A Beginner's Guide to Programming Digital Audio Effects in the kX Project environment
Пособие начинающим программистам в области цифровых аудио эффектов под kx.

Martin Borisov (Tiger M)
Santiago Munoz (eYagos)
revised by
Max Mikhailov (Max M.)

the following parts are translated by
Alexander Pavlov (brainless_beginner)

перевод следующих частей
Александра Павлова (brainless_beginner)

ПРОЦЕССОР ЦИФРОВЫХ СИГНАЛОВ
DIGITAL SIGNAL PROCESSOR

Когда сигнал(в нашем случае аналоговый аудио сигнал)поступает на вход аудиокарты, он направляется на АЦП(ADC, аналого-цифровой преобразователь), который через постоянные промежутки времени(в нашем случае 48000 раз в секунду)преобразует сигнал в числа, тем самым, делая его цифровым. Получили 48000 чисел из 1 секунды сигнала. Затем каждое число из этого массива значений проходит через ПЦС(DSP), где и применяются эффекты. После этого, сигнал идет на ЦАП(DAC, цифро-аналоговый преобразователь), где становится вновь аналоговым, а оттуда на выход аудиокарты. Сигнал можно вместо ЦАП также послать на цифровые выходы(в цифровой форме).

Обработка сигнала эффектами происходит, когда он в цифровой форме(и представляет собой ряд значений), а это означает, что иметь дело мы будем только с математикой и ни с чем более. DSP эффекты – это лишь набор математических функций, большинство из которых известны нам со школы. Мат. операции выполняются над каждым значением цифрового сигнала 48000 раз в секунду.

ПРИСТУПИМ

Откройте редактор kx Editor(клик правой кнопкой мыши на значке kx на панели задач). В окне вы увидите строчки, начинающиеся с «;» -- это просто комментарии. Строчки с «name», «copyright» -- лишь для информации о плагине, можете вписать туда, что хотите. Строчка же «guid» важна. Она гарантирует уникальность плагина, чтобы новый не затер старый при регистрации. Новый номер генерируется автоматически при открытии редактора. Строчка «end» обязательна и означает конец кода.

DSP E-mu 10kx работает с 32 бит дробными от -1 до 1 или целыми числами.?????

РЕГИСТРЫ

Входящие, промежуточные и обработанные данные аудиопотока хранятся в физических регистрах.

1**. input** и **output** регистры. Входящие данные сохраняются в input регистрах, после чего могут быть обработаны. Обработанные данные сохраняются в output регистрах. Каждый микрокод может содержать несколько таких регистров, в зависимости от целей и нужного количества каналов(моно – по одному, стерео – по два, и т. д.).

Объявление:

Input in
Output out ; in и out просто имена. Назначить регистру можно любое имя.

2. **static** и **temp** регистры. Используются для хранения промежуточных данных во время выполнения микрокода. Значение регистра **static** хранится в нем до перезаписи или до переинициализации(при перезагрузке плагина или сбросе настроек). Регистры **temp** используются только в текущем цикле вычислений; при переходе к следующему циклу, их значения обнуляются. **Temp** регистры отличаются еще тем, что могут быть использованы сразу всеми загруженными эффектами, но в настоящее время кх-драйвером это не поддерживается. Рекомендуется использовать **static** в большинстве случаев.

Объявление:

Static st
Temp tmp ;st и tmp опять же просто имена

Регистры **static** и **temp** не требуют присвоения начального значения, но позволяют его задать, если это необходимо

Static st=0.5

При использовании чисел(констант) в микрокоде, в процессе компиляции они превращаются в **static** регистры.

3. **control** регистры. Значения этих регистров можно только считывать, и они требуют присвоения им начального значения. При компиляции для них автоматически создаются ползунки, которые можно передвигать мышью, изменяя тем самым их значения, которые в свою очередь могут варьироваться в пределах от 0 до 1 (0% - 100%).

Объявление:

Control volume=0.2 ;volume – просто имя.

Присвоенные значения являются значениями по умолчанию и будут устанавливаться каждый раз при загрузке или переинициализации плагина.

4. **accum** регистры. Эти регистры предоставляют доступ к накопителю DSP. При выполнении инструкции, ее результат автоматически сохраняется в нем, перезаписывая предыдущее значение, а затем присваивается регистру результата этой инструкции. Это значение вызывается при помощи слова «**accum**», и может быть использовано только в качестве операнда A. Эти регистры используют, когда необходимо не отсеченное и не свернутое промежуточное значение.

Например:

macs 0, 1, 0.5, 1 ;1+ 0.5 = 1.5
macsn out, **accum**, 0.6, 1 ;out=1.5-0.6=0.9

5. **CCR**(Condition Code Register) регистры. Используются для пропуска инструкций. Их значения устанавливается после каждой инструкции, опираясь на ее результат.

6. **TRAM Access Data Registers**. Используются для формирования задержек. Есть регистры записи(которые запоминают значения аудиопотока) и регистры чтения(которые воспроизводят запомненное).

ИНСТРУКЦИИ

Все инструкции имеют следующий синтаксис:

Результат инструкции **R**, операнд **A**, операнд **X**, операнд **Y**.

Операнды – это регистры.

1. **macs** и **macsn**. Произведение двух чисел прибавляется к третьему, и результат записывается в регистр результата. Эти инструкции работают только с дробными числами. Если результат превышает -1 или 1, то лишняя часть отсекается, оставляя -1 или 1. Вы не можете использовать целые числа, кроме «дробных» 0 и 1.

Формула:

Macs R, A, X, Y означает R=A+X*Y
Macsn R, A, X, Y означает R=A-X*Y

Пример:

Регулятор громкости

```
name "Volume control";
copyright "Copyright (c) 2004.";
created "06/27/2004";
engine "kX";
guid "...Будет автоматически сгенерирован!!!..."; не копируйте это

;мы объявляем регистры
input in
output out
control volume=1

macs out, 0, in, volume ;out = 0 + in * volume

end ;←не забудьте это.
```

Можете скопировать и вставить это в редактор. Кликните «save dane source»(«сохранить код в формате dane») (справа в окне), сохраните. Откройте окно DSP, щелкните правой кнопкой в любом месте и выберите «register plugin»(«зарегистрировать плагин»), выберите тот, который сохранили. Теперь можно его добавить: щелкните правой кнопкой в любом месте и выберите «add effect/plugin»(«добавить эффект/плагин») и выберите его.

2. **macw** и **macwn**. То же, что **macs** и **macsn**, только при превышении -1 или 1 результат сворачивается вокруг ноля. Например, при сложении 0.5 и 0.7 результат будет -0.8 (а не 1, как при отсечении).

3. **macints**(отсечение) и **macintw**(свертка). То же, что **macs** и **macw**, только умножение производится на целое число, т.е. эти инструкции автоматически подразумевают, что операнд Y – целое число.

Пример 1:

Информационная часть (name, copyright) отныне будет опущена.

Увеличение громкости в 4 раза:

```
input in
output out
macints out, 0, in, 0x4 ; out = 0 + in * 4
end
```

Пример 2:

То же с возможностью контроля:

```
input in
output out
control gain=0.25
static t

macints t, 0, in, 0x4 ; t = 0 + in * 4
macs out, 0, t ,gain ; out = 0 + t * gain

end
```

4. **acc3**. Инструкция суммирует 3 числа. Она производит отсечение при превышении -1 или 1. Значения операндов могут быть либо все 3 дробные, либо все 3 целые.

Формула:

Acc3 R, A, X, Y означает R=A+X+Y

Пример:

Микс трех моно источников плюс регулятор громкости:

```
input in1, in2, in3
output out
static t
control volume = 0.33

acc3 t, in1, in2, in3 ;t=in1 + in2 + in3
macs out, 0 , t , volume ; out = 0 + t * volume

end
```

5. **macmv**. Результат произведения X*Y прибавляется к предыдущему значению накопителя и сохраняется опять же в накопителе. В то же время, значение A копируется в R. Эта инструкция полезна при создании фильтров, т.к. она одновременно совмещает **mac** и сдвиг данных, необходимый для фильтрования.

Формула:

Macmv R, A, X, Y означает R=A и accum=accum+X*Y

6. **andxor**. Применение не придумано.

7. **tstneg**, **limit**, **limitn**. Предоставляют возможность использования условий.

Формулы:

Tstneg R, A, X, Y означает: если A>=Y то R=X, а если A<Y то R=-X

Эта инструкция позволяет получить абсолютное значение:

```
tstneg out, in, in, 0 ;out = abs(in)
```

limit R, A, X, Y означает: если A=>Y то R=X, а если A<Y то R=Y

limitn R, A, X, Y означает: если A=>Y то R=Y, а если A<Y то R=X

Пример:

Обрезание сигнала по амплитуде

```
input in
output out
static negclip=-0.05, posclip=0.05 ;отрицательный и положительный ограничители
control volume=0.8
static t
```

limitn t, posclip, posclip, in ;если in>posclip, t=posclip; иначе t=in
limit t,negclip, negclip, t ;если t<negclip, t=negclip; иначе t=t
macs out, 0, volume, t ;регулятор громкости

end

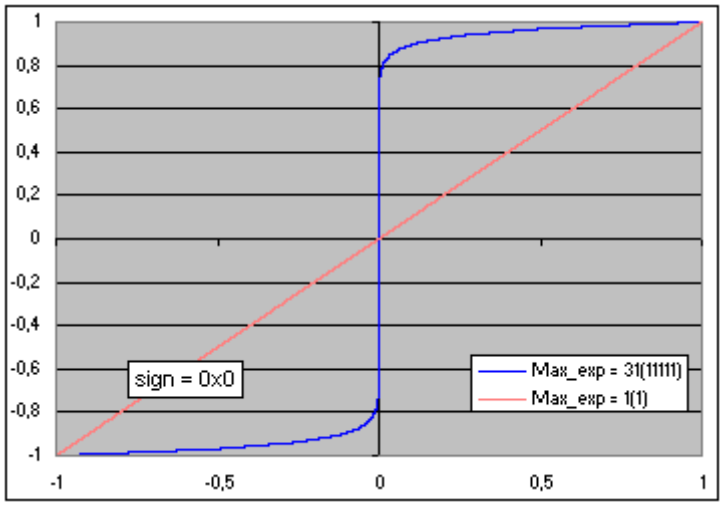
8. **log** и **exp**. **Log** преобразует линейные данные в форму: «знак-экспонента-мантисса», а **exp** выполняет обратное преобразование.

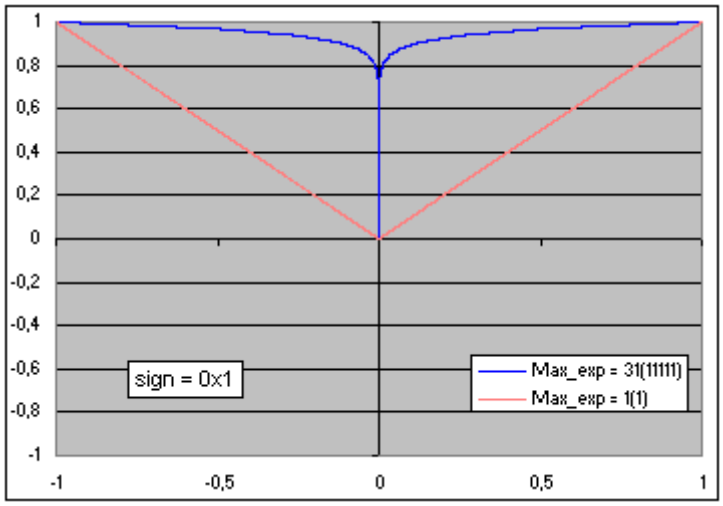
Формула:

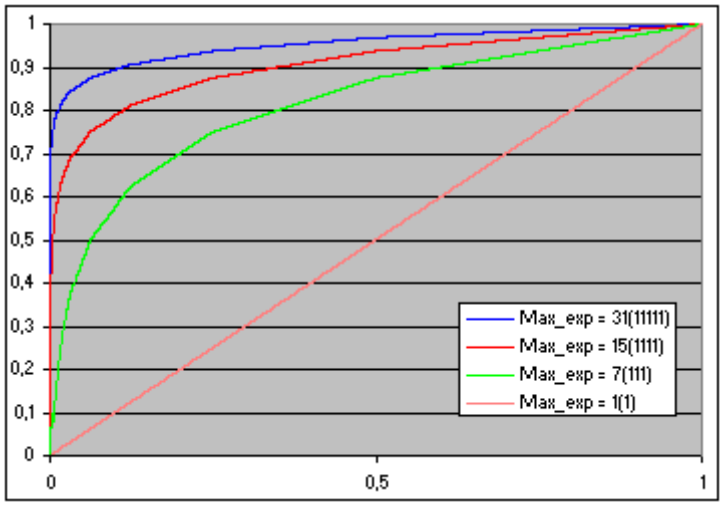
Log R, [входные линейные данные], [разрешение](Max_exp), [тип преобразования](sign)
Exp R, [входные «знак-экспонента-мантисса» данные], [разрешение], [тип преобразования](sign)

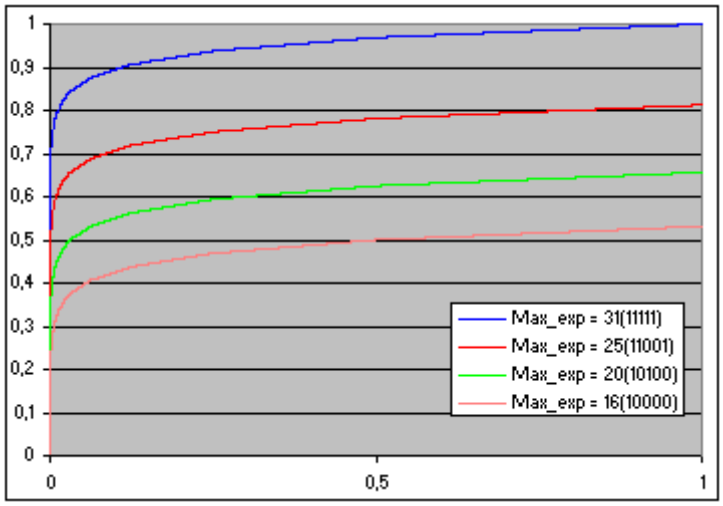
Входные линейные данные – данные для преобразования(дробные), указывается регистр, их содержащий.
Разрешение(Max_exp): значение этого параметра должно лежать в пределах от 1 до 31. Что он делает, будет ясно из графиков 3, 4, 5, 6.
Тип преобразования(sign): значение этого параметра должно лежать в пределах от 0 до 3. параметр указывает знак результата:
0 – нормальное(обычное)
1 – абсолютное значение(всегда положительный)
2 – минус абсолютное значение(всегда отрицательный)
3 – инверсия.

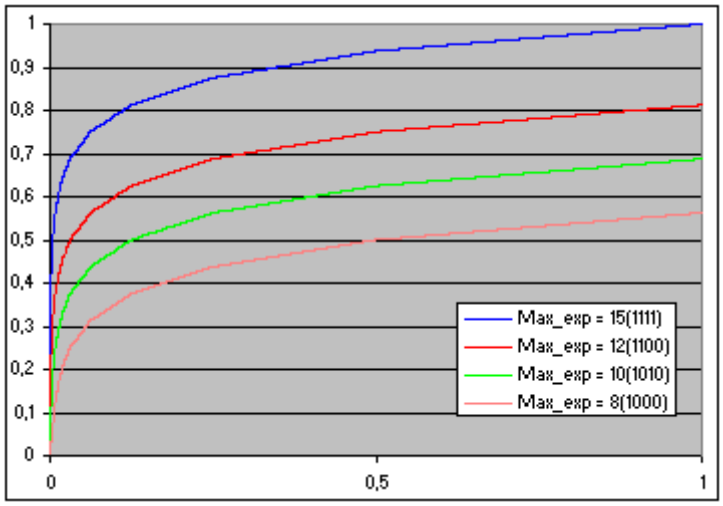
Графики для наглядности

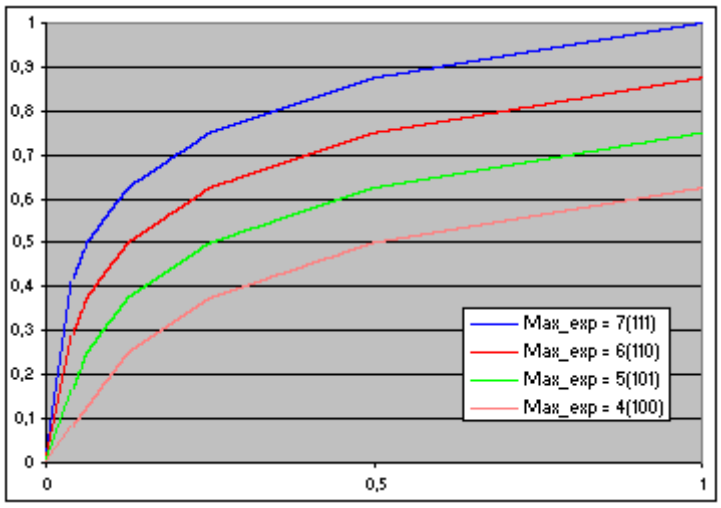



1
0,8
0,6
0,4
0,2
0
-0,2
-0,4
-0,6
-0,8
-1
-1
-0,5
0
0,5
1
sign = 0x1
Max_exp = 31(11111)
Max_exp = 1(1)


1
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
0
0,5
1
Max_exp = 31(11111)
Max_exp = 15(1111)
Max_exp = 7(111)
Max_exp = 1(1)


1
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
0
0,5
1
Max_exp = 31(11111)
Max_exp = 25(11001)
Max_exp = 20(10100)
Max_exp = 16(10000)





Почему эти инструкции называются **log/exp**? Потому что они являются достаточно точными приближениями этих математических функций, и чем больше параметр «разрешение», тем они точнее.

9. **interp**. Эта инструкция выполняет линейную интерполяцию между двумя точками. В основном, ее используют простых НЧ фильтрах, реверберации, и других, не очень требовательных задач фильтрования. Однако, она оказывается очень полезна и при создании регуляторов исходный/обработанный сигнал, регуляторов баланса лево/право, простых ВЧ фильтров и т.д.

Формула:
Interp R, A, X, Y означает R=(1-X)*A+X*Y

Пример 1:

НЧ фильтр:

input in
output out
control filter=0.5

interp out, out, filter, in

end

ВЧ фильтр:

input in
output out
control filter=0.5
static st

interp st, st, filter, in
macsn out, in, st, 1

end

10. **skip**. Корректно перевести не смогу, т.к. сам плохо понял, как это работает.

ЗАДЕРЖКИ

Задержки – обязательная составляющая таких эффектов, как эхо, реверберация и др.

В Dane это реализуется указыванием количества циклов, в течение которых аудиоданные будут записываться во внутреннюю(на чипе) или внешнюю(RAM, оперативная память) память. Напомню, что 48000 циклов формируют 1 секунду аудиопотока. Соответственно, задержка в 48000 циклов сформирует задержку в 1 секунду.

Объявление задержки (выделение памяти под плагин):

itramsize 6000 ;(0.125 сек задержка) – внутренняя память
или
xtramsize 12000 ;(0.25 сек задержка) – внешняя память

Объявление tram регистров:

idelay write wr at 0 ;wr – просто имя. Начало записи в память (обычно в ноле)
idelay read rd at 12000 ;rd – просто имя. Конец записи в память (через 12000 циклов)
или
xdelay write...

Пример 1:

Простейшая статичная 0.25 сек задержка:

input in
output out

xtramsize 12000

xdelay write wr at 0

xdelay read rd at 12000

macs wr, in, 0, 0 ;←запись задержки производится со входа(регистр in)
macs out, rd, in, 1 ;←на выход подается входной сигнал, смешанный со считанным из сформированной задержки.

end

Пример 2:

Эхо эффект с демпфирующим фильтром:

input in
output out
control filter=0, feedback=0

xtramsize 12000

xdelay write wr at 0
xdelay read rd at 12000

macs out, in, rd, feedback
interp wr, out, filter, wr

end