

$$seg[i] = \frac{\sum_{k=0}^{size-1} e^{-ijk2\pi/size} d[k]}{\sqrt{size}},$$

где *d* [*i*] — компоненты частоты *IDFT*, 0 <*i* <*size* (*d* [*i*] = *re* [*i*] + *j im* [*i*])

seq [*k*] являются выходными выборками, 0 <*k* <*size*;

Также на первом звонке *ifft* выходная точка синтеза устанавливается в 0.

При первом обращении к коду операции *ifft* выполняется следующее вычисление. Выходное значение кода операции присваивается значениям выходного буфера в выходной точке. Затем выходная точка постепенно увеличивается. Если выходная точка будет равна *shift*, то должны выполняться следующие шаги:

1. Первые выборки *shift* выходного буфера отбрасываются, остающиеся выборки *len- shift* выходного буфера смещаются в начало буфера, и последние выборки *shift* устанавливаются в 0.

2. *IDFT* текущее значение звуковых таблиц *re* и *im* вычисляется как описано выше. Первые *len* значения реальной части получающейся, из аудио последовательности умножаются на *win* окна синтеза, и результат добавляется к выходному буферу ([*i*] = [*i*] + *seq* [*i*] * *win* [*i*] * *shift* / *len* для 0 <*i* <*len*).

3. Выходная точка устанавливается в 0.

FFTs и *IFFTs* не должны быть реализованы одновременно. Могут использоваться другие методы (такие как табличные вычисления) для генерации спектра, которые будут превращены в звук при помощи *IFFT*, или могут быть созданы инструменты аудио распознавания образов, которые вычисляют функции *FFT* и возвращают или передают ее и результаты.

9.11 Регулировка усиления

9.11.1 *rms*

specialop rms (asig x [ivar length])

Код операции *rms* вычисляет мощность сигнала. Он принимает значения на звуковом уровне, и возвращает их только на *k* уровне.

Если *length* не будет оценена, то она должна быть установлена в продолжительность контролируемого периода. Если *length* оценена и не строго положительна, то это ошибка вычисления. Параметр *length* определяется в секундах.

Исходное значение вычисляется следующим образом. Пусть *l* будет нижним значением (*length* *SR*), где *SR* является частотой дискретизации оркестра. Буфер *b* [] длины *l,* сохраняет новые значения параметра *x*. Каждый период управления *RMS* этих значений вычисляется как

$$p = \sqrt{\frac{\sum_{i=0}^{l-1} b[i]^2}{l}}$$

и исходное значение является *p*.

9.11.2 *gain*

aopcode gain (asig x, ksig gain [ivar length])

Код операции *gain* ослабляет или усиливает амплитуду сигнала и делает его мощность, равной заданному уровню мощности.

Если *length* не оценена, то ее продолжительность должна быть установлена в контролируемый период. Если *length* оценена и не строго положительна, то это ошибка вычисления. Параметр *length* определяется в секундах.

Исходное значение вычисляется следующим образом. Пусть *l* будет нижним значением (*length* * *SR*), где *SR* является частотой дискретизации оркестра.

При первом обращении к коду операции уровень затухания устанавливается в 1. При каждом последующем обращении входное значение *x* должно быть сохранено в буфере *b* [] длиной *l*. Когда буфер полон, уровень затухания повторно вычисляется как

$$\frac{gain}{\sqrt{\frac{\sum_{i=0}^{l-1} b[i]^2}{l}}}$$

и буфер очищается.

Исходное значение при каждом обращении равно *x* * *A*, где *A* является текущим уровнем затухания.

9.11.3 *balance*

aopcode balance (asig x, asig ref [ivar length])

Код операции *balance* ослабляет или усиливает амплитуду сигнала и делает его мощность равной уровню мощности в опорном сигнале.

Если *length* не будет обеспечена, то она должна быть установлена в продолжительность периода управления. Это — ошибка времени выполнения, если *length* обеспечивается и не строго положительна. В секундах определяется параметр *length*.

Исходное значение вычисляется следующим образом. Пусть *l* будет значением *(length * SR)*, где *SR* частота дискретизации оркестра.

При первом обращении к коду операции уровень затухания устанавливается в 1. При каждом последующем обращении входное значение *x* должно быть сохранено в буфере *b* [] длиной *l*, и входное значение *ref* сохраненного в буфере *r* [] длиной *l*. Когда буферы переполнены, повторно вычисляется уровень затухания как

$$\frac{\sqrt{\frac{\sum_{i=0}^{l-1} r[i]^2}{l}}}{\sqrt{\frac{\sum_{i=0}^{l-1} b[i]^2}{l}}}$$

и буферы очищаются.

Исходное значение при каждом обращении равно *x* * *A*, где *A* является текущим уровнем затухания.

9.11.4 *compressor*

aopcode compressor (asig x, asig comp, ksig nfloor, ksig thresh, ksig loknee, ksig hiknee, ratio ksig, ksig att, ksig rel, ivar look)

Код операции *compressor* функционирует, как аудио компрессор ограничивающий, расширяющий или подавляющий шумы. Необходимы два входных сигнала аудиочастоты *x* и *comp*, первый из которых изменяется анализом работы второго. Оба сигнала могут быть одинаковы или первый может быть изменен различными сигналами управления.

Если какое-либо из следующих условий выполняется, то это ошибка вычисления: *thresh <nfloor, loknee < thresh, hiknee <loknee, ratio* равно, или меньше 0, *look* отрицателен, *att* отрицательно, или *rel* отрицателен.

compressor сначала исследует *comp* сигнала управления, выполняя детектирование огибающей. Это управляется двумя контрольными значениями *att* и *rel*, и значением инициализации *look,* определяющем начало, окончание и предварительные времена (в секундах) огибающей детектора.

look является предварительным временем (в секундах) огибающей детектора. Это определяет, как долго детектор ищет новый пик в затухающем сигнале. Если новый пик находится, то огибающая корректируется, чтобы интерполировать между текущими и будущими пиками.

att и *rel* являются временем начала и окончания огибающей детектора (в секундах). Для определения времени берут огибающую от точки пикового значения (для *att*) до точки нуля (для *rel*).

nfloor установит для системы абсолютный низ в децибелах.

Оценка огибающей преобразовывается в децибелы, затем проходит через функцию отображающую характеристики компрессора чтобы определить, какие меры должен предпринять компрессор. Отображающаяся функция определяется четырьмя областями: нулевая область, область 1:1 (никаких изменений), излом и область сжатия/расширения.

Расположения этих областей определяются контрольными значениями *thresh, loknee, hiknee,* и *ratio*.

thresh минимальный уровень в децибелах, который будет возможен. Для подавления шума это значение будет больше, чем *nfloor*.

loknee и *hiknee* являются значениями в децибелах, которые определяют, где начнется сжатие или расширение. Они устанавливают границы излома кривой, присоединяющейся к кривой без изменений и кривой сжатия/расширения с более высокой амплитудой.

ratio, данное в дБ, определяет сжатие выше излома. Это определяет изменение выходной мощности для изменения на 1 дБ. *Ratio,* оцененое на 1 дБ выше результата в сжатии, приводит к большему сжатию.

Параметры кода операции сжатия *nfloor, thresh, loknee,* и *hiknee* определяются для базового кода операции *dbamp* в децибелах.

При инициализации должно быть выделено место для двух буферов *xdly* и *compdly*. Длина в выборках этих буферов будет *SR * look*, где *SR* является частотой дискретизации оркестра. Начальные значения обоих буферов будут обнулены.

Место выделяется для следующих переменных:

gain — амплитудный множитель, который будет применен.

change — предполагаемое изменение огибающей от выборки до выборки.

*comp*1 — текущее значение *comp,* дБ.

*comp*2 — значение *comp* (от *compdelay*), дБ, инициализированное для того чтобы обнулить.

env — текущая оценка огибающей.

projEnv — спроектированное значение огибающей.

При каждом обрашении к *compressor* ( ), происходит следующее:

1. Выборка *x* помещается в начало буфера *xdly*. Все значения отодвигаются в конец и сохраняются как *oldval*.

2. Следующее значение огибающей вычисляется так:

abs(comp) конвертируют в децебелы. Это значение называется *comp*1

$comp1 = 90 + 20\log_{10}(abs(comp))$

Это значение помещается в конец буфера *compdly*. Значение *comp*2 используется в начале буфера *compdly*.

```
if (comp2 > env)
change = (comp2 – env) / (SR * att);
else {
projEnv = max(change * SR * look, nfloor)
if ((comp1 > projEnv) && (comp2 > comp1))
change = (comp1 – comp2) / (SR * rel);
else
change = 0;
}
env = max(env + change, nfloor);
```

3. Амплитудный множителя *gain* вычисляется следующим образом:

если (*env* < *thresh*),

gain сначала уменьшается равномерно от 1 до 0 и затем, оставаясь в 0, перемещается от *thresh* к *nfloor*, чтобы создать подавление шума. Точная кривая ввода — вывода в этом интервале не нормирована.

```
else {
if (env <= loknee)
gain = 1;
else if (env >= hiknee)
```

gain должно быть вычислено так, чтобы выше *hiknee* увеличение *ratio* во входной мощности на 1дБ привело к увеличению на 1дБ в выходной мощности. Кривая ввода/вывода в децибелах должна быть непрерывной в *hiknee* или с кривой излома.

```
else
```

gain должно быть гладко интерполировано между точками *loknee* и *hiknee* так чтобы создать кривую ввода-вывода с гладким изломом. Эта кривая должна монотонно увеличиваться и иметь непрерывную производную, равную 1 в *loknee* и 1/*ratio* в *hiknee*.

```
}
```

4. Выходное значение кода операции должно быть *oldval,* умноженным на *gain*.

9.12 Демонстрационное преобразование

9.12.1 *decimate*

specialop decimate (asig input)

Код операции *decimate* уменьшает сигнал в десять раз от уровня звука до контрольного уровня.

9.12.2 *upsamp*

asig upsamp (ksig input [, table win])

Код операции *upsamp* дискретизирует управляющий сигнал к аудиосигналу. Параметр *win* является дополнительным окном интерполяции. Если *win* не оценен, берется окно импульсов длины *SR/KR*, где *SR* является частотой дискретизации оркестра и *KR* является уровнем управления оркестра. Если *win* оценен и короче выборки *SR/KR*, то *SR/KR* в конце дополняется нулем (сама таблица не изменяется).

При первом обращении к *upsamp*, создается выходной буфер *win* и обнуляется. Кроме того, выходное значение устанавливается в 0.

При первом обращении к *upsamp* в *k* цикле выходной буфер смещается на *SR / KR*. Первая выборка *SR/KR* отбрасывается, остающиеся выборки смещаются к передней части выходного буфера, и последняя выборка *SR/KR* устанавливаются в 0. Затем оконная функция масштабируется *input* и добавляется в выходной буфер (*buf [i] = buf [i] + input * win [i]*, 0 <*i* < *length (win))*. Затем выходная точка устанавливается в 0.

При первом и каждом последующем обращении к *upsamp*, исходное значение является значением выходного буфера в текущей точке. Затем выходная точка должна быть постепенно увеличена.

9.12.3 *downsamp*

specialop downsamp (asig input [, table win])

Код операции *downsamp* дискретизирует звуковой сигнал к контрольному сигналу. Параметр *win* является дополнительным аналитическим окном.

Если *win* короче выборки *SR/KR*, где *SR* является частотой дискретизации оркестра, и *KR* является уровнем управления оркестра, то это ошибка вычисления.

Исходное значение вычисляется следующим образом. В каждом *k* цикле значение каждой выборки *input*, определенное в предыдущем цикле, помещают в буфер. Если *win* не оценен, то исходное значение является средним значением выборок в буфере. Если *win* оценено, то исходное значение вычисляется, умножая каждую точку аналитического окна на входной сигнал (*rtn* = Σ *input* [*i*] * *win* [*i*] для 0 <*i*<*SR/KR*, где *SR* является частотой дискретизации оркестра и *KR* является уровнем управления оркестра).

9.12.4 *samphold*

samphold opcode (xsig ввод, ksig gate)

Код операции *samphold* пропускает сигнал с контрольным сигналом.

Исходное значение вычисляется следующим образом. При первом обращении к *samphold* последнее переданное значение устанавливается в 0. Если значение *gate* не является нулевым, то последнее переданное значение устанавливается *input*.

9.12.5 *sblock*

specialop sblock (asig x, table t)

Код операции *sblock* создает контрольные уровни блоков выборок и размещает их в звуковую таблицу.

Исходное значение этого кода операции всегда 0.

9.13 Задержки

9.13.1 *delay*

aopcode delay (asig x, ivar t)

Код операции *delay* реализует фиксированную длину задержки от начала до конца. Параметр *t* дает задержку в секундах.

Пусть *y* будет меньшим из выбороки *(t*SR)*, где *SR* является частотой дискретизации оркестра. При каждом обращениик к коду операции *delay*, значение *x* вставляется в буфер *FIFO* длиной *y*. Исходное значение является значеним, которое было вставлено в строку задержки *y* в предыдущих обращениях.

9.13.2 *delay*1

aopcode delay1 (asig x)

Код операции *delay*1 реализует первую задержку.

При каждом обращении к *delay*1 сохраняется значение *x* и исходное значение является значением, сохраненным при предыдущем обращении.

9.13.3 *fracdelay*

aopcode fracdelay (ksig метод [xsig p1, xsig p2])

Код операции *fracdelay* реализует дробную переменную, и/или строки задержки. Существуют несколько методов для управления строкой задержки.

Семантика *p*1 и *p*2 и вычисление обратного значения отличаются в зависимости от значения *method*. Если *method* меньше, чем 1 или больше, чем 5, то это ошибка вычисления.

Если *method* - 1, то он нормирован как «initiallise». В этом случае *p*1 является длиной строки задержки в секундах. Если *p*1 не оценено или является меньше чем 0, то это ошибка вычисления. Любая существующая в настоящий момент строка задержки в состоянии кода операции должна быть уничтожена. Будет создана новая строка задержки с длиной *floor (p*1 * *SR),* где *SR* является частотой дискретизации оркестра и все значения на этой строке задержки должны быть инициализированы в 0. Исходное значение равно 0. Если *p*2 оценен, то он не используется и игнорируется.

Если *method* - 2, то он нормирован как «tap». В этом случае *p*1 является точкой касания в секундах. Если *method* 1 еще не вызвали для этого кода операции, или если *p*1 не оценен, или если *p*1 - меньше чем 0, или если *p*1 больше чем новая длина инициализации, то это ошибка вычисления. Исходное значение является текущем значением строки задержки в позиции *p*1**SR*, где *SR* является частотой дискретизации оркестра. Если *p*1*SR не будет целым числом, то исходное значение должно быть интерполировано из соседних значений. Если *p*2 оценен, то он не используется и игнорируется.

Если *method* - 3, то он нормирован как «set». В этом случае *p*1 является позицией вставки в секундах, и *p*2 является значением вставки. Если *method* 1 для этого кода операции еще не вызвали, или если *p*1 не оценен, или если *p*1 меньше чем 0, или если *p*1 больше чем новая длина инициализации, или если *p*2 не оценен, то это ошибка вычисления. Значение строки задержки в позиции *floor* (*p*1 * *SR*), где *SR* является частотой дискретизации оркестра, обновляется в *p*2. Исходное значение равно 0.

Если *method* - 4, то он нормирован как «and into». В этом случае *p*1 является позицией вставки в секундах, и *p*2 является значением вставки. Если *method*1 для этого кода операции еще не вызвали, или если *p*2 не оценен, то это ошибка вычисления. Пусть *x* будет текущем значением строки задержки в позиции *floor* (*p*1 * *SR*), где *SR* является частотой дискретизации оркестра. Тогда значение строки задержки в этой позиции обновляется в *x* + *p*2. Исходное значение равно *x* + *p*2.

Если *method* - 5, то он нормирован как «shift». Если *method*1 для этого кода операции еще не вызвали, то это — ошибка вычисления. Все значения строки задержки смещаются вперед на одну выборку. Исходное значение является значением, смещенным из конца строки задержки, которая является текущим значением выборки *L*. Если *p*1 и *p*2 оценены и не используются, то они игнорируются.

9.14 Эффекты

9.14.1 *reverb*

*aopcode reverb (asig x, ivar f*0 *[, ivar r*0*, ivar f*1*, ivar r*1*, ivar ...])*

Код операции ядра *reverb* производит эффект реверберации согласно заданным параметрам.

Если значение *f* или *r* отрицательно, или если число параметров четное и больше чем 2, то это ошибка вычисления.

Если только одно значение *f*0 дается как параметр, то оно берется в качестве времени реверберации, то есть время задерживается до тех пор пока звуковая амплитуда не ослабнет до 60 дБ по сравнению с исходным звуком (*RT*60).

Если дано большее значение, пары *f* — *r* представляют отчеты в различных частотах. В каждой частоте *f,* данной в качестве параметра, время реверберации (RT60) в этой частоте дается соответствующим значением *r*.

Точный метод вычисления реверберации согласно указанным параметрам не нормирован.

Вывод должен быть отраженным звуковым сигналом.

9.14.2 *chorus*

asig chorus (asig x, ksig rate, ksig depth)

Код операции *chorus* создает звук с эффектом хора, с уровнем *rate* и глубиной *depth* от входного звука *x*. Уровень *rate* определяют в циклах в секунду. *depth* определяется как процент отклонения с 0 <=*depth* <=100.

9.14.3 *flange*

asig flange (asig x, ksig rate, ksig depth)

Код операции *flange* создает звук с боковым эффектом, с уровнем *rate* и глубиной *depth* от входного звука *x*. Уровень *rate* определяется в циклах в секунду. *depth* определяется как процент отклонения с 0 <=*depth* <=100.

9.14.4 *fx_speedc*

kopcode fx_speedc (ivar speed_control_factor)

Код операции *fx_speedc* создает звук с эффектом изменения скорости. Этот код операции доступен только в обрабатывающих эффекты оркестрах. Это не может использоваться в блоке *SAOL* потока битов структурированного аудио.

Код операции *fx_speedc* получает доступ к специальной шине *input_bus* и руководит изменением скорости в *k* уровне. Затем обработанные выборки, сохраненые в специальном буфере, определенном только для регулировки скорости, выводятся как сигналы уровня *k/speed_control_factor*, где *k* = *SR/KR*, где *SR* является частотой дискретизации оркестра и *KR* является уровнем управления оркестра.

Точный метод изменения скорости не нормирован и открыт для разработчиков.

9.14.5 *speedt*

iopcode speedt (table in, table, ivar factor)

Код операции *speedt* изменяет звуковую выборку в соответствии с масштабом времени.

Код операции *speedt* заполняет звуковую выборку в звуковой таблице со звуком, полученным из звуковой таблицы *in*, растягивая ее во времени. Если *factor* <1, звук сжимается (ускоренный), если *factor* > 1 звук расширяется (замедленный).

Точный метод изменения скорости не нормирован.

9.15 Функции темпа

9.15.1 *gettempo*

Opcode gettempo ([xsig dummy])

Код операции *gettempo* обратен значению текущего темпа оркестра в ударах в минуту. По умолчанию темп равен 60 ударам в минуту, но может быть изменен с помощью строки *tempo* или кода операции *settempo*.

9.15.2 *settempo*

kopcode settempo (ksig x)

Код операции *settempo* изменяет значение темпа оркестра. Параметр *x* определяет новый темп в ударах в минуту. Если *x* не строго положителен, то это ошибка вычисления. Исходным значением является *x*.

10 Генераторы звуковой таблицы *SAOL*

10.1 Введение

Все генераторы звуковой таблицы должны быть реализованы в терминале, который может декодировать объект тип 3 или 4.

Для каждого генератора звуковой таблицы описывается следующее:

Используемые описания, показывающие параметры которые обязаны быть в таблице которой пользуется звуковой генератор .

Семантика генератора. Семантика описывает, как вычислить значения и разместить их в звуковую таблицу, используя этот генератор.

Для каждого генератора звуковой таблицы первое поле в табличном определении является именем генератора, а значение выражения во втором поле является размером звуковой таблицы. Многие генераторы звуковой таблицы также показывают значение – 1 в поле динамического вычисления для размера звуковой таблицы.

Последующие выражения являются необходимыми и дополнительными параметрами к генератору. Каждому из этих полей дадут имя в описании генераторов.

У каждой звуковой таблицы, так же как блока данных, есть четыре параметра: цикл частоты дискретизации и запуска, конец цикла, и основная частота. Для всех генераторов звуковой таблицы кроме *sample* эти параметры первоначально должны быть обнулены.

10.2 Выборка

*t*1 *table (sample, size, which [, skip])*

Код генератора звуковой таблицы *sample* позволяет включение аудиосэмплов (или другие блоки данных) в потоке битов и последующий доступ в оркестр.

Если *size* будет –1, то размер таблицы должен быть длиной аудиосэмпла. Если *size* больше, чем длина аудиосэмпла, то аудиосэмпл должен быть дополнен нулем в конце длины *size*. Если *size* меньше, чем длина аудиосэмпла, то должны использоваться только первые выборки *size*.

Поле *which* идентифицирует выборку. *which* это любой символ. Когда генератор обращается к выборке в потоке битов это символ числа. Когда генератор обращается к выборке, сохраненной в *AudioBuffer* это число.

В случае, где генератор обращается к выборке в потоке битов, для совместимых реализаций потока битов, данные являются просто потоком необработанных значений с плавающей точкой. Новый блок данных с именем должен быть помещен в звуковую таблицу. Если блок данных потока битов содержит частоту дискретизации, цикл запуска и конец цикла, и/или значения основной частоты, то эти параметры звуковой таблицы должны быть установлены. Если частота дискретизации не будет оценена, то она должна быть установлена в частоту дискретизации оркестра по умолчанию. Любые другие, не оцененные параметры должны быть установлены к 0.

В случае, когда генератор обращается к выборке, сохраненной в *AudioBuffer*, для сжатия выборки может использоваться любой аудио кодер. В этом случае поля *children* узла *AudioSource*, ответственного за инстанцирование этого оркестра, обращаются к узлам *AudioBuffer*. Каждый *AudioBuffer* после буферизации содержит несколько каналов аудиоданных. Если у первого элемента есть каналы n_0, вторые каналы n_1 и до k-1, то у узла *AudioSource* есть всего $K = n_0 + n_1 + ... + n_{k-1}$ каналов и *which* должен быть значением между 0 и K-1. Канал *which* (где *which* в случае необходимости округляется в самое близкое целое число), нумеруется в порядке элементов и их каналов которые, и будут помещены в звуковую таблицу. Частота дискретизации звуковой таблицы должна быть установлена в частоту дискретизации узла *AudioBuffer*, из которыго берется канал *which*. Цикл запускается, конец цикла и значения основной частоты должны быть установлены в 0.

Если *skip* обеспечивается и является положительным значением, то он округляется к самому близкому целому числу, и данные, помещенные в звуковую таблицу, начинаются с выборки *skip*+1 потока битов или данными из *AudioBuffer*.

10.3 Данные

*t*1 *table (data, size, p*1, *p*2, *p*3...*)*

Код генератора звуковой таблицы *data* позволяет оркестру помещать значения данных непосредственно в звуковую таблицу.

Если *size* будет –1, то размер таблицы должен быть числом определенных значений данных. Если *size* будет дан и будет больше, чем число значений данных, то звуковая таблица в конце длины *size* должна быть дополнена нулем. Если *size* будет дан и будет меньше, чем число значений данных, то должны использоваться только первые значения *size*.

*p*1, *p*2, *p*3... поля являются значениями с плавающей точкой, которые должны быть помещены в звуковую таблицу.

10.4 *Random*

*t*1 *table (random, size, dist, p*1 *[, p*2*])*

Код генератора звуковой таблицы *random* заполняет звуковую таблицу псевдослучайными числами согласно данному распределению. Для всех псевдослучайных алгоритмов генерации числа должны быть повторно отобраны после запуска оркестра так, чтобы каждое исполнение оркестра, содержащего эти инструкции, генерировало различные числа.

Если поле *size* будет положительным значением, то оно должно быть длиной таблицы, и много независимых случайных чисел должны быть вычислены и помещены в таблицу.

Поле *dist* определяет, какое случайное распределение использовать, и соответственно изменяются значения полей *p*1 и *p*2.

Если *dist* равно 1, то используется универсальное распределение. Псевдослучайные числа вычисляются так, что у всех значений с плавающей точкой между *p*1 и *p*2 включительно есть равная вероятность того, чтобы быть выбранным для любой выборки.

Если *dist* равно 2, то используется линейно умещающееся распределение. Псевдослучайные числа вычисляются так, что функции распределения вероятности *x* для любой выборки дают

p(x) = 0 *если* $x \le p1$ *или* $x > p2$, и *abs (2/ (p*2 – *p*1*)* x *[(x* – *p*1*)/ (p*2 – *p*1*)])* в других случаях.

Если *dist* равно 3, то используется экспоненциальное распределение. Псевдослучайные числа вычисляются так, что функция распределения вероятности *x* для любой выборки

$p(x) = 0$ если $x \le 0$, и $k\ exp\ (-\ kx)$, где $k = 1/\ p1$ в других случаях.

Если *dist* равно 3, то *p*2 не используется и игнорируется.

Если *dist* равно 4, то используется Гауссово распределение. Псевдослучайные числа вычисляются так, что, функция распределения вероятности x для любой выборки

$$p\ (x) = \frac{e^{-(mean-x)^2/(2var)}}{\sqrt{2\pi x\ var}},$$

то есть, $p\ (x) \sim N\ (p1, p2)$, где *p*1 является средним значением и *p*2 значением нормального распределения.

Если *dist* равно 4, то *p*2 должен быть строго больше чем 0.

Если *dist* равно 5, то моделируется процесс Пуассона, где среднее число выборок дается экспоненциальным распределением со средним значением *p*1. Псевдослучайное значение вычисляется согласно *p(x)* как для *dist* равным 3 (экспоненциальное распределение). Это значение округляется к самому близкому целому числу *y*. Первые значения *y* из таблицы (элементы от 0 до *y*-1) устанавливаются в 0, и следующее значение (элемент *y*) в 1. Другое псевдослучайное значение вычисляется как будто *dist* равно 3, и округляется к самому близкому целому числу *z*. Следующие значения *z* (элементы от *y* + 1 до *y* + *z*) устанавливаются в 0, и следующее значение (элемент *y* + *z* + 1) в 1. Этот процесс повторяется, пока таблица не переполнена элементами *size*. Получающиеся таблицы имеют размеры *size* независимые от значений, сгенерированных в псевдослучайном процессе; последний элемент может быть нулем или 1.

Если *dist* равно 5, то *p*2 не используется и игнорируется.

10.5 *Step*

*t*1 *table (step, pазsize, x*1, *y*1, *x*2, *y*2,...)

Код генератора звуковой таблицы *step* позволяет произвольным ступенчатым функциям быть помещенными в звуковую таблицу. Ступенчатая функция вычисляется из пар значений (*x, y*).

Если *size* будет равен -1, то размер звуковой таблицы должен быть размером самого большого параметра значения *x*. Если *size* больше, чем самый большой параметр значения *x*, то звуковая таблица должна быть дополнена 0 значениями в конце размера *size*. Если *size* будет меньшим, чем самое большое значение *x*, то должны быть вычислены и использоваться только первые значения *size*.

Если *x*1 не 0, значения *x* не являются неубывающей последовательностью или есть четное число параметров, не считая параметр *size*, то это ошибка вычисления.

10.6 *Lineseg*

*table t*1 *(lineseg, size, x*1, *y*1, *x*2, *y*2...)

Код генератора звуковой таблицы *lineseg* позволяет произвольным линейным функциям быть помещенными в звуковую таблицу. Линейная функция вычисляется из пар значений (x, y).

Если *size* будет равен -1, то размер звуковой таблицы должен быть размером самого большого параметра значения *x*. Если *size* больше, чем самый большой параметр значения *x*, то звуковая таблица должна быть дополнена 0 значениями в конце размера *size*. Если *size* будет меньшим, чем самое большое значение *x*, то должны быть вычислены и использоваться только первые значения *size*.

Если *x*1 не 0, значения *x* не являются неубывающей последовательностью или есть нечетное число параметров, не считая параметр *size*, то это ошибка вычисления.

10.7 *Expseg*

*t*1 *table (expseg, размер, x*1, *y*1, *x*2, *y*2...)

Код генератора звуковой таблицы *expseg* позволяет произвольным экспоненциальным функциям быть помещенными в звуковую таблицу. Функция вычисляется из пар значений (*x, y*).

Если *size* будет равен -1, то размер звуковой таблицы должен быть размером самого большого параметра значения *x*. Если *size* больше, чем самый большой параметр значения *x*, то звуковая таблица должна быть дополнена 0 значениями в конце размера *size*. Если *size* будет меньшим, чем самое большое значение *x*, то должны быть вычислены и использоваться только первые значения *size*.

Если *x*1 не 0, значения *x* не являются неубывающей последовательностью, есть нечетное число параметров, не считая параметр *size*, и если все значения *y* не имеют один знак или любое значение *y* равно 0, то это ошибка вычисления.

10.8 *Cubicseg*

*t*1 *table (cubicseg, size, infl*1, *y*1, *x*1, *y2, infl*2, *y*3, *x*2, *y*4, *infl*3, *y*5,...)

Код генератора звуковой таблицы *cubicseg* создает функцию, составленную из сегментов кубических полиномиалов. Каждый сегмент определяется с точки зрения конечных точек и точки перегиба.

Если *size* будет равен -1, то размер звуковой таблицы должен быть размером самого большого параметра значения *x*. Если *size* больше, чем самый большой параметр значения *x*, то звуковая таблица должна быть дополнена 0 значениями в конце размера *size*. Если *size* будет меньшим, чем самое большое значение *x*, то должны быть вычислены и использоваться только первые значения *size*.

Если *infl*1 не 0, значения *x* не являются неубывающей последовательностью, любое значение *infl* не между двумя соседними значениями *x*, значение *infl* меньше двух значений *x* или последовательность значений управления не заканчивается значением *y*, то это ошибка вычисления.

10.9 *Spline*

*t*1 *table (spline, size, x*1, *y*1, *k*2, *x*2, *y*2, *k*3, ...)

Код генератора звуковой таблицы *spline* создает гладкую переменную функцию для ряда контрольных точек.

Если *size* будет равен -1, то размер звуковой таблицы должен быть размером самого большого параметра значения *x*. Если *size* больше, чем самый большой параметр значения *x*, то звуковая таблица должна быть дополнена 0 значениями в конце размера *size*. Если *size* будет меньшим, чем самое большое значение *x*, то должны быть вычислены и использоваться только первые значения *size*.

Если *x*1 не 0, значения *x* не являются неубывающей последовательностью, значений *x* меньше двух значений *x*, количество параметров, не включая *size*, меньше 4 или последний параметр не является значением *k*, то это ошибка вычисления.

10.10 *polynomil*

*t*1 *table (polynomil, size, xmin, xmax, a*0, *a*1, *a*2...)

Код генератора звуковой таблицы *polynomil* позволяет произвольному разделу произвольной полиномной функции быть помещенным в звуковую таблицу. Используется полиномная функция $p(x) = a0 + a1x + a2x^2 + \ldots$. Она оценивается в диапазоне $[x_{min}, x_{max}]$.

Если *size* не положителен, количество параметров, не включая *size*, меньше 3 или если $x_{min} = x_{max}$, то это ошибка вычисления.

10.11 *Window*

*t*1 *table (window, size, type [, p])*

Код генератора звуковой таблицы *window* позволяет функции работы с окнами быть помещенной в таблицу.

Это — ошибка времени выполнения, если параметр *size* не строго положителен, или если *type* = 5 или параметр *p* не включается.

Тип окна определяется параметром *type*. Этот параметр должен быть округлен к самому близкому целому числу, и затем интерпретирован следующим образом:

Если *type* = 1, должно использоваться окно Ханнинга. Для демонстрационного номера *x* в диапазоне [0, *size* – 1] значение, помещенное в таблицу, должно быть

$0{,}54 - 0{,}46 \cos(2\pi x/(size -))$.

Если *type* = 2, должно использоваться окно Ханнинга. Для демонстрационного номера *x* в диапазоне [0, *size* – 1] значение, помещенное в таблицу, должно быть

$0{,}50\ (1 - cos\ (2\pi x\ /\ (size - 1)))$.

Если *type* = 3, должно использоваться Бартлетт (треугольное) окно. Для демонстрационного номера *x* в диапазоне [0, *size* – 1] значение, помещенное в таблицу, должно быть

$1 - 2 \mid x - (size - 1) / 2 \mid / (size - 1)$.

Если *type* = 4, должно использоваться Гауссово окно. Для демонстрационного номера x в диапазоне [0, size 1], значение, помещенное в таблицу, должно быть

$e^{-c1\ (c2-x)\ (c2-x)}$, где $c2 = size\ /2$ и $c1 = 18\ /\ (size)^2$.

Если type = 5, должно использоваться окно Кайзера с параметром *p*. Для демонстрационного номера *x* в диапазоне [0, *size* – 1] значение, помещенное в таблицу, должно быть

$$\frac{I_0\left[P\sqrt{\left(\frac{SIZE-1}{2}\right)^2-\left(x-\frac{SIZE-1}{2}\right)^2}\right]}{I_0\left[P\left(\frac{SIZE-1}{2}\right)\right]},$$

где $I_0(x)$ является модифицированной функцией Бесселя нулевого порядка первого вида.

Если *type* = 6, должно использоваться окно серии длинных импульсов. Каждая выборка в диапазоне [0, *size* – 1] должна иметь значение 1.

10.12 *Harm*

*t*1 *table (harm, size, f*1, *f*2, *f*3...)

Код генератора звуковой таблицы *harm* создает один цикл составной формы сигнала, составленной из суммы нулевых фазовых синусоид.

Если *size* не строго положителен, то это ошибка вычисления.

Для каждой выборки *x* в диапазоне [0, *size* –1] должно быть присвоено значение

*f*1 sin *(2* π *x/size)* + *f*2 sin *(4*π *x/size)* + *f3* sin *(6* π *x/size)* +...

10.13 *Harm_phase*

*t*1 *table (harm_phase, size, f*1, *ph*1, *f*2, *ph*2, ...)

Код генератора звуковой таблицы *harm_phase* создает один цикл составной формы сигнала, составленной из суммы нулевых *DC* синусоид, каждая из имеет начальную фазу в радианах.

Если *size* не строго положителен или если нечетное число параметров, не считая параметр *size*, то это ошибка вычисления.

Для каждой выборки *x* в диапазоне [0, size –1] должно быть присвоено значение

*f*1 sin *(2* π *x/size* + *ph*1*)* + *f*2 sin *(4* π *x/size* + *ph2)* + *f*3 *sin (6* π *x/size* + *ph*3*)* +...

10.14 *Periodic*

*t*1 *table (periodic, size, p*1, *f*1, *ph*1, *p*2, *f*2, *ph*2, ...)

Код генератора звуковой таблицы *periodic* создает один цикл сигнала произвольной периодической формы, параметризованной как сумма нескольких синусоид с произвольной частотой, амплитудой и фазой. Фазовые значения (*ph*1, *ph*2...) определяются в радианах.

Если *size* не строго положителен или если число параметров, не считая параметр *size*, не делится на три, то это ошибка вычисления.

Для каждой выборки *x* в диапазоне [0, *size* –1] должно быть присвоено значение *f*1 sin(2*p*1π*x/size* + + *ph*1)+*f*2sin(2*p*2π*x/size* + *ph*2)+*f*3 sin(2*p*3π*x/size* +*ph*3)+...

10.15 *Buzz*

*t*1 *table (buzz, size, nharm, lowharm, rolloff)*

Код генератора звуковой таблицы *buzz* создает один цикл суммы серии спектрально обрезанного косинуса *partials* (ограниченная полосой последовательность импульсов).

Если *size* и *nharm* не строго положительны, то это ошибка вычисления.

Перед дальнейшей обработкой *lowharm* и *nharm* должны быть округлены к самому близкому целому числу.

Если *size* не строго положителен, то размер таблицы дается самой высокой включенной гармоникой, так что *size* = 2 (*lowharm* + *nharm*).

Если *nharm* не будет строго положителен, то число гармоник должно быть дано размером таблицы так, что *nharm* является самым большим целым числом, меньшим чем *sizel*2 – *nharm*.

Для каждой выборки *x* в диапазоне [0, *size*–1] должно быть присвоено значение

$$scale * \sum_{f=lowharm}^{lowharm+\ nharm} rolloff^{\ f-lowharm} \cos 2\ \pi fp$$

где *p* является значением *x* / size и *scale* являются значением (1–abs (*rolloff*)) / (1–abs (*rolloff*^{nharm})).

10.16 *Concat*

*table t*1 *(concat, size, ft*1, *ft*2, ...)

Код генератора звуковой таблицы *concat* позволяет нескольким таблицам быть связанными вместе в новую таблицу.

Если *size* не будет строго положителен, то размер звуковой таблицы должен быть суммой параметров *size* звуковых таблиц. Если *size* будет строго положительным, но меньшим, чем сумма параметров *size* звуковых таблиц, то только первые параметры точки *size* звуковых таблиц должны использоваться. Если *size* будет больше, чем сумма параметров *size* звуковых таблиц, то сгенерированные звуковые таблицы должны быть дополнены нулем в конце *size*.

Значения звуковой таблицы должны быть вычислены следующим образом: для каждой выборки *x* в диапазоне [0, s_1-1], где s_1 является размером звуковой таблицы, на которую ссылается *p*1, должно быть присвоено то же самое значение, как выборке *x* для *p*1. Для каждой выборки *x* в диапазоне [s_1, *s*1+*s*2-1], где *s2* является размером звуковой таблицы, на которую ссылается *p*2, должно быть присвоено то же самое значение, как выборке *x* - s_1 *p*2 и так далее.

10.17 *Emply*

*t*1 *table (emply, size)*

Код генератора звуковой таблицы *empty* выделяет место и заполняет его нулями.

Если *size* не строго положителен, то это ошибка вычисления.

Для каждой выборки в диапазоне [0, *size* 1] присваивается 0.

11 Синтаксис и семантика *SASL*

11.1 Введение

SASL позволяет описывать простые события, которые использует оркестр чтобы генерировать звук, включая примечания, контроллеры и динамическую генерацию звуковой таблицы.

Все случаи в файле счета (время запуска и продолжительность) определяются в *score time*. Каждая строка счета может быть снабжена предисловием с опцией * тег. Этот тег указывает, что событие является высокоприоритетным событием.

11.2 Синтаксическая форма

<score file> -> <score line> [<score file>] <score file> -> <score line>
<score line> -> (*) <instr line> <newline>
<score line> -> (*) <control line> <newline>
<score line> -> (*) <tempo line> <newline>
<score line> -> (*) <table line> <newline>
<score line> -> <end line> <newline>
<instr line> -> [<ident> :] <number> <ident> <number> <pflist>
<control line> -> <number> [<ident>] control <ident> <number>
<tempo line> -> <number> tempo <number>
<table line> -> <number> table <ident> <ident> <pflist>
<end line> -> <number> end
<pflist> -> <number> [<pflist>]
<pflist> -> <NULL>

11.3 *Instr line*

instr line определяет конструкцию для инструментального инстанцирования в установленное время.

Первый идентификатор является меткой, которая используется, чтобы идентифицировать инстанцирование для использования с дальнейшими контрольными событиями.

Первое число является временем события. Время определяется с большой точностью. Инструменты диспетчеризируются с такой же скоростью, как уровень управления оркестра.

Второй идентификатор (первый необходимый идентификатор) является именем используемого инструмента, чтобы выбрать его из оркестра, описанного в элементе потока битов *SAOL*. Если в оркестре нет инструмента с этим именем, когда оркестр запускается,то это синтаксическая ошибка,.

Второе число является продолжительностью счета инструмента. Когда инструментальное инстанцирование будет создано, событие завершения должно быть запланировано в то время, которое дано суммой времени инстанцирования и продолжительности. Если это поле будет равно –1, то у инструмента не должно быть никакой запланированной продолжительности.

pflist является списком полей параметра, которые передадут инструменту, когда он создается. Если будет больше *pfields*, определенных в инструментальном объявлении, чем элементов этого списка, то после инстанцирования остающийся *pfields* должен быть установлен в 0. Если будет меньше *pfields,* чем элементов, то дополнительные элементы должны быть проигнорированы.

11.4 *Control line*

Control line определяет команду управления, которую передадут оркестру или ряду инструментов.

Первое число является временем счета события управления. Когда это время прибывает в оркестр, событие управления диспетчеризируется согласно его определенной семантике.

Первый идентификатор является меткой, определяющей, какие инструменты должны использоваться. Если эта метка обеспечивается, когда событие управления диспетчеризируется, то любые активные инструменты, которые создавались *instr* событиями с той же самой меткой, получают событие управления. Если метка будет обеспечена, и нет таких активных инструментов, то событие управления должно быть проигнорировано. Если метка не обеспечивается, то событие управления ссылается на глобальную переменную оркестра.

Второй идентификатор (первый необходимый идентификатор) является именем переменной, которое получит событие. Для маркированных строк управления имя ссылается на переменную в инструментах, которые создавались на основе *instr* событий с той же самой меткой. Если не будет такого имени в определенном инструменте, то событие управления должно быть проигнорировано для того экземпляра. Для немаркированных строк имя ссылается на глобальную переменную оркестра с тем же самым именем. Если не будет такой глобальной переменной, то событие управления должно быть проигнорировано.

Второе число является новым значением для переменной управления.

11.5 *Tempo line*

Tempo line определяет новый темп для процесса декодирования. Темп определяется в ударах в минуту. Темп по умолчанию должен быть шестьюдесятью ударами в минуту, и таким образом по умолчанию время счета измеряется в секундах.

Первое число в строке темпа является временем счета, в котором изменяется темп.

Второе число является новым темпом, определенным в ударах в минуту.

11.6 *Table line*

Table line определяет создание или разрушение звуковой таблицы.

Первое число в строке является временем, в которое звуковая таблица создается или уничтожается.

Первый идентификатор является именем звуковой таблицы. Это имя ссылается на звуковую таблицу в глобальном контексте оркестра.

Второй идентификатор является или именем генератора таблиц, или специальным именем *destroy*.

Список *pfield* является списком параметров генератора звуковой таблицы.

Генератор звуковой таблицы *sample* обращается к звуковой выборке. При реализациии текстового интерфейса необходимо обеспечить доступ к обычно используемым форматам "звукового файла" в первом списке *pfield*. Однако, это не нормировано. Нормировано только следующее. В потоке битов *table* может быть установлен бит *refers_to_sample*. Если это верно, тогда маркер *sample* этого объекта должен обратиться к другому объекту потока битов, содержащему контрольные данные, и именно эти контрольные данные должны быть помещены в звуковую таблицу.

Когда получается диспетчеризировать время табличного события, и если табличная строка ссылается на имя *destroy*, то любая глобальная звуковая таблица с тем именем может быть уничтожена, и его память

освобождается. Если табличная строка определяет создание звуковой таблицы, и уже есть глобальная звуковая таблица с тем же самым именем, то новая звуковая таблица заменяет существующую звуковую таблицу. Глобальная звуковая таблица с тем именем может быть уничтожена, и его память освобождается.

Когда новая таблица должна быть составлена, выделяется пространство памяти для таблицы и заполняется данными согласно определенного генератора звуковой таблицы. Любая ссылка на звуковую таблицу с этим именем в существующих или новых инструменах должна быть взята в качестве направления к новой звуковой таблице.

11.7 *End line*

Строка *end line* определяет конец процесса звуковой генерации. Данное число является временем окончания для оркестра. Когда это время достигается, оркестр прекращается, и все будущие буферы, основанные на этом структурированном процессе декодирования аудио содержат только 0 значений.

12 Маркировка *SAOL/SASL*

12.1 Введение

Этот подпункт описывает процесс отображения между текстовым форматом *SAOL*, используя описание синтаксиса и семантику, и маркируемый поток битов.

12.2 *SAOL tokenisation*

Чтобы промаркировать текстовый оркестр *SAOL*, должны выполняться следующие шаги. Во-первых, оркестр должен быть разделен на лексические элементы, где лексический элемент является одним из следующего:

1. Знак препинания,
2. Зарезервированное слово,
3. Стандартное имя,
4. Базовое имя кода операции,
5. Базовое имя генератора звуковой таблицы,
6. Символьная константа (строка, целое число, или константа с плавающей точкой),
7. Идентификатор.

Пробел может использоваться, чтобы разделить лексические элементы. Это требуется, чтобы лексически снять неоднозначность оркестра. Пробел не будет обработанным, как лексический элемент оркестра. Комментарии могут использоваться в текстовом оркестре *SAOL*, но удаляются после лексического анализа. Комментарии не сохраняются через *tokenisation/detokenisation* последовательность.

После лексического анализа все идентификаторы в оркестре должны быть пронумерованы со значениями символа так, чтобы один символ был связан с определенным текстовым идентификатором. Все идентификаторы, которые эквивалентны, должны быть связаны с тем же самым символом независимо от их синтаксического контекста. Эту ассоциацию символов к идентификаторам вызывают таблицей символов.

Используя лексический анализ и таблицу символов, может быть произведено маркируемое представление оркестра. Лексический анализ сканируется в порядке, как это было представлено в текстовом представлении, и для каждого лексического элемента:

- Если элемент имеет тип (1) — (5), значение маркера берется в таблице в Приложении А.
- Если элемент будет иметь тип (6), то в зависимости от типа символьной константы должен быть произведен один из специальных маркеров 0xF1, 0xF2, 0xF3, 0xF4. Для целочисленных констант в диапазоне [0,255] маркер может быть 0xF1 или 0xF4.
- Если элемент будет иметь тип 7, то должен быть произведен специальный маркер 0x*F*0, и последующий элемент потока битов должен быть символом, связанным с идентификатором в таблице символов.

После того, как последовательность лексических элементов, представленных в текстовом оркестре, маркируется, должен быть в конце оркестра произведен специальный маркер 0x*FF*.

12.3 *SASL tokenisation*

SASL должен маркироваться относительно определенного оркестра *SAOL*, так как для семантики значения символа должны соответствовать.

Чтобы маркировать файл *SASL*, делаются следующие шаги. Во-первых, файл *SASL* делится на лексические элементы, где каждый элемент является или идентификатором, зарезервированным словом,

именем базового генератора звуковой таблицы, или числом. После лексического анализа каждый идентификатор должен быть связан с соответствующим числом символа из ссылки оркестра *SAOL*. Таким образом, для оркестра *SAOL*, если в оркестре есть идентификатор эквивалентный идентификатору в счете, идентификатор в счете должен получить то же самое число символа, которое было получено в оркестре. Если в оркестре нет такого идентификатора, то любое неиспользованное число символа может быть присвоено идентификатору в счете.

Используя лексический анализ и таблицу символов, может быть произведено маркируемое представление оркестра. Каждая строка счета проводится поочередно, в порядке, представленном в текстовом представлении.

13 Синтаксис и семантика банка выборок

13.1 Введение

Этот подпункт описывает работу метода синтеза банка выборок для объектов типа 2 и 4. В объекте типа 2 в потоке битов должны появиться только банк выборок и *MIDI,* и этот подпункт описывает процесс генерирования звука от элемента данных потока битов банка и последовательности инструкций *MIDI*. В объекте типа 4 банки используются в контексте инструмента *SAOL*, и этот подпункт описывает процесс генерирования звука и возврата к процессу декодирования *SAOL*, в зависимости от элемента данных потока битов банка и определенных обращением *sasbf*.

13.2 Элементы потока битов

Элемент потока битов *sasbf* является блоком данных, определенных структурой файла *MIDI DLS*.

13.3 Процесс декодирования

13.3.1 Объект типа 2

13.3.1.1 Краткий обзор

В объекте типа 2 весь синтез выполняется посредством синтеза банка звуковой таблицы. Управление с помощью стандартного элемента потока битов *MIDIFile* и элемента потока битов команды *MIDI*.

13.3.1.2 Каналы, демонстрационный формат и частота дискретизации

Чтобы присоединить декодер объекта типа 2, структурированное аудио к узлу *AudioBIFS AudioSource*, у получающегося аудиопотока должно быть два канала, 32-разрядный с плавающей запятой и с частотой дискретизации 22050 Гц. Вычисление не обязано происходить в 32-разрядных выборках стерео. Если внутренний формат будет иным, то после того, как декодирование будет выполнено, результат должен быть преобразован в этот формат.

13.3.1.3 Конфигурация декодера

В потоковом заголовке (элемент конфигурации декодера) могут появиться один или более блоков *sbf*. Блоки *sbf* передают к синтезатору *SASBF*, который использует эти данные, чтобы подготовить их к синтезу в соответствии с его семантикой.

13.3.1.4 Декодирование времени выполнения

Два типа событий могут управлять синтезом времени выполнения в объекте типа 2: кэшируемые события *MIDI*, которые были переданы как файл *MIDI* в потоковом заголовке, и события *MIDI* в реальном времени.

Decoding clock сохраняются, чтобы управлять диспетчеризацией событий, но точные свойства этих часов не нормативны. В каждом шаге планировщик *MIDI* должен диспетчеризировать любые события *MIDI*, которые пришли в потоке битов с декодированием отметки времени меньше чем текущее значение часов декодирования, так же как любые события *MIDI*, упорядоченные в блоках *midi_file* в потоковом заголовке, развернутые метки времени которого меньше чем текущее значение часов декодирования.

Интерактивные манипуляции к полю *speed* узла *AudioSource*, указывающего на этот процесс декодирования, влияют на скорость воспроизведения кэшируемых событий *Standard MIDIFile*, но не имеют никакого эффекта для диспетчеризирования потоковой передачи событий *MIDI*.

Результирующий звук, описанный процессом синтеза в *MIDI* диспетчеризируется планировщиком. Эти звуковые выборки обеспечиваются для узла *AudioSource*, который ссылается на этот поток битов как на вывод структурированного аудио декодера объекта типа 2.

13.3.2 Объект типа 4

13.3.2.1 Краткий обзор

Данные *MIDI,* в объекте типа 4 непосредственно не управляют синтезатором *sasbf*, но диспетчеризируют звуки в *SAOL*. Функцию диспетчеризации выполняет *sasbf* оператор.

13.3.2.2 Конфигурация декодера

При работе объектов типа 4, *sbf* блоки данных в заголовке конфигурации потока битов передаются к синтезатору *sasbf*, где они используются, чтобы подготовка к синтезу была в реальном времени.

13.3.2.3 Декодирование времени выполнения

При работе объектов типа 4 синтез каждого звука выполняется отдельно. Звук оформляется при команде, содержащейся в *sasbf* выражении. Это выражение содержит звук, скорость, предварительную установку и значения выборки банка. Синтез одного звука, обозначенного предварительно установленным числом и банком, выполняется для этого звука и скорости согласно инструменту *sasbf*. Получающийся звук стерео возвращается *sasbf* выражением.

Декодер *sasbf* должен использовать *MIDI*-контроллер и другую непрерывно изменяющуюся информацию о *MIDI* для определенного канала. Эти данные не передают непосредственно в синтезатор *SASBF* в *sasbf* команде.

14 Семантика *MIDI*

14.1 Введение

Этот подпункт описывает процесс декодирования для реализаций объектов типа 1, и отображение событий MIDI в заголовке информации о потоке и данных потока битов в семантику *SAOL* для реализаций объектов типа 3 и 4.

14.2 Процесс декодирования объектов типа 1

В потоке битов объекта типа 1 должны быть только элементы потока битов *midi* и *midi_file*.

14.3 Отображение событий *MIDI* в управление оркестра

14.3.1 Введение

Для объектов типа 3 и 4 кодированные события, когда они будут получены в терминале, как часть события стандартного *MIDI FILE* или *MIDI*, должны быть преобразованы в данные *MIDI* и соответствующую семантику планировщика. Эта семантика применяется только к объектам типа 3 и 4, и не к объектам типа 1 и 2.

14.3.2 События MIDI

14.3.2.1 Введение

Этот подпункт описывает семантику событий различных типов, которые могут быть в потоке битов объекта *MIDI_event*.

14.3.2.2 Расширенные значения канала

У фактического события *MIDI Channel* есть номер канала в диапазоне 0 … 15. Каждый *MIDI* имеет входной порт, выходной порт или отслеживаемый блок, связывается с потоком или набором событий *MIDI* и соответствующим набором 16 каналов (некоторые из которых могут быть не использованы). Приложения *MIDI* обычно используют имена порта, отслеживают имена или другие метки для идентификации различных каналов.

Чтобы избежать потребности в таких метках набора канала, в MPEG–4 используются расширенные номера каналов. В MPEG–4 значение *channel MIDI_event* не ограничивается диапазоном 0 … 15. Вместо этого сгенерировано расширенное значение канала, основанное на исходном номере канала *MIDI* и на числе, связанном с портом или потоком, который является источником события.

14.3.2.3 *NoteOn*

noteon channel note velocity

Когда будет получено событие *noteon* с ненулевой скоростью, канал инструмента в оркестре нужно инстанцировать с продолжительностью – 1 и первыми двумя *p-fields* наборами *note* и *velocity*. Каждое значение *MIDIctrl* [] в инструменте устанавливается в новое значение изменения контроллера или в значение по умолчанию, если не было никаких изменений контроллера на том канале. Значение *MIDIbend* устанавливается в новое значение изменения *MIDI*. Значение *MIDItouch* устанавливается в новое значение корректировки на канале.

Инструмент, создаваемый в ответ на сообщение *noteon* на канале, упоминается как находящийся на этом канале.

Сообщения *noteon* со скоростью 0 должны быть обработаны как *noteoff* сообщения.

14.3.2.4 *NoteOff*

noteoff channel note velocity

Когда будет получено событие *noteoff*, каждый инструмент на канале, который инстанцировали со звуковым числом *note*, планируется для завершения в конце *k* цикла. То есть устанавливается флаг *released*, и инструмент не вызывает *extend*, его нужно деинстанцировать после текущего *k* цикла вычисления.

Если *MIDIctrl [64]* на обозначенном канале будет ненулевым, то выполнение события *noteoff* должно быть задержано, пока *MIDIctrl [64]* на обозначенном канале не станет нулем. Это поддерживается установкой значения *MIDIctrl [64]* в потоке битов или присвоением ему стандартного имя *MIDIctrl*.

14.3.2.5 *Control change*

cc channel controller value

Когда изменяется управление событием *cc,* новое значение контроллера устанавливается в *value*. Это значение должно кэшироваться так, чтобы на данном канале у будущих инструментов к нему был доступ. У всех активных в настоящий момент на канале инструментов должно быть стандартное имя *MIDIctrl [controller]* обновленное *value*.

14.3.2.6 *Aftertouch*

touch channel note velocity

Когда будет получено событие *touch* , значение переменной *MIDItouch* каждого инструмента на канале, который инстанцировали со звуковым числом *note*, устанавливается в *value*.

14.3.2.7 *Channel aftertouch*

ctouch channel velocity

Когда будет получено событие *ctouch*, значение переменной *MIDItouch* каждого инструмента на канале устанавливается в *velocity*.

14.3.2.8 Изменение программы

pchange channel program

Когда будет получено событие *pchange*, текущие значения инструмента на канале должны быть изменены на инструмент со значением числа *program*. Если нет никакого инструмента с этим числом, то будущие события на канале будут проигнорированы, пока не будут получены изменения программы.

14.3.2.9 *Bank select*

bankselect channel bank

Когда будет получено событие *bankselect*, текущие значения инструмента на канале должны быть изменены на инструмент с предварительно установленным числом *bank* * 128 + *program*. Событие *bankselect* не оказывает прямого влияния. Событие *bankselect* только изменяет значение будущего *pchange* события на канале.

14.3.2.10 *Pitch wheel change*

pwheel channel value

Когда будет получено событие *pwheel*, значение *MIDIbend* для каждого инструмента на канале должно быть установлено в *value*.

14.3.2.11 *All notes off*

notesoff

Когда будет получено событие *notesoff*, все события инструментов в оркестре, создаваемом *MIDI NoteOn,* планируются для завершения в конце текущего *k* цикла.

Если значение *MIDIctrl* [64] для инструмента будет ненулевым, то выполнение завершения должно быть задержано до тех пор пока значение *MIDIctrl* [64] не станет нулем.

Семантика значений соответствует поведению команды *MIDI All Notes Off*. Все значения *MIDIctrl* [123] обычно 0, и должны быть установлены в 1 для всех доступных *MIDIctrl*.

14.3.2.12 *Tempo change*

tempochange value

Когда будет получено событие *tempochange*, глобальный темп оркестра изменяется. Значение *value* указывает на количество ударов в минуту.

14.3.2.13 *All sound off*

soundoff

Когда будет получено событие *soundoff*, все инструменты в оркестре, создаваемом событиями *MIDI NoteOn*, завершаются в конце текущего *k* цикла. Значение *MIDIctrl* [120], обычно 0, и оно должно быть

установлено в 1 для всех доступных *MIDIctrl*. Весь звук обрабатывается так, чтобы динамические инструменты, порожденные от инструмента *MIDI*, могли обнаружить команду *All Sound Off*.

14.3.2.14 Сообщения *MIDI*

У следующих сообщений *MIDI* нет никакого значения в MPEG–4 для объекта типа 3 и 4:

Local Control
Omni Mode On/Off
Mono Mode On/Off
Poly Mode On/Off
System Exclusive
Tune Request
Timing Clock
Song Select/Continue/Stop
Song Position
Active Sensing
Reset

14.3.2.15 Ведущий канал *MIDI*

Все инструменты, создаваемые операторами *SASL instr* и *SAOL send,* и любые динамические инструменты, которые инстанцируются от инструментов, создаваемых *SASL instr* и *SAOL send*, видят состояние ведущего канала *MIDI* в значениях *MIDIctrl[], MIDIbend, MIDIwheel, channel* и стандартных именах *preset*. Для этих инструментов значение стандартного имени *channel* отражает расширенный номер ведущего канала *MIDI*, и значение стандартного имени *preset* отражает значение последнего *pchange* события в ведущем канале *MIDI*. Значения *MIDIctrl*[], *MIDIbend*, *MIDIwheel* идентичны значениям в инструментах, которые инстанцируются через событие *MIDI NoteOn* от ведущего канала *MIDI*.

Идентификационные данные ведущего канала *MIDI* определяются следующим образом:

Если источник *MIDI* непосредственно соединяется с оркестром, то ведущий канал *MIDI* является 0 каналом *MIDI* от этого источника.

Если никакой источник *MIDI* непосредственно не соединяется с оркестром, и если источник *MIDI SA_access_unit* присутствует, ведущий канал *MIDI* является 0 каналом от этого источника.

Если никакой источник *MIDI* непосредственно не соединяется с оркестром, и если никакой источник *MIDI SA_access_unit* не присутствует, ведущий канал *MIDI* является 0 каналом из первой дорожки файла *MIDI*, который содержит команды *Noteoff, NoteOn, CChange, PChange, Pwheel, Touch* или *CTouch* для 0 канала.

14.3.3 Стандартные *MIDI* файлы

14.3.3.1 Введение

У файлов *MIDI* есть данные с той же самой семантикой, как у сообщений *MIDI*, однако семантика синхронизации усложняется из-за использования многократных дорожек и дельта- временных меток.

14.3.3.2 Краткий обзор *MIDI* файла обработки

Чтобы обработать элемент потоковый информационный *midi_file*, должны быть сделаны следующие шаги. Во-первых, весь потоковый элемент анализируется и кэшируется. Затем, используя инструкции и дельта времена, различные события *midi_file* преобразовываются во времена события счета (в ударах). Во время этого шага фактические номера каналов этих событий также отображаются, как расширенные значения *channel*. Преобразование дельта времен требует преобразования каждого блока дорожки *midi_file* в *timelist*, содержащей объекты *midi_event,* и затем в чередующиеся различные дорожки *timelists*.

14.3.3.3 Преобразование файла *MIDI,* отслеживающего блоки

Чтобы преобразовать блок дорожки в *timelist,* сначала анализируют блок дорожки, чтобы сгенерировать серию событий *MIDI*. Это требует преобразования дельта-времен файла *MIDI* ко временам счета события *MIDI* относительно начала каждого блока дорожки. Это также необходимо, чтобы отобразить номера каналов события *midi_file,* как расширенные значения *channel midi_event*. Когда событие *Set Tempo midi_file* будет обработано, чтобы сгенерировать соответствующее изменение темпа *midi_event*, значение темпа должно быть преобразовано из микросекундных модулей в модули ударов в минуту. Если файл *MIDI* не определит запускающий *tempo*, то по умолчанию должно использоваться значение 120 ударов в минуту.

Поскольку события преобразовываются от *MIDI* до семантики *SAOL*, то каждое событие должно быть зарегистрировано в планировщике согласно времени события и его семантике.

14.3.3.4 Преобразование *MIDI* каналов в каналы планировщика

Отображение каналов событий в файлах *MIDI* в *midi_event channel* выполняется следующим образом. Последовательные блоки дорожки в пределах *midi_file* являются присвоенными номерами дорожек

в возрастающей монотонной последовательности, с начальным номером дорожки 0. Значения *channel* для определенного *midi_file* события дают:

*channel = midi_file event channel number + (track number * 16).*

Если есть многократные блоки *midi_file* в пределах заголовка потока битов, то у последующих элементов *midi_file* есть дорожки, пронумерованные последовательно от конца первого блока. Таким образом, если у i^{th} элемента *midi_file* есть k_i дорожки, где $0 < i < n$, тогда дорожки сначала *midi_file* нумеруются $0..\ k_0-1$, затем $k_0..\ k_1-1$ и т. д.

14.3.4 Значения контроллера по умолчанию

Таблица 3 дает значения по умолчанию для определенных непрерывных контроллеров. Значение контроллеров не перечисленых здесь по умолчанию должно быть нулем.

Таблица 3 — Значения *MIDI*-контроллера по умолчанию

Контроллер	Функция	Значение по умолчанию
1	*Mod whell*	0
5	*Portamento speed*	0
7	*Volume*	100
10	*Pan*	64
11	*Expression*	127
64	*Sustain pedal*	0
65	*Portamento on\off*	0
66	*Sostenuto*	0
67	*Soft pedal*	0
84	*Portamento control*	0
Pitch Bend	*Pitch bend*	8192

15 Входные звуки и отношение с *AudioBIFS*

15.1 Введение

Этот подпункт описывает использование *SAOL* оркестров для функции, обрабатывающей эффекты в *AudioBIFS* системе. *SAOL* используется не только в качестве метода описания синтеза звука, но также в качестве метода описания звуковых эффектов и для завершения алгоритмов.

15.2 Входные источники и *phaseGroup*

Каждый узел *BIFS*, который содержит код *SAOL*, является или узлом *AudioSource* или узлом *AudioFX*. Если нет никаких входных источников к оркестру *SAOL*, то глобальная переменная оркестра *inchannels* по умолчанию имеет значение 0. В этом случае специальная шина *input_bus* не может быть отправлена к инструменту или использоваться в оркестре.

Если дочерние узлы узла *AudioFX* обеспечивают несколько каналов входного звука оркестру, то каналы входного звука помещаются в специальную шину *input_bus*. От этой шины они могут быть отправлены любому требуемому инструменту (ам), и аудиоданные должны обрабатываться. Число каналов ввода оркестра по умолчанию — это значение глобальной переменной оркестра *inchannels,* является суммой чисел каналов звука, обеспеченного каждым из дочерних элементов.

15.3 Узел *AudioFX*

15.3.1 Введение

Узел *AudioFX* в *AudioBIFS* используется, чтобы загрузить алгоритмы, обрабатывающие звуковые эффекты в пределах комплекта инструментальных средств *AudioBIFS*. *SAOL* является языком для описания алгоритмов, обрабатывающих звуковые эффекты в *AudioBIFS*.

15.3.2 Параметры оркестра *AudioFX*

В оркестре *SAOL* из узла *AudioFX* инстанцируют только файл оркестра (поле *orch* в узле) и, дополнительно, файл счета *SASL* (поле *score*). Эти файлы соответствуют маркируемым последовательностям оркестра и данных счета, формирующих элементы потока битов *orchestra* и *score_file* .

15.3.3 Инстанцирование оркестра *AudioFX*

Инстанцирование оркестра для узла *AudioFX* требует следующих шагов:

1. Декодирование элементов *orch* и *score* в узле
2. Грамматический разбор и проверка синтаксиса этих элементов
3. Инстанцирование отправляет инструменты в оркестр.

Каждый из переданных инструментов должен сохраняться, пока он не будет выключен оператором *turnoff*, или узел, содержащий оркестр, не будет удален. Если оператор *turnoff* будет использоваться в одном из этих инструментов, то это должно быть взято в качестве нулевого значения для будущего времени.

15.3.4 Выполнение оркестра *AudioFX*

Процесс синтеза времени выполнения продолжается согласно правилам для стандартного процесса декодирования *SA* со следующими исключениями и дополнениями:

Поскольку никакие устройства доступа не будут получены процессом *AudioFX*, то не должна сохраняться связь с системным уровнем. Используются только события, которые находятся в поле счета узла непосредственно. Каждый раз оркестр *AudioFX* должен запросить из системного уровня входные буферы аудио, которые соответствуют дочерним узлам. Эти аудио буферы должны быть помещены в специальную шину *input_bus* и затем отправлены любым инструментам, которые определяются в глобальном заголовке оркестра.

Кроме того, каждый шаг уровня управления *params*[] поля узла *AudioFX* должен быть скопирован в глобальную переменную *params*[] массива оркестра. В конце каждого цикла управления *params*[] значения массива должны быть скопированы обратно в соответствующие поля узла *AudioFX* и затем направлены к другим узлам.

В каждом моменте времени вывод оркестра становится выводом узла *AudioFX*.

15.3.5 Функциональность изменения скорости в узле *AudioFX*

Функциональность изменения скорости для звуков, обеспеченных из входных источников, поддерживается в узле *AudioFX*. Это обеспечивает код операции *SAOL fx_speedc* .

15.4 Интерактивные 3-*D* пространственные аудио сцены

Когда узел *AudioSource* или *AudioFX* является дочерним элементом узла *Sound*, коды пространственного расположения, направления, образец распространения звука от узла *Sound* и пространственного расположения и направления слушателя прописываются в узле *SAOL*.

С этой целью используются стандартные имена *position*, *direction*, *listenerPosition*, *listenerDirection*, *minFront*, *maxFront*, *minBack* и *maxBack*.

Приложение А
(справочное)

Кодирование таблиц

А.1 Введение

Приложение содержит маркерную таблицу потока битов. Определенные маркеры обозначаются как *reserved*, что означает, что они в настоящий момент не используются в потоке битов, но могут использоваться в будущих версиях стандарта.

А.2 Маркерная таблица потока битов

Маркер	Текст	Маркер	Текст
0x00	*(reserved)*	0x3*A*	*inGroup*
0x01	*aopcode*	0x3*B*	*released*
0x02	*asig*	0x3*C*	*cpuload*
0x03	*else*	0x3*D*	*position*
0x04	*exports*	0x3*E*	*direction*
0x05	*extend*	0x3*F*	*listenerPosition*
0x06	*global*	0x40	*listenerDirection*
0x07	*if*	0x41	*minFront*
0x08	*imports*	0x42	*minBack*
0x09	*inchannels*	0x43	*maxFront*
0x0*A*	*instr*	0x44	*maxBack*
0x0*B*	*iopcode*	0x45	*params*
0x0*C*	*ivar*	0x46	*itime*
0x0*D*	*kopcode*	0x47	*(reserved)*
0x0*E*	*krate*	0x48	*channel*
0x0*F*	*ksig*	0x49	*input bus*
0x10	*map*	0x4*A*	*output bus*
0x11	*oparray*	0x4*B*	*startup*
0x12	*opcode*	0x4*C*-0x4*F*	*(reserved)*
0x13	*outbus*	0x50	&&
0x14	*outchannels*	0x51	| |
0x15	*output*	0x52	>=
0x16	*return*	0x53	<=
0x17	*route*	0x54	i =
0x18	*send*	0x55	= =
0x19	*sequence*	0x56	-
0x1*A*	*sasbf*	0x57	*
0x1*B*	*spatialize*	0x58	/
0x1*C*	*srate*	0x59	+
0x1*D*	*table*	0x5*A*	>
0x1*E*	*tablemap*	0x5*B*	<
0x1*F*	*template*	0x5*C*	?
0x20	*turnoff*	0x5*D*	:
0x21	*while*	0x5*E*	(
0x22	*with*	0x5*F*	)
0x23	*xsig*	0x60	{
0x24	*interp*	0x61	}
0x25	*preset*	0x62	[
0x26-0x2*F*	*(reserved)*	0x63	]
0x30	*k_rate*	0x64	;
0x31	*s_rate*	0x65	,
0x32	*inchan*	0x66	=
0x33	*outchan*	0x67	!
0x34	*time*	0x68-0x6*E*	*(reserved)*
0x35	*dur*	0x6F	*sample*
0x36	*MIDIctrl*	0x70	*data*
0x37	*MIDItouch*	0x71	*random*
0x38	*MIDIbend*	0x72	*step*
0x39	*input*	0x73	*lineseg*

0x74	*expseg*
0x75	*cubicseg*
0x76	*polynomial*
0x77	*spline*
0x78	*window*
0x79	*harm*
0x7A	*harm phase*
0x7B	*periodic*
0x7C	*buzz*
0x7D	*concat*
0x7E	*empty*
0x7F	*(reserved)*
0x80	*int*
0x81	*frac*
0x82	*dbamp*
0x83	*ampdb*
0x84	*abs*
0x85	*exp*
0x86	log
0x87	sqrt
0x88	sin
0x89	cos
0x8A	*atan*
0x8B	*pow*
0x8C	log10
0x8D	*asin*
0x8E	*acos*
0x8F	*floor*
0x8D	*asin*
0x90	*ceil*
0x91	*min*
0x92	*max*
0x93	*pchoct*
0x94	*octpch*
0x95	*cpspch*
0x96	*pchcps*
0x97	*cpsoct*
0x98	*octcps*
0x99	*pchmidi*
0x9A	*midipch*
0x9B	*octmidi*
0x9C	*midioct*
0x9D	*cpsmidi*
0x9E	*midicps*
0x9F	*sgn*
0xA0	*ftlen*
0xA1	*ftloop*
0xA2	*ftloopend*
0xA3	*ftsetloop*
0xA4	*ftsetend*
0xA5	*ftbasecps*
0xA6	*ftsetbase*
0xA7	*tableread*
0xA8	*tablewrite*
0xA9	*oscil*
0xAA	*loscil*
0xAB	*doscil*
0xAC	*koscil*
0xAD	*kline*
0xAE	*aline*
0xAF	*sblock*
0xB0	*kexpon*
0xB1	*aexpon*
0xB2	*kphasor*
0xB3	*aphasor*
0xB4	*pluck*
0xB5	*buzz*
0xB6	*grain*
0xB7	*irand*
0xB8	*krand*
0xB9	*arand*
0xBA	*ilinrand*
0xBB	*klinrand*
0xBC	*alinrand*
0xBD	*iexprand*
0xBE	*kexprand*
0xBF	*aexprand*
0xC0	*kpoissonrand*
0xC1	*apoissonrand*
0xC2	*igaussrand*
0xC3	*kgaussrand*
0xC4	*agaussrand*
0xC5	*port*
0xC6	*hipass*
0xC7	*lopass*
0xC8	*bandpass*
0xC9	*bandstop*
0xCA	*fir*
0xCB	*iir*
0xCC	*firt*
0xCD	*iirt*
0xCE	*biquad*
0xCF	*fft*
0xD0	*ifft*
0xD1	*rms*
0xD2	*gain*
0xD3	*balance*
0xD4	*decimate*
0xD5	*upsamp*
0xD6	*downsamp*
0xD7	*samphold*
0xD8	*delay*
0xD9	*delay1*
0xDA	*fracdelay*
0xDB	*comb*
0xDC	*allpass*
0xDD	*chorus*
0xDE	*flange*
0xDF	*reverb*
0xE0	*compressor*
0xE1	*gettune*
0xE2	*settune*
0xE3	*ftsr*
0xE4	*ftsetsr*
0xE5	*gettempo*
0xE6	*settempo*
0xE7	*fx speedc*
0xE8	*speedt*
0xE9-0xEF	*(reserved)*
0xF0	*<symbol>*
0xF1	*<number>*
0xF2	*<integer>*
0xF3	*<string>*
0xF4	*<byte>*
0xF5-0xFF	*(free)*
0xFF	*<EOO>*

Приложение Б
(справочное)

Кодирование

Б.1 Введение

Приложение обеспечивает инструкции для создания типичного структурированного аудио кодера. Приложение описывает возможные функции инструментов для создания потока битов. Методы, описанные здесь носят справочный характер.

Б.2 Основное кодирование

Б.2.1 Введение

Этот подпункт описывает работу *basic encoder*. Основной структурированный аудио кодер берет, как входной компонент, модули потока битов и преобразовывает их в узаконенное представление потока битов.

Предполагается, что компонентные модули находятся в следующих форматах: *SAOL* и программы *SASL* в соответствующих текстовых форматах. Звуковые выборки индивидуально сохранены в компьютерном формате звукового файла, таком как *AIFF* или *WAVE*. Данные *MIDI* хранятся, как стандартный файл *MIDI*. Банки *sasbf* сохранены, как файлы двоичных данных.

Шаги, требуемые в создании потока битов, следующие: *tokenisation SAOL* и программ *SASL*, дизассемблирования звуковых выборок блока информации о конфигурации декодера и реорганизации счета и событий *MIDI* в потоковую передачу данных.

Б.2.2 *Tokenisation* данных *SAOL*

Этот процесс преобразовывает программу *SAOL*, данную в текстовом формате, в двоичный блок данных. Во время этого процесса, перечисляя имена инструментов, определяемых пользователем кодов операции, звуковых таблиц и сигнальных переменных в оркестре, может быть создана *symbol table,* связывающая каждого из них с числовым значением. Эта таблица может быть включена в заголовок конфигурации декодера потока битов.

Б.2.3 *Tokenisation* данных *SASL*

Этот процесс преобразовывает программу *SASL,* данную в текстовом формате, в двоичный блок данных. Любые символы, используемые в счете *SASL,* могут быть включенны в таблицу символов, если они созданы в процессе, описанном в подпункте Б.2.2.

Б.2.4 Дизассемблирование звуковых выборок

Звуковые выборки, сохраненные как компьютерные звуковые файлы, дизассемблируются в блоки демонстрационных значений. Не допустимо включать звуковые выборки с отформатированными данными (такие как *AIFF* или файл *WAVE*) непосредственно в структурированный аудио поток битов. Длина (в выборках), частота дискретизации (в Гц), основная частота (в Гц), цикл запуска и конечные точки (в демонстрационном числе) считаются от отформатированной информации в компьютерном звуковом файле. Звуковые выборки преобразовываются из любого формата, если они были сохранены в компьютерном звуковом файле в любом 16-разрядном целочисленном значении со знаком (то есть, значения масштабируются в диапазон [-32768, 32767]), или в 32-разрядное значение с плавающей запятой со знаком. Любой формат может использоваться для выборки в структурированном аудио потоке битов.

Б.2.5 Блок информации о конфигурации декодера

Заголовок конфигурации декодера создается согласно формату от маркируемого оркестра *SAOL*, и одного или более маркируемого множества *SASL*, звуковых выборок, файлов *MIDI* и банков *sasbf*. Блоки могут быть в любом требуемом порядке, и индексируются битовыми полями *more_data* и *chunk_type*. Бит каждого события счета *high_priority* в заголовке может быть установлен для каждой строки счета или для любого требуемого набора важных событий.

Б.2.6 Блок потоковой передачи потока битов

В структурированном аудио формате потока битов передавать данные потоком в форме устройств доступа строго не требуется. Вся информация, запрошенная для того чтобы декодировать, может присутствовать в заголовке конфигурации декодера.

Звуковые выборки, события счета и команды *MIDI* могут все быть включены в часть данных потоковой передачи структурированного аудио потока битов. Звуковая выборка включается, упаковывая звуковые данные, после того, как это было дизассемблировано от компьютерного формата звукового файла в устройство доступа. Событие счета может быть включено с или без метки времени. Если событие счета включается с меткой времени, то это подвергается внутреннему управлению темпом оркестра. Если событие счета включается без меток времени, это легче перенести на уровне устройства доступа. Если нет никакой явной метки времени, синхронизацией события управляет информация о синхронизации в устройстве доступа. В этом случае может использоваться тег *use_if_late*.

Команды *MIDI* сначала преобразовываются из стандартного формата файла *MIDI* в представленные *MIDI* данные. Чтобы выполнить это, абсолютное время каждого события в стандартном *MIDIFile* вычисляется согласно синтаксису и семантике. Затем события не включаются с дельта-временами, а помещаются непосредственно в устройстве доступа так, чтобы информация о синхронизации в устройстве доступа управляла синхронизацией для событий *MIDI*. Данные *MIDI* в каждом устройстве доступа в потоке битов являются тем же самым, которые передаются в протоколе *MIDI*.

Приложение В
(справочное)

Грамматика *LEX/YACC* для *SAOL*

В.1 Введение

Это приложение описывает грамматики, используя широко доступные инструменты *'lex'* и *'yacc'*, которые соответствуют спецификации *SAOL*.

Ссылочное программное обеспечение для структурированного аудио создает лексический и синтаксический анализатор для грамматики *SAOL*, увязывая их с большим количеством структур данных.

В.2 Лексическая грамматика для *SAOL*

```
STRCONST        \"(\\.|[^\\\"])*\"
IDENT           [a-zA-Z ][a-zA-Z0-9_ ]*
INTGR           [0-9]+
NUMBER          [0-9]+(\.[0-9]*)?(e[-+]?[0-9]+)?|-?\.[0-9]*(e-+?[0-9]+)?
%{
void comment(void);
%}
%%
"//"            { comment(); }
"aopcode"       { return(AOPCODE) ; }
"asig"          { return(ASIG) ; }
"else"          { return(ELSE) ; }
"exports"       { return(EXPORTS) ; }
"extend"        { return(EXTEND) ; }
"global"        { return(GLOBAL) ; }
"if"            { return(IF) ; }
"imports"       { return(IMPORTS); }
"inchannels"    { return(INCHANNELS) ; }
"instr"         { return(INSTR); }
"interp"        { return(INTERP); }
"iopcode"       { return(IOPCODE); }
"ivar"          { return(IVAR) ; }
"kopcode"       { return(KOPCODE); }
"krate"         { return(KRATE) ; }
"ksig"          { return(KSIG) ; }
"map"           { return(MAP) ; }
"oparray"       { return(OPARRAY) ; }
"opcode"        { return(OPCODE) ; }
"outbus"        { return(OUTBUS) ; }
"outchannels"   { return(OUTCHANNELS) ; }
"output"        { return(OUTPUT) ; }
"preset"        { return(PRESET) ; }
"return"        { return(RETURN) ; }
"route"         { return(ROUTE) ; }
"send"          { return(SEND) ; }
"sequence"      { return(SEQUENCE) ; }
"sasbf"         { return(SASBF) ; }
"spatialize"    { return(SPATIALIZE) ; }
"srate"         { return(SRATE); }
"table"         { return(TABLE); }
"tablemap"      { return(TABLEMAP); }
"template"      { return(TEMPLATE); }
"turnoff"       { return(TURNOFF); }
"while"         { return(WHILE); }
"with"          { return(WITH); }
"xsig"          { return(XSIG) ; }
"&&"            { return(AND); }
"||"            { return(OR); }
```

```
">="                { return(GEQ); }
"<="                { return(LEQ); }
" !="               { return(NEQ); }
"= ="               { return(EQEQ); }
"-"                 { return(MINUS); }
"*"                 { return(STAR); }
"/"                 { return(SLASH); }
"+"                 { return(PLUS); }
">"                 { return(GT); }
"<"                 { return(LT); }
"?"                 { return(Q); }
":"                 { return(COL); }
"("                 { return(LP); }
")"                 { return(RP); }
"{"                 { return(LC); }
"}"                 { return(RC); }
"["                 { return(LB); }
"]"                 { return(RB); }
";"                 { return(SEM); }
","                 { return(COM); }
"= "                { return(EQ); }
"!"                 { return(NOT); }
{STRCONST}          { yytext[strlen(yytext)-1] = 0; /* strip quotes */
                    yylval = strdup(&yytext[1]);
                    return(STRCONST); }
{IDENT}             { yylval = strdup(yytext); return(IDENT) ; }
{INTGR}             { yylval = strdup(yytext); return(INTGR) ; }
{NUMBER}            { yylval = strdup(yytext); return(NUMBER) ; }
[ \t\n\r]           { /* whitespace */ }
.                   { printf("Line %d: Unknown character: '%s'\n",
                        yyline,yytext); } /* parse error */
%%
void commen t()     {
char c;
while ((c = input()) != '\n'); /* skip */
yyline++;
thisline[0] = 0;
yycol = 0;
}
```

B.3 Грамматика синтаксиса для *SAOL* в *YACC*

Код для структурированного аудио использует эту грамматику, как основную при анализе конструкцией синтаксического дерева.

Этот подпункт разрешает выражения в предварительных установках и списках карты. Чтобы поддерживать полный шаблонный синтаксис, лексический анализатор, который обнаруживает выражения в предварительной установке и карте, перечисляет и предпринимает специальные меры если, это необходимо.

```
*/
%token      IDENT INTGR NUMBER STRCONST AOPCODE ELSE EXPORTS EXTEND GLOBAL
%token      IF IMPORTS INCHANNELS INTERP
%token      INSTR IOPCODE IVAR TABLE KOPCODE KRATE KSIG ASIG MAP
%token      OPARRAY OPCODE OUTBUS OUTCHANNELS OUTPUT ROUTE SEND SEQUENCE
%token      SRATE TEMPLATE TURNOFF WHILE WITH XSIG AND OR GEQ LEQ
%token      NEQ EQEQ MINUS STAR SPATIALIZE SASBF TABLEMAP
%token      SLASH PLUS GT LT Q COL LP RP LC RC LB RB SEM COM EQ RETURN NOT
%token      ARRAYREF OPCALL IMPEXP VARDECL NOTAG SPECIALOP PRESET
%
%start      orcfile
%right      Q
%left       OR
%left       AND
%left       EQEQ NEQ
%left       LT GT LEQ NEQ
%left       PLUS MINUS
```

```
%left          STAR SLASH
%right         UNOT UMINUS
%token         HIGHEST
%%
Orcfile        : proclist
;
Proclist       : proclist instrdecl
               | proclist opcodedecl
               | proclist globaldecl
               | proclist templatedecl
               |/* null */
               | error;
               ;
instrdecl      : INSTR IDENT LP identlist RP miditag LC vardecls block RC
               ;
miditag        : PRESET int_list
               | /* null */
               ;
int list       : int list INTGR
               | INTGR
               ;
opcodedecl     : optype IDENT LP paramlist RP LC opvardecls block RC
               ;
globaldecl     : GLOBAL LC globalblock RC
               ;
templatedecl:  TEMPLATE LT identlist GT /* with preset */
               PRESET mapblock
               LP identlist RP
               MAP LC identlist RC
               WITH LC mapblock RC LC
               vardecls block RC
               | TEMPLATE LT identlist GT /* no preset */
               LP identlist RP
               MAP LC identlist RC
               WITH LC mapblock RC LC
               vardecls block RC
               ;
mapblock       : mapblock COM LT terminal_list GT
               | LT terminal_list GT
               |
               ;
terminal list  : terminal_list COM terminal
               | terminal
               ;
terminal       : IDENT
               | const
               | STRCONST
               ;
globalblock    : globalblock globaldef
               | /* null */
               ;
globaldef      : rtparam
               | vardecl
               | routedef
               | senddef
               | seqdef ;
rtparam        : SRATE INTGR SEM
               | KRATE INTGR SEM
               | INCHANNELS INTGR SEM
               | OUTCHANNELS INTGR SEM
               | INTERP INTGR SEM
               ;
```

```
routedef        : ROUTE LP IDENT COM identlist RP SEM
                ;
senddef         : SEND LP IDENT SEM exprlist SEM namelist RP SEM ;
seqdef          : SEQUENCE LP identlist RP SEM ;
block           : block statement
                | /* null */
                ;
statement       : lvalue EQ expr SEM
                | expr SEM
                | IF LP expr RP LC block RC
                | IF LP expr RP LC block RC ELSE LC block RC
                | WHILE LP expr RP LC block RC
                | INSTR IDENT LP exprlist RP SEM
                | OUTPUT LP exprlist RP SEM
                | SPATIALIZE LP exprlist RP SEM
                | OUTBUS LP IDENT COM exprlist RP SEM
                | EXTEND LP expr RP SEM
                | TURNOFF SEM
                | RETURN LP exprlist RP SEM
                ;
lvalue          : IDENT
                | IDENT LB expr RB
                ;
identlist       : identlist COM IDENT
                | IDENT
                | /* null */
                ;
paramlist       : paramlist COM paramdecl
                | paramdecl
                |  /* null */
                ;
vardecls        : vardecls vardecl
                | /* null */
                ;
vardecl         : taglist stype namelist SEM
                | stype namelist SEM
                | tabledecl SEM
                | TABLEMAP IDENT LP identlist RP SEM
                ;
opvardecls      : opvardecls opvardecl
                | /* null */
                ;
opvardecl       : taglist otype namelist SEM
                | otype namelist SEM
                | tabledecl SEM
                | TABLEMAP IDENT LP identlist RP SEM
                ;
paramdecl       : otype name
                ;
namelist        : namelist COM name
                | name
                ;
name            : IDENT
                | IDENT LB INTGR RB
                | IDENT LB INCHANNELS RB
                | IDENT LB OUTCHANNELS RB
                ;
stype           : IVAR
                | KSIG
                | ASIG
                | TABLE
                | OPARRAY
                 ;
```

```
otype          : XSIG
               | stype
               ;
tabledecl      : TABLE IDENT LP IDENT COM exprstrlist RP
               ;
taglist        : IMPORTS
               | EXPORTS
               | IMPORTS EXPORTS
               | EXPORTS IMPORTS
               ;
optype         : AOPCODE
               | KOPCODE
               | IOPCODE
               | OPCODE
               ;
expr           : IDENT
               | const
               | IDENT LB expr RB
               | SASBF LP exprlist RP
               | IDENT LP exprlist RP
               | IDENT LB expr RB LP exprlist RP | expr Q expr COL expr %prec Q
               | expr LEQ expr
               | expr GEQ expr
               | expr NEQ expr
               | expr EQEQ expr
               | expr GT expr
               | expr LT expr
               | expr AND expr
               | expr OR expr
               | expr PLUS expr
               | expr MINUS expr
               | expr STAR expr
               | expr SLASH expr
               | NOT expr    %prec UNOT
               | MINUS expr
               | LP expr RP
               ;
exprlist       : exprlist COM expr
               | expr
               /* null */
               ;
exprstrlist    : exprstrlist COM expr
               | exprstrlist COM STRCONST
               | STRCONST
               | expr ;
const          : INTGR | NUMBER
               ;
```

Приложение Г
(справочное)

Непосредственно соединенный *MIDI* и управление микрофоном оркестра

Г.1 Введение

Живое управление *MIDI* оркестра позволяет использовать в реальном времени структурированное аудио устройство декодера в качестве музыкального инструмента в ситуациях с записью или воспроизведением. Качество звука и гибкость структурированных аудио инструментов улучшается по сравнению с фиксированными аппаратными синтезаторами, используемые в таких ситуациях.

Г.2 Методы для *MIDI*-контроллера

Поскольку формат потока битов *MIDI* подобен управляющей информации MIDI, сгенерированной живым *MIDI* устройством (это точно инкапсулирует байты данных, сгенерированные таким устройством в *MPEG–4* устройстве доступа), то не должно быть доступа к структурированному аудио декодеру для включения управления оркестром такими устройствами. Должно быть выполнено соединение между вводом *MIDI* терминала и вводом планировщика, так чтобы события с меткой времени от ввода *MIDI* были переданы непосредственно к планировщику.

Рекомендуются следующие методы:

Ввод *MIDI* не должен быть преобразован в структурированный аудио поток битов, и должен генерировать события непосредственно в планировщике.

Любое разрешение на события, сгенерированные с живым *MIDI*-устройством, не должно быть оформлено в порядке, предписанном глобальными правилами упорядочивания. Инстанцирование примечания и первый *k* цикл для инструментов должны быть выполнены в текущей передаче оркестра, как можно скорее, после того, как они получаются оркестром. На второй *k* цикле они проходят через инструмент и он начинает обрабатывать их согласно глобальным правилам упорядочивания. Задержка между временем инициирования живым исполнителем и временем первого *k* цикла, в которое звук является слышимым, должно быть не больше 5 мс.

Живые события *MIDI* не должны подвергнуться управлению темпом оркестра.

Живые события *MIDI* должны быть обработаны оркестром, как любое другое событие *MIDI*. Потоковая передача производительности и живого выступления должны быть одновременны.

Если *MIDI* устройства многократно соединяются с одним и тем же терминалом, то нумерация канала должна быть такой, чтобы номер "*MIDI channel* 1" от устройства *A* отличался от номера "*MIDI channel* 1" от устройства *B*. С этой целью каждому *MIDI*-устройству присваиваются номер в возрастающей монотонной последовательности. Значением *channel* для определенного живого события *MIDI* будет:

channel = live MIDI event channel 1 + *(device number* * 16*)*.

Начальный номер устройства должен быть выбран так, чтобы присвоенные значения *channel* события *MIDI*, не конфликтовали с присвоенными значениями *channel*, события *midi_file* .

Г.3 Методы для живого микрофона

Аудиоданные, полученные микрофоном, помещаются в специальную шину *input_bus*, по которой они отправляются другим инструментам для обработки.

Рекомендуются следующие методы:

Определяется специальная шина *input_bus*. У этой шины есть число каналов, определенных *inchannels* глобальным параметром. Если *inchannels* глобальный параметр не определен, то эта шина не может использоваться.

В начале каждого цикла управления ввод микрофона выбирается на частоте дискретизации оркестра и помещается в специальную шину *input_bus*. Если каналов ввода микрофона больше, чем каналов на *input_bus*, то используются только первые каналы, а остальные отбрасываются. Если каналов ввода микрофона меньше, то "дополнительные" каналы *input_bus* устанавливаются в значение 0. Если нет никакого соединенного микрофона, то все каналы *input_bus* устанавливаются в значение 0.

input_bus обрабатывается, как любая другая шина в оркестре.

Библиография

[1] 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*)

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 *Е. В. Беспрозванная*
Корректор *Л. Я. Митрофанова*
Компьютерная верстка *Т. Ф. Кузнецовой*

Сдано в набор 16.04.2014. Подписано в печать 02.07.2014. Формат $60\times84^1/_8$. Бумага офсетная. Гарнитура Ариал. Печать офсетная. Усл. печ. л. 10,23. Уч.-изд. л. 8,70. Тираж 51 экз. Зак. 714

ФГУП «СТАНДАРТИНФОРМ», 123995 Москва, Гранатный пер., 4.
www.gostinfo.ru info@gostinfo.ru
Набрано и отпечатано в Калужской типографии стандартов, 248021 Калуга, ул. Московская, 256.