Ｖ３ＭＭＬ説明書

for V3MML Version 1.28
最終更新日： 2025-12-13 [A]
著者：Akira SUZUKI

1.基本情報

Ｖ３ＭＭＬは、Ｍａｃ用のＭＭＬ演奏および、ＭＭＬ作成支援アプリです。

動作環境：
　　macOS 12.0 以降。
波形生成：
　　全てのトラックにおいて、サンプリングレート 384kHz、変位 64bit浮動小数点数にて生成。
波形再生：
　　macOS の AudioUnit の都合、全トラックの加算合成結果を、384kHz, 32bit浮動小数点数に変換して受け渡し、再生。
トラック数：
　　CPUパワーまたはメモリが許す限り（ＦＭ音源が最も処理が重いです）。
エンベロープ：
　　音量、フィルタ。
ＬＦＯ：
　　ピッチ、音量、パンポット、フィルタ、音源制御CMD(Y-control)。
エフェクタ：
　　ディレイ、双二次フィルタ。
音源モジュール：
　　全てのトラックにおいて、次の仮想音源を全て持っており、１つだけ選択して使用可能（選択変更は随時可能）。
　　　　サイン波
　　　　三角波
　　　　ノコギリ波
　　　　パルス波（DUTY比可変）
　　　　複合波（サイン波、三角波、ノコギリ波、パルス波）
　　　　波形メモリ音源
　　　　ＦＭ音源（OPMベース）
　　　　ホワイトノイズ
　　　　PSGノイズ
　　　　FCノイズ
　　　　GBノイズ
　　　　波形メモリノイズ
　　　　PCM音源（ADPCM/DPCMもPCMに展開して駆動）

1.1. 特徴

１つのトラック定義で、複数の内部チャネルを駆動する「ポリモード」を
Ｖ１ＭＭＬから踏襲しています。これにより、１つのトラック定義で和音を鳴らしたり、リリース音が重なるような制御ができます。

波形メモリ音源、サイン波、ノコギリ波、三角波、パルス波、複合波の各音源モジュールにおいては、複数のオペレータを持ち、現在の仕様では９重まで定義でき、単純コーラスや、オクターブ違いユニゾン、和音を音源側で構成することができます。（音源側の複数オペレータではパンポットを別にしたり、ノートオンの時間差は作れません）

ＦＭ音源モジュール以外の仮想音源のエンベロープは、最大2048個までのマルチポイントで節を設定可能で、ループも設定できます。（ループ機能は、トレモロ効果を想定）

仮想音源の音程管理を 64bit浮動小数点数化（細かいcentずれ指定が受け付け可能）。さらに、複数の音階スケールパターンにも対応。

ＰＣＭ音源は、2kHz～384kHzまでのサンプリング周波数に対応。

ＯＰＭベースＦＭ音源モジュールを採用していて、機能拡張は次の通り。
＊全てのオペレータにセルフフィードバックを設定可能。
＊マルチプルの 64bit浮動小数点数指定が可能。
＊フィードバックの 64bit浮動小数点数指定が可能。
＊DETUNE3 新設（64bit浮動小数点数の cent単位の周波数ずれ指定）。
＊DETUNE4 新設（64bit浮動小数点数の Hz単位の周波数ずれ指定）。
＊オペレータごとに、キーオン時のエンベロープ初期出力レベルの指定が可能。
　（従来のレベル引き継ぎモードに加え、強制的に初期レベルを指定可能）
＊オペレータごとに、キーオン時のサイン波の初期位相を指定可能。
　（従来の０スタートに加え、好きな位置からスタート可能）
＊サインテーブル及びエンベロープの解像度を向上。
　（FM音源特有のシュワシュワいうノイズを低減）
＊６オペモード（重厚化）と２オペモード（軽量化）も選択可能。
　（例えば６オペは旋律、２オペは和声、４オペはリズムに振るなど）

1.2. 基本的な使い方

基本的な使い方は、MMLウィンドウに演奏内容をテキスト入力し、ＰＬＡＹボタンをクリックするだけです。

ＭＭＬ(Music Macro Language)とは、プレーンテキストで書かれた音楽演奏用の言語です。いわばテキストで書かれた楽譜のようなものです。
ファイル読み込み時のプレーンテキストのエンコードは、
・UTF-8
・BOM無し
・改行コードはLFのみ
の形式を想定しています。

Ｖ３ＭＭＬのＭＭＬコマンドは、すべて半角文字で記述します。全角文字は、コメントエリア、もしくは曲名などのメタデータ内でのみ使用できます。

【サポートサイト】
https://mmlsound.sakura.ne.jp/

1.3. トラック定義

Ｖ３ＭＭＬでは、セミコロン「;」がトラック定義の区切り文字としての役割を果たしています。例えば、

```
ccc;
eee;
ggg;
```

このように書くと、トラックが３本定義され、ドミソの和音が３回発音されます。
テキスト先頭から最初のセミコロンまでが第１トラック、さらにその次のセミコロンまでが第２トラックといった具合です。この要領で、次々とトラックを増やすことが出来ます。
セミコロンは、厳密には次のトラックを開始する命令と解釈されるので、この場合、最後のセミコロンは無くても正常に演奏されます。しかし、後からトラック追加する時に備える意味で、トラックの終端には常にセミコロンを置いておくのが無難です。演奏情報が含まれない空のトラック定義は、演奏前に自動的に破棄されます。

1.4. ＭＭＬテキストの記述

ＭＭＬテキストの記述で定義する内容は、大きく分けると次の３種類です。

・ＭＭＬ定義（トラック定義群。実際に発音させる内容）
・マクロ定義
・メタデータ定義

再生要求を受けると、ＭＭＬテキストはコンパイルされてから演奏されます。
コンパイルされる内容のうち、時間ごとにシーケンスされるのは、ＭＭＬ定義の部分だけです。
ＭＭＬ定義以外の部分は、コンパイル前の処理（プリプロセッサ）により、波形データに変換されたり、文字置換されたりするため、直接のシーケンスデータにはなっていない点に注意してください。

・ＭＭＬ定義について

「cdefg;」というように、発音させたい内容を書いてトラック定義します。
ＭＭＬ定義内では、前もって定義したマクロを「$名前」で参照することも出来ます。ＭＭＬ定義内容をマクロ参照だけで満たしても構いません。

ひとつの発音トラックは、基本的に単音（和音にならない）フレーズを鳴らす内容になります。しかしながら、ひとつの発音トラックで和音を鳴らす、特別な定義方法もあります。

また、Ｖ３ＭＭＬでは、あらゆる記述で、大文字・小文字が区別されます。
Ｖ１ＭＭＬ、Ｖ２ＭＭＬでは、ＭＭＬコマンドで大文字・小文字の区別をしませんでしたが、Ｖ３ＭＭＬでは区別しますのでご注意ください。

・マクロ定義について

トラック先頭（ＭＭＬコマンドを１つも記述してない状態のトラック）で、
「$名前=内容;」という書き方をすると、マクロとして定義されます。
マクロの内容には改行を含んでも良いので、長いフレーズの定義も可能です。
また、トラック先頭であれば、複数のマクロを定義できます。
定義されたマクロは、それ以降に記述されたトラック内にて「$名前」で参照され、「内容」に置き換えられます。この置き換えは、プリプロセッサにより行われます。
すでに定義されたマクロを、以降のマクロ定義で参照するような、入れ子的なマクロ定義も可能です。

マクロ定義はセミコロンで締め括りますが、トラック定義とは別物と考えてください。
トラック先頭において「$」で始まる部分は、マクロ定義モードと解釈されます。
しかし、マクロ定義後に、１つでもＭＭＬコマンドを記述すると、トラック定義モードとなり、「$」は定義済みマクロの参照開始とみなされますので、ＭＭＬ記述中にマクロを定義することは出来ません。

・メタデータ定義について

メタデータとは、データのためのデータ、のことです。
ＭＭＬのモード設定や、波形データのユーザー定義など、付随情報全般をサポートする記述です。メタデータの定義は、行頭から「#」を付けたコマンドで行いますが、大きく２種類に分けられます。
（１）メタライン（#ML:）のグループ
（２）メタブロック（#MB:）のグループ
メタラインでは、改行を定義終端として認識しています。
メタブロックでは、中かっこ｛｝によってデータを括っていて、中かっこ閉じ「｝」を定義終端として認識するため、途中に改行が入ることが許されます。
メタライン、メタブロック共に、終端にセミコロンは不要です。
メタデータを記述する位置は、ＭＭＬ定義の前でも後でも大丈夫です。
これは、メタデータもマクロ同様、ＭＭＬが解釈される前にテキスト全体検索で情報取得後、すぐに除去されるため、ＭＭＬが解釈される時には無かったことにされているからです。
メタデータ定義がグループ化されている理由は、検索回数を最小限に抑え、コンパイル時間を短くするためです。

1.5. 数値の記述（計算式記述）

ＭＭＬコマンドやメタデータを記述する際、数値を指定する箇所が多数あります。Ｖ３ＭＭＬでは、一部の例外を除き、数値の指定を行う場所では計算式でも記述することが出来ます。

計算機能

（vコマンドの数値記述を計算式にした場合）
加算：v13+2　　→v15相当
減算：v15-2　　→v13相当
乗算：v5*3　　→v15相当
除算：v10/2　　→v5相当
冪乗：v2^3　　→v8相当
剰余：v13%2　　→v1相当
加算減算は同レベル、乗算除算剰余は同レベルで加算減算より優先、冪乗は乗算除算剰余より優先です。
冪乗が続く場合の計算方法は、エクセル同様の左結合です。
v2^3^2　　→v64相当
右結合で計算させたい場合はカッコを使います。
v"2^(3^2)"　　→v512相当

カッコの扱い

ＭＭＬコマンドでは、小カッコ「()」は、相対音量に使われているので、区別するために、数値指定の部分を "" で括った場合にのみ、小カッコ「()」を使うことが出来ます。
メタデータ定義における数値指定では、"" で括らずに小カッコ「()」を使用できます。

計算式記述が出来ない箇所

ＭＭＬコマンドにおける、音符の音長指定、休符長指定の部分には計算式記述が使えません。音長の加算は「&」、減算は「^」を使用します。

数値の表現

整数：12
浮動小数：3.51
１６進数：0x7ff
これらはいずれも正負(+-)記号をつけることが出来ます。
また、いずれも内部的には 64bit浮動小数点数で扱われます。記述できる有効な桁数は 64bit浮動小数点数に収まる範囲になります。
整数で数値を扱う場所は、最終的に整数に四捨五入されて利用されます。
１６進数の小数表現は対応していません。

「#」の記述

【記述例】

```
v13 cde v#-2 edc v#-2 ccc
```

cdeはv13、edcはv11、cccはv9で演奏されます。

【解説】
数値の代わりに「#」を記述すると、前回指定した値（前回が無ければ初期値）として解釈されます。（相対表現サポート）
多くのＭＭＬコマンドは、!sコマンドによるステータス表示の都合上、初期値や前回指定した値を記憶しています。計算式内でそれを利用することが出来ます。

メタデータ定義では、原則的に「#」は０として解釈されます。
ただし、
・波形メモリ音源用の波形データにおける、csv表現の箇所
・ＬＦＯテーブル定義における、テーブル値csv表現の箇所
・キーイベント定義（TRGとCODE）における、テーブル値csv表現の箇所
の場合は、「#」記述は前回値（初回の場合は０）として機能します。

【相対表現サポート状況】

コマンド	「#」の評価
abcdefg	使用不可。
k[1],[2]	使用不可。
key[1]	「#」は前回の値。初期値=-1
#MB:KEY_ASSIGN	「#」は常に０。
n[1],[2]	使用不可。
u[1],[2]	使用不可。
r	使用不可。

コマンド	「#」の評価
o[1]	「#」は前回の値。初期値=4
><	「#」は常に１。加えて数値の１文字目に「#」は使えない
ps[1]	「#」は前回の値。初期値=-1
#MB:PITCH_SCALE	「#」は常に０。
ns[1]	「#」は前回の値。初期値=0
nt[1]	「#」は前回の値。初期値=0
@d[1]	「#」は前回の値。初期値=0
l[1]	使用不可。
q[1],[2]	「#」は前回の値。[1]初期値=16　[2]初期値=16
qr[1]	「#」は前回の値。初期値=0
q%[1]	「#」は前回の値。初期値=0
@q[1]	「#」は前回の値。初期値=0
mv[1]	「#」は前回の値。初期値=-1
#MB:MIXING_VOL	「#」は常に０。
@p[1]	「#」は前回の値。初期値=0
p[1]	「#」は前回の値。初期値=0
vs[1]	「#」は前回の値。初期値=-1
#MB:VOLUME_SCALE	「#」は常に０。
v[1]	「#」は前回の値。初期値=13
)(	「#」は常に１。
vl[1]	「#」は前回の値。初期値=15
vf[1],[2]	「#」は常に０。
@fo[1],[2]	「#」は常に０。
@fi[1],[2]	「#」は常に０。
t[1]	「#」は前回の値。初期値=120
;	「#」利用箇所なし。
/* ... */	「#」利用箇所なし。
//	「#」利用箇所なし。
|:[1] ... :|	使用不可。
@rp	「#」利用箇所なし。
#MB:CONFIG	「#」は常に０。
#ML:INCLUDE	「#」利用箇所なし。

コマンド	「#」の評価
!n[1]	「#」は常に-1。
!r[1]	「#」は常に-1。
!s[str]	「#」利用箇所なし。
!c[1],[str]	「#」は常に０。
!m[1]	「#」は常に０。
!d[1],[2]	「#」は常に０。
![str]	「#」利用箇所なし。
!	「#」利用箇所なし。
@dsp[1],[2],[3]	[1]の「#」は常に０。 [2]と[3]の「#」は前回の値。[2]初期値=1　[3]初期値=1
$名前=内容;	マクロの名前部分には「#」利用箇所なし。 内容部分は最終的にMML解釈されるため記述内容に依存。
#ML:REPORT_TICKS	「#」利用箇所なし。
@@[str]	「#」利用箇所なし。
@[1]	「#」は前回の値。初期値=0 前回の値は、@@[str]による音源ごとに保存される。
@mh[1],[2],[3], [4],[5],[6],[7]	[1]...[7]全て：「#」は前回の値。初期値=0
@mha[1],[2],[3], [4]	[1]...[4]全て：「#」は前回の値。初期値=0
@mhz	「#」利用箇所なし。
@mr[1]	「#」は前回の値。初期値=-1
@mz[1]	「#」は前回の値。初期値=0
@mze[1]	「#」は前回の値。初期値=0
@ph[1],[2]	[1]の「#」は前回の値。初期値=0 [2]の「#」は前回の値。初期値=-1
@w[1],[2]	[1]の「#」は前回の値。初期値=4 [2]の「#」は前回の値。初期値=8
@n[1]	「#」は前回の値。初期値=1
@nr[1]	「#」は前回の値。初期値=1
y[str],[1]	「#」は常に０。
ym[str1],[str2], [1]	「#」は常に０。
yp[1]	「#」は前回の値。初期値=0
#MB:Y_PACK	「#」は常に０。

コマンド	「#」の評価
@dl[1]	「#」は前回の値。初期値=-1
#MB:DELAY	「#」は常に０。
@dlz	「#」利用箇所なし。
@fd[1]	「#」は前回の値。初期値=-1
#MB:FILTER_D	「#」は常に０。
@fdz	「#」利用箇所なし。
@fs[1]	「#」は前回の値。初期値=-1
#MB:FILTER_S	「#」は常に０。
@fsz	「#」利用箇所なし。
#MB:WVM_WAVE	波形csv表現の箇所：「#」は前回の値。初期値=0。 それ以外の箇所：「#」は０。
#MB:FMS_6OP	「#」は常に０。
#MB:FMS_4OP	「#」は常に０。
#MB:FMS_2OP	「#」は常に０。
#MB:FMS_OPM	「#」は常に０。
#MB:FMS_OPNA	「#」は常に０。
#MB:FMS_OPN	「#」は常に０。
#MB:LOAD_SAMPLING	「#」は常に０。
@eoc[str],[1]	「#」は常に０。
@eor[str],[1]	「#」は常に０。
@ea[1]	「#」は前回の値。初期値=-1
#MB:ENV_A	「#」は常に０。
@ef[1]	「#」は前回の値。初期値=-1
#MB:ENV_F	「#」は常に０。
@zr[str],[1]	「#」は常に-1。
z	「#」利用箇所なし。
@az[1]	「#」は前回の値。初期値=0
@loc[str],[1]	「#」は常に０。
@lor[str],[1]	「#」は常に０。
@lp[1]	「#」は前回の値。初期値=-1
@la[1]	「#」は前回の値。初期値=-1
@lb[1]	「#」は前回の値。初期値=-1

コマンド	「#」の評価
@lf[1]	「#」は前回の値。初期値=-1
@ly[1]	「#」は前回の値。初期値=-1
@l*z	「#」利用箇所なし。
@l*r	「#」利用箇所なし。
@l*t[1],[2]	[1]の「#」は０。 [2]の「#」は前回の値。 [1]が１のとき初期値=0、[1]が２のとき初期値=-1。
#MB:LFO_P	「#」は常に０。
#MB:LFO_A	「#」は常に０。
#MB:LFO_B	「#」は常に０。
#MB:LFO_F	「#」は常に０。
#MB:LFO_Y	「#」は常に０。
#MB:LFOTBL_ATK	テーブル値csv表現の箇所：「#」は前回の値。初期値=0。 それ以外の箇所：「#」は０。
#MB:LFOTBL_REL	テーブル値csv表現の箇所：「#」は前回の値。初期値=0。 それ以外の箇所：「#」は０。
@pl[1],[2]	[1]の「#」は前回の値。初期値=1 [2]の「#」は前回の値。初期値=0
xa[str],[1]	「#」は常に０。
xm[str],[1]	「#」は常に０。
@kt[1]	「#」は前回の値。初期値=-1
@kc[1]	「#」は前回の値。初期値=-1
@kl[1]	「#」は前回の値。初期値=-1
#MB:KEYEVENT_TRG	テーブル値csv表現の箇所：「#」は前回の値。初期値=0。 それ以外の箇所：「#」は０。
#MB:KEYEVENT_CODE	テーブル値csv表現の箇所：「#」は前回の値。初期値=0。 それ以外の箇所：「#」は０。
#MB:KEYEVENT_LEN	「#」は常に０。
#ML:TITLE	「#」利用箇所なし。
#ML:ARTIST	「#」利用箇所なし。
#ML:COMMENT	「#」利用箇所なし。
#ML:CODING	「#」利用箇所なし。

「?」の記述

【記述例】

```
@@"pls" @w0.005+?*0.495,1 cdef;
```

この場合、演奏の都度、「cdef」のデューティ比が、0.005～0.5 の範囲で、
ランダムに変わります。

```
@@"nzp" @n64 p1  @nr?1 ccc;
@@"nzp" @n64 p-1 @nr?1 ccc;
```

この場合、ランダム値パレットの１番でシフトレジスタ値を設定したノイズを、
別トラックで左右に振って鳴らしていますが、モノラルに聞こえます。
@nrコマンドが無い（シフトレジスタ値が違う）と、ステレオに聞こえます。

【解説】
「?」の記述は、0.0 ～ 1.0 の範囲のランダム値（浮動小数点数）として解釈されます。
「?数値」と記述した場合は、数値による番号のランダム値パレット配列から読み出した値として解釈されます。
ランダム値パレットとは、0.0 ～ 1.0 の範囲の浮動少数点数のランダム値を格納する配列です。配列の要素は 0番～99番の100個が用意されていて、ＭＭＬコンパイル開始時に、その都度、値が生成されます。
ランダム値パレット参照に、整数 0～99 以外の値を指定した場合は、エラーになります。

2.音程関連

2.1. cdefgab：音符の音名指定（ドレミファソラシ）

【記述例】

・８分音符の「ド＃」

```
c+8
```

・１６分音符の「ソ」のダブルシャープ

```
g++16
```

・付点１６分音符の「ラ♭」

```
a-16.
```

付点を１個つけた場合は音長が1.5倍、２個付けた場合は1.75倍になります。
A4.. と A4&8&16 は同じ長さです。

・音長96tickカウントの「ド」

```
c%96
```

・初期設定音長を使った記述

```
l4 c d8 e8 f g a8 b8
```

この場合「ド」「ファ」「ソ」の音長が４分音符になります。

【解説】

音名をアルファベットで指定し、ノートオンします。
トラック先頭での、初期設定のアルファベットと音名の対応は次の通りです。

【音名対応表】

音名	ド	レ	ミ	ファ	ソ	ラ	シ
対応記述	c	d	e	f	g	a	b

（オクターブは別途 oコマンド等で指定します）

音名の直後にプラス（+）をつけると、半音上がります（シャープ記述）。
音名の直後にマイナス（-）をつけると、半音下がります（フラット記述）。
ダブルシャープ（++）、ダブルフラット（--）の記述も受け付けます。

音名は、大文字（CDEFGAB）による記述でも小文字同様に機能します。

シャープやフラットの記述の後に数字をつけると、音長を指定できます。
音長の数値を省略すると、lコマンドによる初期設定音長が指定されたものと解釈します。

音長の後にピリオド（.）を付けると付点音符になります。
複数の付点の記述も受け付けます。

音長の延長（&）や、短縮（^）も可能です。
音長の延長や短縮をする場合、数値は省略できません。

別途指定する「#MB:KEY_ASSIGN」により、音名をどの音程に割り当てるか変更することが出来ます。

2.2. k[1],[2]：音符のキー番号指定

【記述例】
・８分音符のオクターブ４の「ド＃」

```
o4 k1+,8
```

・付点１６分音符のオクターブ４の「ラ♭」

```
o4 k6-,16.
```

付点を１個つけた場合は1.5倍、２個付けた場合は1.75倍の音長になります。

・２種類の音長表現

```
k5,4..
```

```
k5,4&8&16
```

上記２種類は同じ長さです。

・音長96tickカウントのオクターブ４の「ド」

```
o4 k1,%96
```

・初期設定音長を使った記述

```
l4 k1 k2,8 k3,8 k4 k5
```

この場合、k1、k4、k5 の音長が４分音符になります。

・ポルタメント記述

```
o4 k1_k6,4&k6,1
```

この場合、o4c から o4a まで４分音符でポルタメントし、そのあと全音符で o4a を維持します。

【解説】
キー番号を指定することで、ノートオンします。

指定する引数は次の通りです。
引数[1]...キー番号（keyNo）
引数[2]...音長（length）
引数はカンマで区切って指定します。カンマとlengthの組は省略できます。

引数[1]（keyNo）

発音したいキー番号を、数値で指定します。
初期設定のキー番号は次の通りです。

【デフォルト設定のキー番号表】

音名	c	d	e	f	g	a	b
キー番号	1	2	3	4	5	6	7

キー番号の直後にプラス（+）をつけると、半音上がります（シャープ記述）。
キー番号の直後にマイナス（-）をつけると、半音下がります（フラット記述）。
ダブルシャープ（++）、ダブルフラット（--）の記述も受け付けます。

引数[2]（length）

keyNo の後にカンマで区切って、発音したい音長を整数で指定します。
カンマとlengthのセットを省略した場合、lコマンドによる初期設定音長が指定されたものと解釈します。
音長の後にピリオド（.）を付けると付点音符になります。
複数の付点の記述も受け付けます。
音長の延長（&）や、短縮（^）も可能です。
音長の延長や短縮をする場合、数値は省略できません。

2.3. key[1]：音階スケール

2.4. #MB:KEY_ASSIGN：音階スケール定義

【記述例】

```
#MB:KEY_ASSIGN key=1 {
   c=0, d=2, e=4, f=5, g=7, a=9, b=11,
   k1=0, k2=2, k3=4, k4=5, k5=7, k6=9, k7=11,
}

l4 key1
cdefgab>c2<
k1 k2 k3 k4 k5 k6 k7 >k1,2<
;
```

このように書くと、

- #MB:KEY_ASSIGNにより、定義番号１番に音階パターンを定義。
  この場合、音名cdefgabにメジャースケール、キー番号k1...k7にも同様にメジャースケールのパターンを割り当てています。
- key1によって、定義番号１番の#MB:KEY_ASSIGNの内容を適用。

といった定義と適用を記述できます。

【解説】

音階スケールコマンド（key[1]）：
音階スケールコマンドでは、事前に定義した音階スケールを選んで適用します。
引数[1]には、音階スケール定義番号を、0 ～ 127 の整数で指定します。

音階スケール定義（#MB:KEY_ASSIGN）：
音階スケールコマンド（key[1]）で使用する、音階スケールパターンを定義します。

音階スケール定義方法の概要は、
・行頭からの記述で、キーワード「#MB:KEY_ASSIGN」を書く。
・続けて、スペースを置き、「key=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られたパターンデータを書く。
です。

パターンデータの開始は「 { 」で認識され、終了は「 } 」で検知されます。
パターンデータの各パラメータは、カンマ区切りで記述します。スペースや改行は読み飛ばされます。（最後のパラメータの後にカンマがあっても可）

#MB:KEY_ASSIGN の 定義番号：
定義番号は、keyコマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 127 です。

#MB:KEY_ASSIGN の パターンデータ：
パターンデータのフォーマットは、
・７種類の音名（cdefgab）に、割り当てたいオクターブ０の音程番号を定義（音名=音程番号）する。
・加えて、キー番号に、オクターブ０の音程番号を定義する。
キー番号は、少なくともk1...k7の７種類に割り当てが必要。
（最大k24までの２４種類に割り当て可能）

※オクターブ０の音程番号を割り当てる理由は、オクターブコマンドの影響を受けるため。オクターブ０以外の音程番号を割り当てることも出来る。

※割り当て方によって、メジャー、マイナーだけでなく、シンメトリカルスケールや、民族音階などを、cdefgabに対し自由に割り当て可能。

※キー番号を使えば、１０種類からなるブルーノートスケールも割り当てて、シャープやフラットなしで参照が可能。

2.5. n[1],[2]：音符の音程番号指定

【記述例】

・８分音符のオクターブ４の「ド＃」

```
n49,8
```

・付点１６分音符のオクターブ４の「ラ♭」

```
n56,16.
```

付点を１個つけた場合は1.5倍、２個付けた場合は1.75倍の音長になります。

・２種類の音長表現

```
n56,4..
```

```
n56,4&8&16
```

上記２種類は同じ長さです。

・音長96tickカウントのオクターブ４の「ド」

```
n48,%96
```

・初期設定音長を使った記述

```
l4 n48 n50,8 n52,8 n53 n55
```

この場合、n48、n53、n55 の音長が４分音符になります。

・ポルタメント記述

```
n48_n57,4&n57,1
```

この場合、o4c から o4a まで４分音符でポルタメントし、そのあと全音符で o4a を維持します。

【解説】

音程番号を指定することで、ノートオンします。

指定する引数は次の通りです。
引数[1]...音程番号（noteNo）
引数[2]...音長（length）
引数はカンマで区切って指定します。カンマとlengthの組は省略できます。

引数[1]（noteNo）

発音したい音程番号を、数値で指定します。
指定する数値は、音程番号表による音程番号です。

引数[2]（length）

noteNo の後にカンマで区切って、発音したい音長を整数で指定します。
カンマとlengthのセットを省略した場合、lコマンドによる初期設定音長が指定されたものと解釈します。
音長の後にピリオド（.）を付けると付点音符になります。
複数の付点の記述も受け付けます。
音長の延長（&）や、短縮（^）も可能です。
音長の延長や短縮をする場合、数値は省略できません。

2.6. 音程番号表

【解説】
ＶＳＭＭＬでは、音程に番号を振って、音程番号を管理しています。
オクターブ０の「c（ド）」の音を０番とし、半音上がるごとに番号が１増えることとしています。

初期設定では、オクターブ４の「a（ラ）」の音に440Hzの周波数を割り当てて、それを基準に他の音程には１２平均律に従った周波数を割り当てています。

【音程番号表】（octave/key）※o-2からo11までの抜粋

	c	c+	d	d+	e	f	f+	g	g+	a	a+	b
o-2	-24	-23	-22	-21	-20	-19	-18	-17	-16	-15	-14	-13
o-1	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
o0	0	1	2	3	4	5	6	7	8	9	10	11
o1	12	13	14	15	16	17	18	19	20	21	22	23
o2	24	25	26	27	28	29	30	31	32	33	34	35
o3	36	37	38	39	40	41	42	43	44	45	46	47
o4	48	49	50	51	52	53	54	55	56	57	58	59
o5	60	61	62	63	64	65	66	67	68	69	70	71
o6	72	73	74	75	76	77	78	79	80	81	82	83
o7	84	85	86	87	88	89	90	91	92	93	94	95
o8	96	97	98	99	100	101	102	103	104	105	106	107
o9	108	109	110	111	112	113	114	115	116	117	118	119
o10	120	121	122	123	124	125	126	127	128	129	130	131
o11	132	133	134	135	136	137	138	139	140	141	142	143

・KeyDispの表示では、音程番号 0 ～ 120 の範囲に応答します。
（0 未満は 0、120 より大きい場合は 120 として表示されます）

2.7. u[1],[2]：音符の周波数番号指定

【記述例】ピッチスケール設定が、type=12_TET,halftone_reso=100, の場合
・８分音符のオクターブ４の「ド＃」

```
u4900,8
```

・付点１６分音符のオクターブ４の「ラ♭」

```
u5600,16.
```

付点を１個つけた場合は1.5倍、２個付けた場合は1.75倍の音長になります。

・２種類の音長表現

```
u5600,4..
```

```
u5600,4&8&16
```

上記２種類は同じ長さです。

・音長96tickカウントのオクターブ４の「ド」

```
u4800,%96
```

・初期設定音長を使った記述

```
l4 u4800 u5000,8 u5200,8 u5300 u5500
```

この場合、u4800、u5300、u5500 の音長が４分音符になります。

・ポルタメント記述

```
u4800_u5700,4&u5700,1
```

この場合、o4c から o4a まで４分音符でポルタメントし、そのあと全音符で o4a を維持します。

【解説】
周波数番号を指定することで、ノートオンします。

指定する引数は次の通りです。
引数[1]...周波数番号（freqNo）
引数[2]...音長（length）
引数はカンマで区切って指定します。カンマとlengthの組は省略できます。

引数[1]（freqNo）
発音したい音程（周波数番号）を、数値で指定します。
指定する数値のパターンは、MMLで指定したピッチスケール（PSコマンド）の設定内容（type指定）によって変わります。

type=12_TET または
type=12_TET_trunc のとき（１２平均律）：
freqNo は、音程番号×halftone_reso で指定します。
この場合の freqNo は、小数以下の指定も受け付けます。
例えば、halftone_resoが 100 ならば、オクターブ４のラの４分音符の記述は
```
u5700,4
```
となります。

type=f_number のとき（Block/F-Number）：
freqNo は、14ビットの整数値で指定します。
14ビットのうち、上位３ビットはBlock（オクターブ相当）で、0 ～ 7 の範囲で指定できます。
下位11ビットは F-Number で、1～2047 の範囲で指定できます。F-Number には 0 は指定できないので注意してください。
したがって、この場合の freqNo は、14ビットの数値である 0 ～ 16383 のうち、下位11ビットが０にならない値が設定可能なものになります。
また、この場合の freqNo は、通常の10進数の整数指定のほか、接頭語に「0x」を付けることで16進数で記述することもできます。例えば、
```
u0x2410,4
```
となります。

type=divrate_octavelast または
type=divrate_octavefirst または
type=divrate_fullkey のとき（PSG分周比）：
freqNo は、０より大きい、31ビットの整数値で指定します（０は指定不可）。
「#MB:PITCHSCALE」で定義したPSG分周比のビット幅に合うよう指定してください。また、接頭語に「0x」を付けることで16進数で記述することもできます。
例えば、
```
u0x8d,4
```
となります。

type=steprate_mode0 または
type=steprate_mode1 または
type=steprate_mode2 のとき（WSGステップレート）：
freqNo は、０より大きい、31ビットの整数値で指定します（０は指定不可）。
「#MB:PITCHSCALE」で定義したWSGステップレートのビット幅に合うよう指定してください。また、接頭語に「0x」を付けることで16進数で記述することもできます。

引数[2]（length）
freqNo の後にカンマで区切って、発音したい音長を整数で指定します。
カンマとlengthのセットを省略した場合、lコマンドによる初期設定音長が指定されたものと解釈します。
音長の後にピリオド（.）を付けると付点音符になります。
複数の付点の記述も受け付けます。
音長の延長（&）や、短縮（^）も可能です。
音長の延長や短縮をする場合、数値は省略できません。

2.8. &：タイまたはスラー

【記述例】

・タイの記述

```
a4&16&8
```

この場合、ゲートタイムの設定は合計音長に対して作用します。
この記述方法は、休符でも使えます。
仕様上、lで初期設定音長を指定した際に省略できる音長記述は、音名表記直後の音長だけになります。

・スラーの記述

```
c4&d4&e4
```

cとd、dとeの間が切れ目なく（ゲートタイム一時無効で）発音されます。
この場合、e にのみ、指定のゲートタイムが作用します。
この記述方法は、休符には使えません。

```
a4&a16&a8
```

この記述はタイのように見えますが、スラーの記述と見なされます。
タイとの違いは、ゲートタイムが最後の「a8」にしか作用しない点です。

【解説】

タイは、同じ音の高さで音長を伸ばします。
スラーは、異なる音の高さで、切れ目なく（ゲートタイムも一時無効になりつつ）発音します。

2.9. ^：音長短縮記号（タイ拡張機能）

【記述例】

```
g64 a4^64
```

この場合、64分音符で装飾音gを発音してから、64分音符だけ短縮した４分音符のaを発音します。

```
g%2 a4^%2
```

この場合、２ticksで装飾音gを発音してから、２ticksだけ短縮した４分音符のaを発音します。

```
g%2 r4^%2
```

この場合、２ticksで装飾音gを発音してから、２ticksだけ短縮した４分休符を置きます。

【解説】

タイでは音長を延長しますが、音長短縮記号を使用して音長を繋いだ場合には、音長が短縮されます。
この機能は、ＭＭＬで装飾音を書く場合に、例えば64分音符を使って、あとから装飾の64分音符を短縮して帳尻を合わせる際に、短縮結果の音長を書く部分を簡単に書けるようにします。
音長短縮記号は、休符にも使用できます。

2.10.r：休符

【記述例】

```
l16 c4 r d8 r8 r.
```

この場合、「r」は１６分休符、「r.」は付点１６分休符になります。

```
c4 r2&8 c4
```

この場合、２分休符を８分休符延長しています。

```
c64 r8^64 c4
```

この場合、８分休符を６４分休符短縮しています。

```
c4 r8&r8 c4
```

この場合、エラーとなります。

【解説】

直前までノートオンされていればノートオフし、音符同様の長さ指定に従って休符が実行されます。
休符でも音符のように、長さの延長（&）や、短縮（^）が出来ます。
ただし、スラー（r4&r4）の記述はエラーとなります。

休符は、大文字（R）による記述でも小文字同様に機能します。

2.11. o[1]：オクターブ

【記述例】

```
o4 cdefg o5 edceg
```

【解説】

オクターブを整数値で指定します。

引数[1]には、オクターブ数を整数で指定します。
トラック先頭における初期設定は、o4 です。

オクターブ数そのものに制限は設けていませんが、
オクターブ指定を加味した音程に対応する、音程番号の範囲は、使用するピッチスケール応じて次のように制限されます。（音程番号とは）

ピッチスケール	音程番号の範囲
type=12_TET type=12_TET_trunc （12平均律）	-240 ～ +240
type=f_number （F-Number）	-24 ～ +24 のいずれかを開始音程番号として、 以降の96個の範囲
type=divrate_octavelast type=divrate_octavefirst type=divrate_fullkey type=steprate_mode0 type=steprate_mode1 type=steprate_mode2 （PSG/WSG）	0 ～ 120

（KeyDispの表示では、音程番号 0 ～ 120 の範囲に応答します）

【備考】
高いオクターブ設定では、再生サンプリング周波数の限界によって波形描画品質が低下する恐れがありますのでご注意ください。

2.12. > <：相対オクターブ

【記述例】

```
o4 ceg>dfa;
```

この場合、o4でcegを演奏し、次にo5でdfaを演奏します。

```
o3 ceg >3 dfa;
```

この場合、o3でcegを演奏し、次にo6でdfaを演奏します。

【解説】

当コマンド指定時の oコマンド の設定状態から、
相対的に数値変更したオクターブに設定します。
>　　オクターブ値を１上げます。（o4だった場合o5にする）
<　　オクターブ値を１下げます。（o4だった場合o3にする）

２以上変更したい場合は、複数個記述するか、数字を添えて記述します。
>3　 オクターブ値を３上げます。
<3　 オクターブ値を３下げます。

不等号の向きと上下の対応を反転したい場合は、
「#MB:CONFIG {...}」の「relative_sign」項目によって反転できます。

【例】

```
#MB:CONFIG { relative_sign: oct_updown=><: vol_updown=)(, }
```

この場合、「>」でオクターブUP、「<」でオクターブDOWNになります。

2.13. ps[1]：ピッチスケール

2.14. #MB:PITCH_SCALE：ピッチスケール定義

【記述例】

```
#MB:PITCH_SCALE ps=1 {
   type = 12_TET,         //12 tone equal temperament
   base_note = 57,
   base_freq = 440.0,
   halftone_reso = 100,
}

ps1 cdefg;
```

このように書くと、

- #MB:PITCH_SCALE により、定義番号１番にピッチスケール内容を定義。
  この場合、o4 a に 440Hzを割り当てて、それ以外を１２平均率で計算します。
- ps1によって、定義番号１番の #MB:PITCH_SCALE の内容を適用。

といった定義と適用を記述できます。

【解説】

ピッチスケールコマンド（ps[1]）：
音程名または音程番号から発音周波数を算出する方式を、選んで適用します。
トラック先頭での初期設定は、

```
type = 12_TET,
base_note = 57,
base_freq = 440.0,
halftone_reso = 100,
```

に設定されています。
引数[1]には、ピッチスケール定義番号を、０～２５５の整数で指定します。

ピッチスケール定義（#MB:PITCH_SCALE）：
ピッチスケールコマンド（ps[1]）で使用する、ピッチスケールパターンを定義します。

定義方法の概要は、
・行頭からの記述で、キーワード「#MB:PITCH_SCALE」を書く。
・続けて、スペースを置き、「ps=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られたパターンデータを書く。
です。

パターンデータの開始は「 { 」で認識され、終了は「 } 」で検知されます。
パターンデータの各パラメータは、カンマ区切りで記述します。スペースや改行は読み飛ばされます。（最後のパラメータの後にカンマがあっても可）

定義番号：
定義番号は、psコマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 255 です。

パターンデータ：
パターンデータのフォーマットは、まず最初のパラメータで指定する種別定義（type=...）で変わります。種別定義が必ず先頭で行われる必要があります。

・type=12_TET のとき（１２平均律）

【記述例】

```
#MB:PITCH_SCALE ps=1 {
   type = 12_TET,         //12 tone equal temperament
   base_note = 57,
   base_freq = 440.0,
   halftone_reso = 100,
}
```

この例では、オクターブ４の a に440Hzを割り当て、それを基準にした１２平均律を定義しています。

【解説】

フォーマットは次の通りです。

```
#MB:PITCH_SCALE ps=定義番号 {
     type=12_TET,
     base_note=[],
     base_freq=[],
     halftone_reso=[],
}
```

base_note：

基準になる音程番号を整数で指定します。（音程番号とは）
設定可能範囲は、0 ～ 119 です。

base_freq：

基準になる音程に割り当てる周波数を指定します。
小数以下の指定も受け付けます。
設定結果、o4a に割り当たる周波数が 195Hz ～ 988Hz の範囲外の場合は、エラーになります。

halftone_reso：

@dコマンドで指定する値を、半音を何分割した単位にするかを設定します。
100分割を指定した場合、@d1とすると１セント上げる設定になります。

・type=12_TET_trunc のとき（１２平均律）

【記述例】

```
#MB:PITCH_SCALE ps=2 {
   type = 12_TET_trunc,  //@dの少数以下を0方向へ丸める。
   base_note = 57,
   base_freq = 440,
   halftone_reso = 100,
}
```

【解説】

基本的に、type=12_TET のときとほぼ同様です。
違いは、

type=12_TET の場合、
@dコマンドで指定されるデチューン値の少数以下の丸め処理は行いません。

type=12_TET_trunc の場合、
@dコマンドで指定されるデチューン値の少数以下を、０方向へ丸める処理がなされます。

・type=f_number のとき（F-Number）

【記述例】

```
#MB:PITCH_SCALE ps=10 {
    type = f_number,
    master_clock = 3993600.0,
    prescale = 72.0,
    start_key = 0,
|   //f_number key00...key11
    0x026A,
    0x028F,
    0x02B6,
    0x02DF,
    0x030B,
    0x0339,
    0x036A,
    0x039E,
    0x03D5,
    0x0410,
    0x044E,
    0x048F
}
```

【解説】

フォーマットは次の通りです。

```
#MB:PITCH_SCALE ps=定義番号 {
      type=f_number,
      master_clock=[],
      prescale=[],
      start_key=[],
|
      KEY00, KEY01, KEY02, KEY03, KEY04, KEY05,
      KEY06, KEY07, KEY08, KEY09, KEY10, KEY11,
}
```

master_clock：

F-Number 計算のもとになるマスタークロックを指定します。小数以下の指定も受け付けます。例えば、3993600.0 などになります。

prescale：
F-Number 計算のもとになるプリスケール値を指定します。小数以下の指定も受け付けます。
これはサンプリング周波数を得るためのプリスケール値になるため、OPNのプリスケーラーそのものではなく、12倍済みの、例えば 72.0 などになります。

start_key：
KEY00～KEY11にて指定する F-Number値 は、Block（オクターブ）が０である前提です。その上で KEY00 に割り当てる音程番号を、-24 ～ 24 で指定します。
0 の時、KEY00 は オクターブ０の C と見なされます。
１増えるごとに半音上がり、１下がるごとに半音下がります。通常 0 を利用します。

```
KEY00：  start_key + 0半音 のF-Number
KEY01：  start_key + 1半音 のF-Number
KEY02：  start_key + 2半音 のF-Number
KEY03：  start_key + 3半音 のF-Number
KEY04：  start_key + 4半音 のF-Number
KEY05：  start_key + 5半音 のF-Number
KEY06：  start_key + 6半音 のF-Number
KEY07：  start_key + 7半音 のF-Number
KEY08：  start_key + 8半音 のF-Number
KEY09：  start_key + 9半音 のF-Number
KEY10：  start_key + 10半音 のF-Number
KEY11：  start_key + 11半音 のF-Number
```

※KEY00～KEY11の各F-Numberは整数１１ビットで指定します。ただし、０は使用できません。

※KEY00 から順に、数値は大きくなるように設定する必要があります。

※F-Numberの数値は10進数でも指定できますが、接頭辞「0x」をつけて16進数で記述することもできます。

【備考】周波数計算は次の要領で行われます。
fMUS = (F-Number*(2^(b-1))*fSAM)/(2^20)
fMUS：周波数（音程）
fSAM：再生サンプリング周波数。
（fSAM = master_clock / prescale）
b：オクターブ番号（0～7）

・type=f_number_t のとき（F-Number）

【記述例】

```
#MB:PITCH_SCALE ps=10 {
   type = f_number_t,
   master_clock = 4000000.0,
   prescale = 72.0,
   start_key = 0,
|  //f_number key00...key11,key12(limit)
   0x0269,
   0x028E,
   0x02B4,
   0x02DE,
   0x0309,
   0x0337,
   0x0368,
   0x039C,
   0x03D3,
   0x040E,
   0x044B,
   0x048D,
   0x04D1
}
```

【解説】

フォーマットは次の通りです。

```
#MB:PITCH_SCALE ps=定義番号 {
     type=f_number_t,
     master_clock=[],
     prescale=[],
     start_key=[],
|
     KEY00, KEY01, KEY02, KEY03, KEY04, KEY05,
     KEY06, KEY07, KEY08, KEY09, KEY10, KEY11, KEY12,
}
```

【備考】「type=f_number」との違い
「type=f_number_t」では、変化の大きいピッチＬＦＯを掛けた場合に、
内部のF-Number値へのピッチ加算結果が、KEY00～KEY12の値の範囲で円環状になることです（繰り上がり・繰り下がりでオクターブ変化）。
このことにより、オクターブを超える変化でも破綻せずに使用できます。
ただし、最低音程(o0c)を下回るピッチ加算は最低音程に制限され、最高音程(およそo8c)を上回るピッチ加算は最高音程に制限されます。

master_clock：
F-Number 計算のもとになるマスタークロックを指定します。小数以下の指定も受け付けます。例えば、4000000.0 などになります。

prescale：
F-Number 計算のもとになるプリスケール値を指定します。小数以下の指定も受け付けます。
これはサンプリング周波数を得るためのプリスケール値になるため、OPNのプリスケーラーそのものではなく、12倍済みの、例えば 72.0 などになります。

start_key：
KEY00～KEY12にて指定する F-Number値 は、Block（オクターブ）が０である前提です。その上で KEY00 に割り当てる音程番号を、-24 ～ 24 で指定します。
0 の時、KEY00 は オクターブ０の C と見なされます。
１増えるごとに半音上がり、１下がるごとに半音下がります。通常 0 を利用します。

KEY00： start_key + 0半音 のF-Number
KEY01： start_key + 1半音 のF-Number
KEY02： start_key + 2半音 のF-Number
KEY03： start_key + 3半音 のF-Number
KEY04： start_key + 4半音 のF-Number
KEY05： start_key + 5半音 のF-Number
KEY06： start_key + 6半音 のF-Number
KEY07： start_key + 7半音 のF-Number
KEY08： start_key + 8半音 のF-Number
KEY09： start_key + 9半音 のF-Number
KEY10： start_key + 10半音 のF-Number
KEY11： start_key + 11半音 のF-Number
KEY12： start_key + 12半音 のF-Number

※KEY00～KEY12の各F-Numberは整数１１ビットで指定します。ただし、０は使用できません。

※KEY00 から順に、数値は大きくなるように設定する必要があります。

※F-Numberの数値は10進数でも指定できますが、接頭辞「0x」をつけて16進数で記述することもできます。

【備考】周波数計算は次の要領で行われます。
fMUS = (F-Number*(2^(b-1))*fSAM)/(2^20)
fMUS：周波数（音程）
fSAM：再生サンプリング周波数。
（fSAM = master_clock / prescale）
b：オクターブ番号（0～7）

・type=divrate_octavelast のとき（PSG分周比）

【記述例】

```
#MB:PITCH_SCALE ps=11 {
   type = divrate_octavelast,
   master_clock = 3993600.0,
   prescale = 32.0,
   divrate_mask = 0x0FFF,
   divrate_offset = 0,
   start_key = 0,
|  //divrate KEY00...KEY11
   0x0EE8,
   0x0E12,
   0x0D48,
   0x0C89,
   0x0BD5,
   0x0B2B,
   0x0A8A,
   0x09F3,
   0x0964,
   0x08DD,
   0x085E,
   0x07E6
}
```

【解説】

フォーマットは次の通りです。

```
#MB:PITCH_SCALE ps=定義番号 {
     type=divrate_octavelast,
     master_clock=[],
     prescale=[],
     divrate_mask=[],
     divrate_offset=[],
     start_key=[],
|
     KEY00, KEY01, KEY02, KEY03, KEY04, KEY05,
     KEY06, KEY07, KEY08, KEY09, KEY10, KEY11,
}
```

「|」記号で区切る

master_clock：
PSG分周比計算のもとになるマスタークロックを指定します。小数以下の指定も受け付けます。例えば、3993600.0 などになります。

prescale：
PSG分周比計算のもとになるプリスケール値を指定します。小数以下の指定も受け付けます。
これはマスタークロックへのプリスケール値になるため、PSGのプリスケーラーそのものではなく、8倍済みの、例えば 32.0 などになります。

divrate_mask：
PSG分周比による周波数計算を行う場合に有効なビット幅を決定する、AND演算に使用するマスク値を整数で指定します。10進数でも指定できますが、接頭辞「0x」をつけて16進数で記述することもできます。
PSGでは通常、分周比は12ビットですので、0x0FFF のように指定します。
DCSGでは通常10ビットですので、0x03FF のように指定します。

divrate_offset：
PSG分周比に対する加算値（オフセット）を整数で指定します。通常は0を指定します。MSX-SCC など、オフセットが必要な場合などに利用します。

start_key：
KEY00～KEY11にて指定する PSG分周比 について、KEY00 の PSG分周比が、どの音程番号に相当するかを指定します。
設定範囲は 0 ～ 108 です。
0 の時、オクターブ0の C の分周比が定義されていると解釈します。１増えるごとに半音上がります。通常は 0 を利用します。

KEY00： start_key + 0半音のPSG分周比
KEY01： start_key + 1半音のPSG分周比
KEY02： start_key + 2半音のPSG分周比
KEY03： start_key + 3半音のPSG分周比
KEY04： start_key + 4半音のPSG分周比
KEY05： start_key + 5半音のPSG分周比
KEY06： start_key + 6半音のPSG分周比
KEY07： start_key + 7半音のPSG分周比
KEY08： start_key + 8半音のPSG分周比
KEY09： start_key + 9半音のPSG分周比
KEY10： start_key + 10半音のPSG分周比
KEY11： start_key + 11半音のPSG分周比

※KEY00～KEY11の各PSG分周比は整数３１ビット以下で指定します。ただし、０は使用できません。

※KEY00 から順に、数値は小さくなるように設定する必要があります。

※PSG分周比は10進数でも指定できますが、接頭辞「0x」をつけて16進数で記述することもできます。

【備考】周波数計算は次の要領で行われます。
fTONE = fm/(fp*(Tp+ofs))
fTONE：周波数（音程）
fm：master_clock
fp：prescale
Tp：PSG分周比
ofs：divrate_offset
(Tp+ofs）の部分は、divrate_maskでAND演算マスクがなされた結果の数値になります。

・type=divrate_octavefirst のとき（PSG分周比）

基本的に、type=divrate_octavelast のときと同様です。
違いは、

type=divrate_octavelast の場合、
オクターブ計算のシフト演算を、デチューン計算の後に行います。

type=divrate_octavefirst の場合、
オクターブ計算のシフト演算を、デチューン計算の前に行います。

・type=divrate_fullkey のとき（PSG分周比）

【記述例】

```
#MB:PITCH_SCALE ps=13 {
   type = divrate_fullkey,
   master_clock = 1789772.5,
   prescale = 16.0,
   divrate_mask = 0x0FFF,
   divrate_offset = 0,
|  //divrate key000...key119
   0x0FFE,0x0FFE,0x0FFE,0x0FFE,0x0FFE,0x0FFE,0x0FFE,0x0FFE,0x0FFE,0x0FD8,0x0EF7,0x0E20,
   0x0D5D,0x0C9C,0x0BE7,0x0B3C,0x0A9B,0x0A02,0x0973,0x08EB,0x086B,0x07F2,0x0780,0x0714,
   0x06AE,0x064E,0x05F4,0x059E,0x054D,0x0501,0x04B9,0x0475,0x0435,0x03F9,0x03C0,0x038A,
   0x0357,0x0327,0x02FA,0x02CF,0x02A7,0x0281,0x025D,0x023B,0x021B,0x01FC,0x01E0,0x01C5,
   0x01AC,0x0194,0x017D,0x0168,0x0153,0x0140,0x012E,0x011D,0x010D,0x00FE,0x00F0,0x00E2,
   0x00D6,0x00CA,0x00BE,0x00B4,0x00AA,0x00A0,0x0097,0x008F,0x0087,0x007F,0x0078,0x0071,
   0x006B,0x0065,0x005F,0x005A,0x0055,0x0050,0x004C,0x0047,0x0043,0x0040,0x003C,0x0039,
   0x0035,0x0032,0x0030,0x002D,0x002A,0x0028,0x0026,0x0024,0x0022,0x0020,0x001E,0x001C,
   0x001B,0x0019,0x0018,0x0016,0x0015,0x0014,0x0013,0x0012,0x0011,0x0010,0x000F,0x000E,
   0x0002,0x0002,0x0002,0x0002,0x0002,0x0002,0x0002,0x0002,0x0002,0x0002,0x0002,0x0002
}
```

【解説】

フォーマットは次の通りです。

```
#MB:PITCH_SCALE ps=定義番号 {
      type=divrate_fullkey,
      master_clock=[],
      prescale=[],
      divrate_mask=[],
      divrate_offset=[],
|
      K000,K001,K002,K003,K004,K005,K006,K007,K008,K009,K010,K011,
      K012,K013,K014,K015,K016,K017,K018,K019,K020,K021,K022,K023,
      K024,K025,K026,K027,K028,K029,K030,K031,K032,K033,K034,K035,
      K036,K037,K038,K039,K040,K041,K042,K043,K044,K045,K046,K047,
      K048,K049,K050,K051,K052,K053,K054,K055,K056,K057,K058,K059,
      K060,K061,K062,K063,K064,K065,K066,K067,K068,K069,K070,K071,
      K072,K073,K074,K075,K076,K077,K078,K079,K080,K081,K082,K083,
      K084,K085,K086,K087,K088,K089,K090,K091,K092,K093,K094,K095,
      K096,K097,K098,K099,K100,K101,K102,K103,K104,K105,K106,K107,
      K108,K109,K110,K111,K112,K113,K114,K115,K116,K117,K118,K119,
}
```

master_clock：

PSG分周比計算のもとになるマスタークロックを指定します。小数以下の指定も受け付けます。例えば、3993600.0 などになります。

prescale：

PSG分周比計算のもとになるプリスケール値を指定します。小数以下の指定も受け付けます。
これはマスタークロックへのプリスケール値になるため、PSGのプリスケーラーそのものではなく、8倍済みの、例えば 32.0 などになります。

divrate_mask：

PSG分周比による周波数計算を行う場合に有効なビット幅を決定する、AND演算に使用するマスク値を整数で指定します。10進数でも指定できますが、接頭辞「0x」をつけて16進数で記述することもできます。
PSGでは通常、分周比は12ビットですので、0x0FFF のように指定します。
DCSGでは通常10ビットですので、0x03FF のように指定します。

divrate_offset：

PSG分周比に対する加算値（オフセット）を整数で指定します。通常は0を指定します。

MSX-SCC など、オフセットが必要な場合などに利用します。

K000～K120：
K000：オクターブ０の c に割り当てるPSG分周比
K001：オクターブ０の c+ に割り当てるPSG分周比
K002：オクターブ０の d に割り当てるPSG分周比
：
K057：オクターブ４の a に割り当てるPSG分周比
：
K119：オクターブ９の b に割り当てるPSG分周比

※各PSG分周比は整数３１ビット以下で指定します。ただし、０は使用できません。

※PSG分周比は10進数でも指定できますが、接頭辞「0x」をつけて16進数で記述することもできます。

【備考】周波数計算は次の要領で行われます。
fTONE = fm/(fp*(Tp+ofs))
fTONE：周波数（音程）
fm：master_clock
fp：prescale
Tp：PSG分周比
ofs：divrate_offset
(Tp+ofs) の部分は、divrate_offsetでAND演算マスクがなされた結果の数値になります。

・type=steprate_mode0 のとき（WSGステップレート）

【記述例】

```
#MB:PITCH_SCALE ps=20 {
    type = steprate_mode0,
    master_rate = 24000.0,
    reg_bit_width = 20,
    reg_bit_mask = 0x0FFFFF,
    detune_width = 8,
    start_key = 96,
|   //steprate key00...key11
    0x02C578,
    0x02EFCB,
    0x031C82,
    0x034BC8,
    0x037DF6,
    0x03B335,
    0x03EB87,
    0x042717,
    0x046669,
    0x04A950,
    0x04F050,
    0x053B68
}
```

【解説】

フォーマットは次の通りです。

```
#MB:PITCH_SCALE ps=定義番号 {
      type=steprate_mode0,
      master_rate=[],
      reg_bit_width=[],
      reg_bit_mask=[],
      detune_width=[],
      start_key=[],
|
      KEY00,KEY01,KEY02,KEY03,KEY04,KEY05,
      KEY06,KEY07,KEY08,KEY09,KEY10,KEY11,
}
```

master_rate：
WSGステップレート周波数計算のもとになる再生周波数を指定します。小数以下の指定も受け付けます。例えば、24000.0 などになります。

reg_bit_width：
WSGステップレート周波数計算のもとになるビット幅を整数で指定します。例えば 20 などになります。

reg_bit_mask：
WSGステップレート周波数計算を行う場合に有効なビット幅を決定する、AND演算に使用するマスク値を整数で指定します。10進数でも指定できますが、接頭辞「0x」をつけて16進数で記述することもできます。WSGでは通常 20ビットですので、0x0FFFFF のように指定します。

detune_width：
WSGステップレートにおけるデチューン計算は、音程ごとのステップレート値の 1/n 倍の値を演算に使用しますが、このn値を 2の何乗にするかを指定します。通常 8 を指定します。

start_key：
KEY00～KEY11にて指定する WSGステップレート について、KEY00 の WSGステップレートが、どの音程番号に相当するかを、0 ～ 108 で指定します。
96 の時、オクターブ８の C のステップレートが定義されていると解釈します。１増えるごとに半音上がります。通常、96 を利用します。

KEY00： start_key + 0半音のステップレート
KEY01： start_key + 1半音のステップレート
KEY02： start_key + 2半音のステップレート
KEY03： start_key + 3半音のステップレート
KEY04： start_key + 4半音のステップレート
KEY05： start_key + 5半音のステップレート
KEY06： start_key + 6半音のステップレート
KEY07： start_key + 7半音のステップレート
KEY08： start_key + 8半音のステップレート
KEY09： start_key + 9半音のステップレート
KEY10： start_key + 10半音のステップレート
KEY11： start_key + 11半音のステップレート

※各WSGステップレートは整数３１ビット以下で指定します。ただし、０は使用できません。

※KEY00 から順に、数値は大きくなるように設定する必要があります。

※WSGステップレートは10進数でも指定できますが、接頭辞「0x」をつけて16進数で記述することもできます。

【備考】周波数計算は次の要領で行われます。
Freq = mRATE * wsgVAL / (2^regWIDTH)
Freq：周波数（音程）
mRATE：master_rate
wsgVAL：WSGステップレート
regWIDTH：reg_bit_width
wsgVAL の部分は、reg_bit_mask で AND演算マスクがなされた結果の数値になります。

- type=steprate_mode1 のとき（WSGステップレート）
- type=steprate_mode2 のとき（WSGステップレート）

基本的に、type=steprate_mode0 のときと同様です。
違いは、

type=steprate_mode0 の場合、
ピッチＬＦＯ演算を、通常モードで行います。

type=steprate_mode1 の場合、
ピッチＬＦＯ演算を、特殊モードで行います。
このモードのピッチＬＦＯは、ＬＦＯの form にユーザー定義テーブルしか使用できません。加えて、テーブルで指定される内容が、相対デチューンの値として解釈されます。

type=steprate_mode2 の場合、
type=steprate_mode1 の場合とほぼ同様ですが、ユーザー定義テーブルで定義する値のフォーマットが、符号あり8bit整数の４バイトパック方式になります。
例えば、0x0000FF03と定義したサンプルは、相対デチューンが0、0、-1、+3のコマンドとして処理されます。

2.15. ns[1]：ノートシフト

【記述例】

```
ns0 cde ns-1 cde ns-1 cde
```

このように書くと、二度目のドレミは半音下がります。
三度目のドレミも、二度目と同様、半音下がったドレミです。

```
ns0 cde ns#-1 cde ns#-1 cde ns0 cde
```

このように書くと、二度目のドレミは半音下がります。
三度目のドレミは、二度目よりさらに半音下がったドレミです。
四度目のドレミは、ずれのないドレミになります。

【解説】

ノートシフトを指定します。

引数[1]には、半音単位でキートランスポーズさせる数値を整数で指定します。
トラック先頭における初期設定は、ns0 です。

ノートシフト指定に範囲制限はありませんが、
ノートシフト指定を加味した音程番号の範囲は、
　-240 ～ +240
に制限されます。（音程番号とは）

【備考】
ノートシフトと、ノートトランスは同じ効果をもたらす機能ですが、互いに独立した変数を持つので、両方かけることが出来ます。
一方をキートランス用途、もう一方を和音記法における音程ずらし用途、のように使うことを想定しています。

2.16. nt[1]：ノートトランス

【記述例】

```
nt0 cde nt-1 cde nt-1 cde
```

このように書くと、二度目のドレミは半音下がります。
三度目のドレミも、二度目と同様、半音下がったドレミです。

```
nt0 cde nt#-1 cde nt#-1 cde nt0 cde
```

このように書くと、二度目のドレミは半音下がります。
三度目のドレミは、二度目よりさらに半音下がったドレミです。
四度目のドレミは、ずれのないドレミになります。

【解説】
ノートトランスを指定します。

引数[1]には、半音単位でキートランスポーズさせる数値を整数で指定します。
トラック先頭における初期設定は、nt0 です。

ノートトランス指定に範囲制限はありませんが、
ノートトランス指定を加味した音程番号の範囲は、
  -240 ～ +240
に制限されます。（音程番号とは）

【備考】
ノートシフトと、ノートトランスは同じ効果をもたらす機能ですが、互いに独立した変数を持つので、両方かけることが出来ます。
一方をキートランス用途、もう一方を和音記法における音程ずらし用途、のように使うことを想定しています。

2.17. @d[1]：デチューン

【記述例】

```
@d0 cdefg;
@d7.5 cdefg;
```

この場合、初期設定では１２平均率で、音程ずれの単位は１セントなので、7.5セントずれたユニゾンで cdefg が演奏されます。

【解説】

細かい音程のずれを与えます。
引数[1]で音程のずれを指定します。初期設定はゼロ（ずれ無し）です。
音程のずれ具合は、ピッチスケールの設定に依存します。

2.18. _：ポルタメント

【記述例】

```
c_>c4
```

ドからオクターブ上のドまで４分音符の長さで滑らかに変化させます。

```
c1&c_>c4&c2.
```

ドを全音符で鳴らした後、オクターブ上のドまで４分音符の長さで滑らかに変化させ、その後符点２分音符の長さで音程を維持します。

```
c_>c1&1&1&1
```

全音符以上の長さで変化をさせたい場合はタイ（音長の延長）で実現できます。
この例では、全音符４つ分の長さで、ドからオクターブ上のドまで滑らかに変化させています。

＜次のような記述は出来ません＞

```
c_<c4_>c4
```

この場合はエラーになります。

【解説】

ポルタメント（音程の滑らか変化）です。
音名を　_　で繋げるとポルタメントになります。
ピッチスケールが１２平均律モードの場合に使用できます。

3.音長関連

3.1. l[1]：数値省略時の音長

【記述例】

```
l8 cdefg
```

cdefgが全て８分音符で演奏されます。

【解説】

数値省略時の音長（音名の後の音長数値を省略した場合の音長）を設定します。
引数[1]には、通常１以上の整数(n)を指定します。
この場合、数値省略時の音長は、(n)分音符になります。
例えば l16 と記述すると、数値省略時の音長が１６分音符になります。
トラック先頭における初期設定は、l4（４分音符）です。

特例として、「%」付きの数値(%n)で記述することも出来ます。
この場合、数値省略時の音長は、(%n)のtickカウント数になります。
例えば l%96 と記述すると、数値省略時の音長は 96(ticks)になります。

(n)分音符に指定できる数値は、

```
#MB:CONFIG {
     tempo_unit: note_ticks=[]: beat_length=[],
}
```

の設定に影響を受けます。
例えば「note_ticks=384」の場合、tickカウント１あたり、384分音符になるので、数値省略時の音長として使えるラインナップは、384が整数で割り切れるものになります。列挙すると、
384, 192, 128, 96, 64, 48, 32, 24, 16, 12, 8, 6, 4, 3, 2, 1
となります。

3.2. q[1],[2]：ゲートタイム TYPE-1

【記述例】

```
l4 @q0 q%0 q32,32 cdefg
q27 gfedc
```

この場合、cdefg はゲートが掛からない「32/32 の発音」となり、
gfedc は「27/32 の発音 + 5/32 の休符」になります。

【注】
本機能と q%（ゲートタイムTYPE-2）は排他制御が掛かります。
q% の指定値が 0 以外の場合、q% が優先され、本機能は無効になります。
ただし、本機能の指定値は保存されているので、q% に 0 を指定して無効化すると、保存されていた指定値で本機能が有効になります。
本機能と @q（ゲートタイムTYPE-3）は排他制御が掛からず、同時に作用します。

【解説】
ゲートタイムの割合を設定します。
ある長さの音符を演奏するとき、実際に指定された音長のうち、何分の何を発音し、残りを休符とするか、を決定します。
スタッカートなどの表現に利用します。

指定する引数は次の通りです。
引数[1]...発音長倍率の分子（num）
引数[2]...発音長倍率の分母（denom）
引数はカンマで区切って指定します。カンマとdenomの組は省略できます。
トラック先頭における初期設定は、q16,16 です。

引数[1]（num）
発音長倍率の分子を、符号無しの整数で指定します。

引数[2]（denom）
発音長倍率の分母を、符号無しの整数で指定します。
denom の初期設定は 16 です。
denom は省略可能で、省略した場合は以前に設定された値が採用されます。
ゲートタイムの設定範囲は、
分子÷分母が 0より大きく、1以下となる範囲です。
範囲外の場合は、設定そのものが無視されます。

トラック先頭で１度 denom 付きで設定したら、以降の設定は denom を省略する使い方を想定しています。
分母が 16 で構わない場合は、トラック先頭から denom を省略してください。

3.3. q%[1]：ゲートタイム TYPE-2

【記述例】

```
#MB:CONFIG { tempo_unit: note_ticks=192: beat_length=4, }
@q0 q%40 c4 g2;
```

この場合、
c は 40ticks発音＋8ticks休符、
g は 40ticks発音＋56ticks休符、
として演奏されます。

【解説】

ゲートタイムを、「ノートオンからの tickカウント数」で設定します。
ある長さの音符を演奏するとき、実際に指定された音長のうち、何tickカウントを発音して残りを休符とするか、を決定します。
スタッカートなどの表現に利用します。
引数[1]には、発音する側のtickカウント数を 0 以上の整数で指定します。
0 は本機能を無効にする特別な指定値です。
トラック先頭における初期設定は、q%0 です。

tickカウントあたりの発音長は、

```
#MB:CONFIG {
     tempo_unit: note_ticks=[]: beat_length=[],
}
```

の設定の影響を受けます。

【注】

本機能と q（ゲートタイムTYPE-1）は排他制御が掛かります。
q を使いたい場合、本機能を q%0 としてください。
本機能と @q（ゲートタイムTYPE-3）は排他制御が掛からず、同時に作用します。

3.4. @q[1]：ゲートタイム TYPE-3

【記述例】

```
#MB:CONFIG { tempo_unit: note_ticks=192: beat_length=4, }
q%0 q16,16 @q5 c4 g2;
```

この場合、
c は 43ticks発音＋5ticks休符、
g は 91ticks発音＋5ticks休符、
として演奏されます。

【解説】

ゲートタイムを、「ノートオフを早める tickカウント数」で指定します。
q または q% で指定されたゲートタイムから、さらに、
指定数値の tickカウント数 だけ引いた発音長にします。
引数[1]には、ノートオフを早める tickカウント数を、
0 以上の整数で指定します。
トラック先頭における初期設定は、@q0 です。

tickカウントあたりの発音長は、

```
#MB:CONFIG {
     tempo_unit: note_ticks=[]: beat_length=[],
}
```

の設定の影響を受けます。

【注】

q または q% と、@q により、
計算上の発音長が１tickカウント未満になってしまう場合では、
実際の発音長は１tickカウントに矯正されます。

3.5. qr[1]：ゲートTYPE-1 の丸めモード

【記述例】

```
#MB:CONFIG { tempo_unit: note_ticks=192: beat_length=4, }
t15 @q0 q1,15 qr0 c4 qr1 d4 qr2 e4;
```

この場合、
c4 は 3ticks発音＋45ticks休符（四捨五入）、
d4 は 3ticks発音＋45ticks休符（切り捨て）、
e4 は 4ticks発音＋44ticks休符（切り上げ）、
として演奏されます。

【解説】

ゲートタイム TYPE-1 によって生じる少数以下のtickカウント数につき、整数に丸めるモードを設定します。
引数[1]には、モード番号を、0, 1, 2 のいずれかで指定します。
0 の場合、四捨五入で整数に丸めます。
1 の場合、切り捨てで整数に丸めます。
2 の場合、切り上げで整数に丸めます。
トラック先頭における初期設定は、qr0 です。

3.6. qtp[1]：ゲートのtickカウント群の指定

【記述例】

```
#MB:CONFIG { tempo_unit: note_ticks=192*8: beat_length=4, }
t100 qtp8 @q1 q16,16 cdefg;    //gate tick pack=8
```

この場合、@q1と指定されていますが、qtp8により８倍されて@qが掛かります。

【解説】

qtp[]は、ゲート設定のtickカウント群をいくつにするか設定します。
引数[1]には、１以上の整数を指定します。
トラック先頭における初期設定は、qtp1 です。

qtp[]は、メタ設定
#MB:CONFIG { tempo_unit: note_ticks=[]: beat_length=[], }
における、note_ticks の指定で、例えば note_ticks=192 で曲を記述したとして、後から note_ticks=192*8 などとしてtickカウントの解像度を上げた場合を想定したコマンドです。
tickカウントの解像度を上げた場合、記述済みの @q[], q%[], @az[] などは、解像度を上げた倍率だけ、同様に倍率を上げなくてはなりません。
しかし、qtp[]で倍率を指定しておけば、記述済みのゲート設定を書き直す必要が無くなります。

・q%[]指定値は、それ以前に記述されたqtp[]で指定された数値倍されます。
・@q[]指定値は、それ以前に記述されたqtp[]で指定された数値倍されます。
・@az[]指定値は、それ以前に記述されたqtp[]で指定された数値倍されます。

・q[],[]では、対象音符のtickカウント数をqtp[]設定値で割って、
ゲートTYPE-1計算し(丸め含む)、qtp[]設定値を掛けたtickカウント数で音長が計算されます。

4.音量関連

4.1. 音量設定の前提事項

【解説】
Ｖ３ＭＭＬでは、音声バッファを -1.0 ～ +1.0 の浮動小数点数（ステレオ）で用意しています。
音声バッファにレンダリングする前の、音源モジュール段階の波形も、
-1.0 ～ +1.0 の浮動小数点数（モノラル）で用意しています。
このため、レベル調整しないと複数トラックであっという間に音割れが発生してしまいます。

レベル調整は大まかに分けて５つの段階があります。
音源モジュールの出力を音声バッファにレンダリングする際は、次の順でレベル調整を行います。
（１）コンフィグ設定：#MB:CONFIG { mixing_vol: master_rate=[], }
（２）ミキシング設定：#MB:MIXING_VOL mv=[] { [] } および mv[]。
（３）音量コマンドのv、エンベロープの@ea（最大で振幅が 1.0 倍に設定）。
（４）音量コマンドのvl、音量ＬＦＯの@la（最大で振幅が 1.0 倍に設定）。
（５）パンポット演算（@p および p）。
となっています。

（１）の設定をしなかった場合の初期設定は0.5倍です。
（２）の設定をしなかった場合の初期設定は 0dB（1.0倍）です。

@p のパンポット演算では注意が必要です。倍率計算を抜粋して挙げると、

@p設定	左側の振幅倍率	右側の振幅倍率
@p-100	1.0 倍	0.0 倍
@p-50	0.75 倍	0.25 倍
@p0	0.5 倍	0.5 倍
@p50	0.25 倍	0.75 倍
@p100	0.0 倍	1.0 倍

となります。@p では、左右の最大振幅が、中央設定の２倍になっています。

一方 p のレガシーパンポットは、中央設定より大きくはなりません。

p設定	左側の振幅倍率	右側の振幅倍率
p-1	1.0 倍	0.0 倍
p0	1.0 倍	1.0 倍
p1	0.0 倍	1.0 倍

したがって、
・ミキシングボリューム関係（mixing_vol/mv[]）を何も設定しない
・v[]、@ea[]、vl[]、@la[]による設定が最大振幅になっている
・パンポットが中心
という条件では、左右双方、音源モジュール出力の振幅の 0.25 倍になります。
この条件のとき、同じ波形、同じ音程、同じ位相でトラックを重ねると、
　　４トラックまでは音割れせず、
　　５トラック以上では音割れする
ことになります。

以上を踏まえて、最終的に、全トラックの音声を加算した結果が、
-1.0 ～ +1.0 の振幅に収まるように、
・#MB:CONFIG { mixing_vol: master_rate=[], }
・#MB:MIXING_VOL mv=[] { [] } および mv[]
・v[]、@ea[]
・vl[]、@la[]
・@p[]、p[]
における倍率を設定していくことになります。

意図せず音割れを起こしてしまった場合に備えて、KeyDispウィンドウには
　　peakLevel(L)
　　peakLevel(R)
の表示を設けてあります。
この表示は、音声出力の最大が何%であったかを示しています。
音割れでクリッピングが発生すると、表示が「over(数値)」となり、数値のサンプル数だけクリッピングを行ったことが分かります。
音割れが発生した場合は、ミキシングボリュームを下げたり等の調整を検討してください。

【備考】
音量計算は統一的に 64bit 浮動小数計算で行っています。そのため、各種音量計算による音質ロスの心配は、ほぼありません。

4.2. デシベル計算

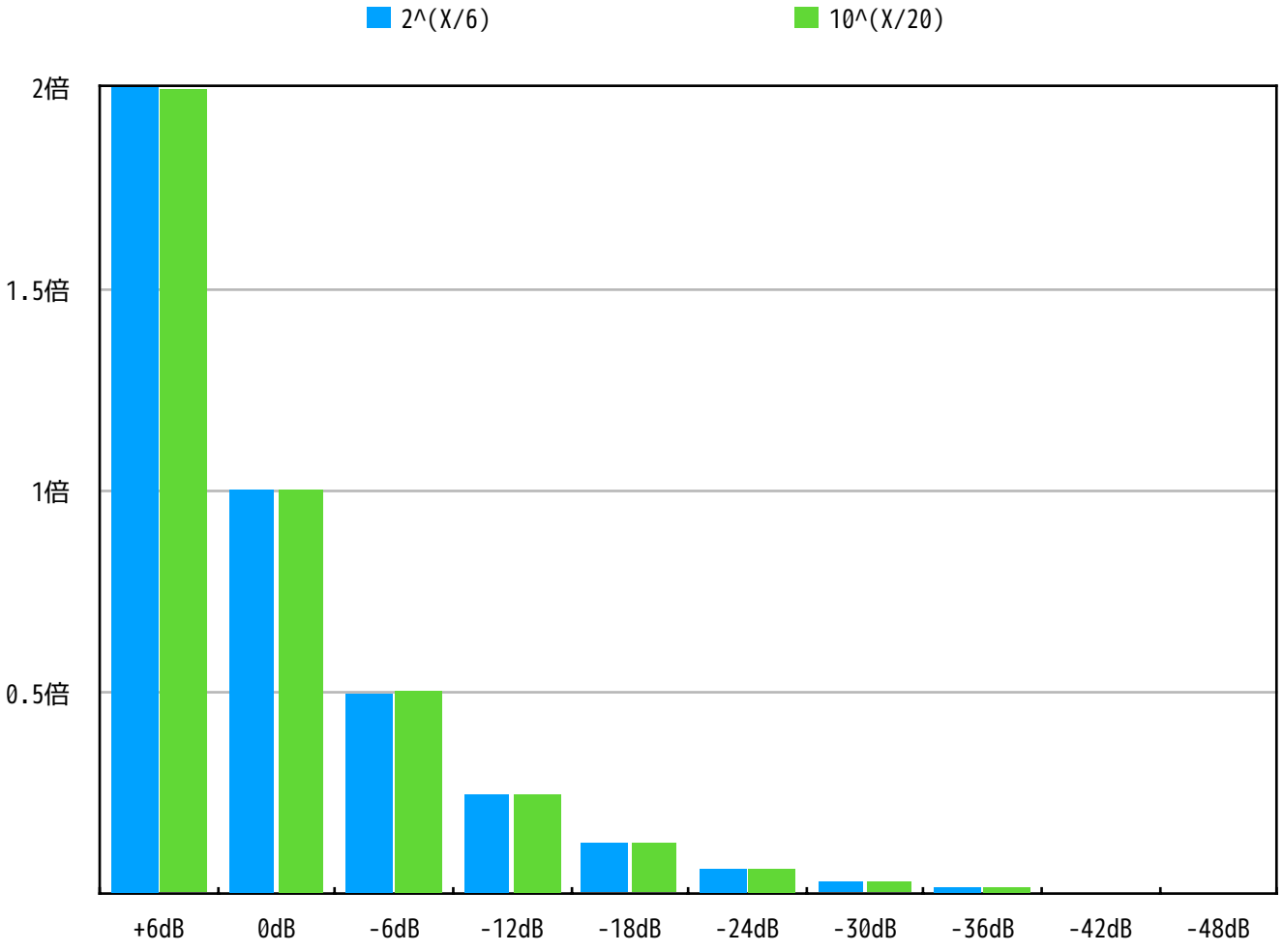
【解説】
ＶＳＭＭＬで扱うデシベル（dB）は、相対量としてのデシベルで、基準量に対する倍率を表すものです。
通常、デシベル計算による音量の振幅倍率は次の式で得られます。

$$倍率＝10^{(dB/20)}$$

また、ＶＳＭＭＬでは特別に、次の式も使用できるようになっています。

$$倍率＝ 2^{(dBB/6)}$$

この式は２を底にしていることから、便宜上、デシベルバイナリモード（dBB）と呼ぶことにします。
また、この計算は通常のデシベル計算とよく似た結果になりますが、相違点は、
・dB は、20増えるごとに丁度10倍、20減るごとに丁度1/10倍。
・dBBは、 6増えるごとに丁度 2倍、 6減るごとに丁度 1/2倍。
となります。
これをグラフと表にしたものは次の通りです。



dBB	倍率：2^(dBB/6)	dB	倍率：10^(dB/20)
24	16	24	15.8489319246111
18	8	18	7.94328234724282
12	4	12	3.98107170553497
6	2	6	1.99526231496888
0	1	0	1
-6	0.5	-6	0.501187233627272
-12	0.25	-12	0.251188643150958
-18	0.125	-18	0.125892541179417
-24	0.0625	-24	0.0630957344480193
-30	0.03125	-30	0.0316227766016838
-36	0.015625	-36	0.0158489319246111
-42	0.0078125	-42	0.00794328234724281
-48	0.00390625	-48	0.00398107170553497
-54	0.001953125	-54	0.00199526231496888
-60	0.0009765625	-60	0.001
-66	0.00048828125	-66	0.000501187233627273
-72	0.000244140625	-72	0.000251188643150958
-78	0.0001220703125	-78	0.000125892541179417
-84	0.00006103515625	-84	0.0000630957344480193
-90	0.000030517578125	-90	0.0000316227766016838
-96	0.0000152587890625	-96	0.0000158489319246111

4.3. mv[1]：ミキシングボリューム

4.4. #MB:MIXING_VOL：ミキシングボリューム定義

【記述例】

```
#MB:MIXING_VOL mv=1 { -3dB }
mv1 l4 cegec;
```

このように書くと、

- #MB:MIXING_VOL によって、定義番号１番にミキシングボリューム内容を割り当てています。
- mv1によって、定義番号１番の #MB:MIXING_VOL の内容を適用。

といった定義と適用を記述できます。

【解説】

ミキシングボリュームコマンド（mv[1]）：
ミキシングボリューム定義（#MB:MIXING_VOL）の定義番号を整数で指定します。
定義番号は 0 ～ 1023 の整数です。
未定義の番号を指定するとエラーになります。
トラック先頭における初期設定には定義番号は無く、0dBの音量設定です。

ミキシングボリューム定義（#MB:MIXING_VOL）：
ミキシングボリュームコマンド（mv[1]）で使用する、ミキシングボリュームのレベル値を定義します。

ミキシングボリューム定義方法の概要は、
・行頭からの記述で、キーワード「#MB:MIXING_VOL」を書く。
・続けて、スペースを置き、「mv=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られたレベル値を書く。
です。

定義番号：
定義番号は、mvコマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 1023 です。

レベル値：
数値（計算式）のみを書いた場合、振幅への倍数として扱われます。この場合、０より大きい値である必要があります。
数値の末尾に「dB」と記述した場合、数値がデシベル値として扱われます。
数値の末尾に「dBB」と記述した場合、数値がデシベルバイナリモード値として扱われます。
dB値 または dBB値の場合、数値が正の場合は増幅になり、負の場合は減衰になります。

【備考】
音量設定の前提事項

4.5. @p[1]：パンポット

【記述例】

```
@p0 ceg @p100 dfa @p-100 egb;
```

【解説】

パンポット（音声出力の左右バランス）を設定します。

引数[1]には、左右バランスを表す数値を指定します。
設定範囲は -100 ～ 100 で、小数以下の指定も受け付けます。
トラック先頭における初期設定は、@p0（中心）です。

設定に対応する左右のバランスは次の通りです。

@p設定	左側の振幅倍率	右側の振幅倍率
@p-100（左側に最大）	1.0 倍	0.0 倍
@p-50（左側に50%）	0.75 倍	0.25 倍
@p0（中央設定）	0.5 倍	0.5 倍
@p50（右側に50%）	0.25 倍	0.75 倍
@p100（右側に最大）	0.0 倍	1.0 倍

（@pでは、左右の最大振幅が、中央設定の２倍になっている点に注意）

【備考】

音量設定の前提事項

4.6. p[1]：レガシーパンポット

【記述例】

```
p0 ceg p1 dfa p-1 egb;
```

【解説】
レガシーパンポット（左右の音声出力のスイッチング）を設定します。

引数[1]には、左右のスイッチングを表す整数を指定します。
設定範囲は、-1、0、1 の三択です。
トラック先頭における初期設定は、p0（左右両方出力）です。

設定に対応する左右のスイッチングは次の通りです。

P設定	左側の振幅倍率	右側の振幅倍率
p-1（左側のみ出力）	1.0 倍	0.0 倍
p0（左右両方出力）	1.0 倍	1.0 倍
p1（右側のみ出力）	0.0 倍	1.0 倍

【備考】
レガシーパンポットは、機械的に左右のオンオフを行うだけです。
@pのパンポットを中心に設定した状態で使用することを想定しています。

※@pコマンドでは片方に完全に寄せた場合、中心と比較して振幅が倍になりますが、pコマンドでは左右のオンオフを行うだけなので、左右に振っても中心の振幅を超えることはありません。

【備考２】
音量設定の前提事項

4.7. vs[1]：ボリュームスケール

4.8. #MB:VOLUME_SCALE：ボリュームスケール定義

【記述例】

```
#MB:VOLUME_SCALE vs=1 {
   type=linear, v=15, @ea=mul, vl=15, @la=mul,
}
t70 l4 vs1 v15 a(3 a(3 a(3 a(3 a(3 a;
```

このように書くと、

- #MB:VOLUME_SCALE により、定義番号１番にボリュームスケール内容を定義。
- vs1によって、定義番号１番の #MB:VOLUME_SCALE の内容を適用。

といった定義と適用を記述できます。

【解説】

ボリュームスケールコマンド（vs[1]）：
定義済みのボリュームスケールを選んで適用します。
トラック先頭での初期設定は、

```
     type=linear, v=15, @ea=mul, vl=15, @la=mul,
```

に設定されています。
引数[1]には、ボリュームスケール定義番号を、０～２５５の整数で指定します。

ボリュームスケール定義（#MB:VOLUME_SCALE）：
ボリュームスケールコマンド（vs[1]）で使用する、ボリュームスケールパターンを定義します。

定義方法の概要は、
・行頭からの記述で、キーワード「#MB:VOLUME_SCALE」を書く。
・続けて、スペースを置き、「vs=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られたパターンデータを書く。
です。

パターンデータの開始は「 { 」で認識され、終了は「 } 」で検知されます。
パターンデータの各パラメータは、カンマ区切りで記述します。スペースや改行は読み飛ばされます。（最後のパラメータの後にカンマがあっても可）

定義番号：
定義番号は、vsコマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 255 です。

パターンデータ：
パターンデータのフォーマットは、まず最初のパラメータで指定する種別定義（type=...）で変わります。種別定義が必ず先頭で行われる必要があります。

・type=table のとき（配列定義モード）

【記述例】

浮動小数点数による配列定義

```
#MB:VOLUME_SCALE vs=1 {
   type=table, vh=15, v=15, @ea=mul, vr=0, vl=15, @la=mul, vlr=0, |
   0.00, 0.07, 0.13, 0.20, 0.27, 0.33, 0.40, 0.47,
   0.53, 0.60, 0.67, 0.73, 0.80, 0.87, 0.93, 1.00,
}
vs1 v15 c(c(c(c(c(c(c(c(c(c(c(c(c(c(c(c;
```

デシベルによる配列定義

```
#MB:VOLUME_SCALE vs=2 {
   type=table, vh=15, v=15, @ea=mul, vr=2, vl=15, @la=mul, vlr=2, |
     0.0,-42dB,-39dB,-36dB,-33dB,-30dB,-27dB,-24dB,
   -21dB,-18dB,-15dB,-12dB, -9dB, -6dB, -3dB,  0dB,
}
vs2 v15 c(c(c(c(c(c(c(c(c(c(c(c(c(c(c(c;
```

この例では、v0、vl0の部分のみ、０倍となるよう、浮動小数点数で定義しています。

【解説】

フォーマットは次の通りです。

```
#MB:VOLUME_SCALE vs=定義番号 {
     type=table, vh=[], v=[], @ea=[], vr=[],
                         vl=[], @la=[], vlr=[], |
     V0_MAGNI,             //V0_MAGNIからVMAX_MAGNIまでの
     V1_MAGNI,             //音量倍率の配列個数は、
     V2_MAGNI,             //vh=[]で指定する数値＋１の個数を
           :               //定義します。
     VMAX_MAGNI,
}
```

「|」記号で区切る

vh：
仮想音量ハードの音量最大値を指定します。
設定範囲は、4 ～ 1024 の整数です。
vhの指定により、音量倍率値の配列定義を、vh＋１個の要素で行う必要があります。
例えば、vh=15とした場合、v0～v15に相当する１６個の音量倍率値を配列に定義しなければなりません。

v：
vコマンドで指定する最大値を指定します。
設定範囲は 4 ～ 10000 の整数です。

@ea：
vコマンドに対応する音量エンベロープ（@ea）の計算方法の名称を指定します。
設定できる名称は、mul または add です。
　　mulの場合は、@eaエンベロープは v 設定値への乗算
　　addの場合は、@eaエンベロープは v 設定値への加算
となります。
※@ea=addでは、v と @ea の分母を vh の分母に合わせて、@eaのカレント値（最大レベルから減った分）を v 設定値に加算します。

vr：
vコマンドと、@eaエンベロープ出力を演算した結果(＊ａ)を、vhのインデックス番号（整数）に丸めるモードを指定します。
０のとき：
　(＊ａ)の小数以下を四捨五入します。
１のとき：
　(＊ａ)の小数以下を切り上げます。
２のとき：
　(＊ａ)に１を足し小数以下を切り捨てます。
　切り捨て結果が v 設定値を超える場合は v 設定値に制限します。
３のとき：
　(＊ａ)の小数以下を切り捨てます。

vl：
vlコマンドで指定する最大値を指定します。
設定範囲は 4 ～ 10000 の整数です。

@la：
vlコマンドに対応する音量ＬＦＯ（@la）の計算方法の名称を指定します。
設定できる名称は、mul または add です。
　　mulの場合は、@la音量ＬＦＯは vl 設定値への乗算
　　addの場合は、@la音量ＬＦＯは vl 設定値への加算
となります。
※@la=addでは、vl と @la の分母を vh の分母に合わせて、@laのカレント値を vl 設定値に加算します。
（@laのカレント値は、０以下の数値になる想定です。０の時が vl の音量そのもので、負数になる程に小さい音量になります。正数の場合は０に矯正されます）

vlr：
vlコマンドと、@la音量ＬＦＯ出力を演算した結果(＊ｂ)を、
整数に丸めるモードを指定します。
０のとき：
　(＊ｂ)の小数以下を四捨五入します。
１のとき：
　(＊ｂ)の小数以下を切り上げます。
２のとき：
　(＊ｂ)に１を足し小数以下を切り捨てます。
　切り捨て結果が vl 設定値を超える場合は vl 設定値に制限します。
３のとき：
　(＊ｂ)の小数以下を切り捨てます。

V0_MAGNI～VMAX_MAGNI：
仮想音量ハードの、インデックス 0～最大値 で使用する倍率値を指定します。
V0_MAGNIからVMAX_MAGNIまでの定義個数は、
　　vh値 + 1(個)
になります。
指定可能な倍率値の範囲は、０～１の浮動小数点数です。
特別な表記方法として、
・デシベルによる指定。※末尾に「dB」を付加する
・デシベルバイナリモードによる指定。※末尾に「dBB」を付加する
が可能です。

v/mul/vr0～3：　　(@ea:0.0～+1.0)

```
     v-mul-vr0:
     音声振幅 * (mix_master*mv) * vhArray[ round( vhDenom * (vNum/
vDenom) * @ea ) ]

     v-mul-vr1:
     音声振幅 * (mix_master*mv) * vhArray[ ceil( vhDenom * (vNum/vDenom)
* @ea ) ]

     v-mul-vr2:
     v = floor( vhDenom * ((vNum+1)/vDenom) * @ea )
     if (v > vNum) { v = vNum; }
     音声振幅 * (mix_master*mv) * vhArray[v]

     v-mul-vr3:
     音声振幅 * (mix_master*mv) * vhArray[ floor( vhDenom * (vNum/
vDenom) * @ea ) ]
```

v/add/vr0～3：　　(@ea:0.0～+1.0)

```
     v-add-vr0:
     音声振幅 * (mix_master*mv)
      * vhArray[ round( vhDenom + (((vNum/vDenom)*vhDenom)-vhDenom) +
(@ea*vhDenom-vhDenom) ) ]

     v-add-vr1:
     音声振幅 * (mix_master*mv)
      * vhArray[ ceil( vhDenom + (((vNum/vDenom)*vhDenom)-vhDenom) +
(@ea*vhDenom-vhDenom) ) ]

     v-add-vr2:
     v = floor( vhDenom + (((vNum+1)/vDenom)*vhDenom - vhDenom) +
(@ea*vhDenom - vhDenom) )
     if (v > vNum) { v = vNum; }
     音声振幅 * (mix_master*mv) * vhArray[ v ]

     v-add-vr3:
     音声振幅 * (mix_master*mv)
      * vhArray[ floor( vhDenom + (((vNum/vDenom)*vhDenom)-vhDenom) +
(@ea*vhDenom-vhDenom) ) ]
```

```
vl/mul/vr0～3：      (@la:0.0～-1.0(sin/saw/tri))
     vl-mul-vr0:
     * vhArray[ round( vhDenom*(vlNum/vlDenom) * (1+(@la*depth/vhDenom))
) ]

     vl-mul-vr1:
     * vhArray[ ceil( vhDenom*(vlNum/vlDenom) * (1+(@la*depth/
vhDenom)) ) ]

     vl-mul-vr2:
     v = floor( vhDenom*((vlNum+1)/vlDenom) * (1+(@la*depth/vhDenom)) )
     if (v > vlNum) { v = vlNum; }
     * vhArray[ v ]

     vl-mul-vr3:
     * vhArray[ floor( vhDenom*(vlNum/vlDenom) * (1+(@la*depth/vhDenom))
) ]
```

vl/add/vr0〜3：　　(@la:0.0〜-1.0(sin/saw/tri))

```
vl-add-vr0:
* vhArray[
     round(
          vhDenom +
          (vhDenom*(vlNum/vlDenom)-vhDenom) +
          ((1+(@la*depth/vhDenom))*vhDenom-vhDenom)
     )
]

vl-add-vr1:
* vhArray[
     ceil(
          vhDenom +
          (vhDenom*(vlNum/vlDenom)-vhDenom) +
          ((1+(@la*depth/vhDenom))*vhDenom-vhDenom)
     )
]

vl-add-vr2:
v = floor(
     vhDenom +
     (((vlNum+1)/vlDenom)*vhDenom - vhDenom) +
     ((1+(@la*depth/vhDenom))*vhDenom - vhDenom)
)
if (v > vlNum) { v = vlNum; }
* vhArray[ v ]

vl-add-vr3:
* vhArray[
     floor(
          vhDenom +
          (vhDenom*(vlNum/vlDenom)-vhDenom) +
          ((1+(@la*depth/vhDenom))*vhDenom-vhDenom)
     )
]
```

・type=linear のとき（線形モード）

【記述例】

```
#MB:VOLUME_SCALE vs=1 {
   type=linear, v=15, @ea=mul, vl=15, @la=mul,
}
t70 l4 vs1 v15 a(3 a(3 a(3 a(3 a(3 a;
```

【解説】

フォーマットは次の通りです。

```
#MB:VOLUME_SCALE vs=定義番号 {
     type=linear, v=[], @ea=[], vl=[], @la=[],
}
```

v：

vコマンドで指定する最大値を指定します。
設定範囲は 4 ～ 10000 の整数です。

@ea：

vコマンドに対応する音量エンベロープ（@ea）の計算方法の名称を指定します。
設定できる名称は、mul または add です。
　　mulの場合は、@eaエンベロープは v 設定値への乗算
　　addの場合は、@eaエンベロープは v 設定値への加算
となります。
※@ea=addでは、@eaの分母を v の分母に合わせて、@eaのカレント値（最大レベルから減った分）を v 設定値に加算します。

vl：

vlコマンドで指定する最大値を指定します。
設定範囲は 4 ～ 10000 の整数です。

@la：

vlコマンドに対応する音量ＬＦＯ（@la）の計算方法の名称を指定します。
設定できる名称は、mul または add です。
　　mulの場合は、@la音量ＬＦＯは vl 設定値への乗算
　　addの場合は、@la音量ＬＦＯは vl 設定値への加算
となります。

※@la=addでは、@laの値を vl の分母に合わせて vl 設定値に加算します。
（@laのカレント値は、０以下の数値になる想定です。０の時が vl の音量そのもの、負数になる程に小さい音量になります。正数になった場合は０に矯正されます）

```
v/mul：          (@ea:0.0～+1.0)
     v-mul:
     音声振幅 * (mix_master*mv) * (vNum/vDenom) * (@ea)

v/add：          (@ea:0.0～+1.0)
     v-add:
     音声振幅 * (mix_master*mv) * (1 + (vNum/vDenom-1) + (@ea-1))

vl/mul：         (@la:0.0～-1.0(sin/saw/tri))
     vl-mul:
     * (vlNum/vlDenom) * (1+(@la*depth/vlDenom))

vl/add：         (@la:0.0～-1.0(sin/saw/tri))
     vl-add:
     * (1 + (vlNum/vlDenom-1) + ((1+(@la*depth/vlDenom))-1))
```

・type=exponential のとき（指数モード）

【記述例】

```
#MB:VOLUME_SCALE vs=1 {
   type=exponential, curve=dB, rate=-3.0,
   v=15, @ea=mul, v0act=mute,
   vl=15, @la=mul, vl0act=mute,
}
```

【解説】

フォーマットは次の通りです。

```
#MB:VOLUME_SCALE vs=定義番号 {
     type=exponential, curve=[], rate=[],
     v=[], @ea=[], v0act=[],
     vl=[], @la=[], vl0act=[],
}
```

curve：

指数カーブのパターン名称を指定します。
設定できるパターン名称は、dB または dBB です。
　　dBの場合は、デシベルスケール
　　dBBの場合は、デシベルバイナリモードスケール
となります。

rate：

仮想音量ハードの変化レートを指定します。
設定範囲は、-24 以上、0 未満で、単位はcurveの設定に従います。
小数以下の指定も受け付けます。
例えば、curve=dB, rate=-1.5, と指定した場合、
音量が１減るごとに、振幅が -1.5dB されます。

v：

vコマンドで指定する最大値を指定します。
設定範囲は 4 ～ 10000 の整数です。

@ea：

vコマンドに対応する音量エンベロープ（@ea）の計算方法の名称を指定します。
設定できる名称は、mul または add です。
　　mulの場合は、@eaエンベロープは v 設定値への乗算
　　addの場合は、@eaエンベロープは v 設定値への加算
となります。
※@ea=addでは、@eaの分母を v の分母に合わせて、@eaのカレント値（最大レベルから減った分）を v 設定値に加算します。

v0act：

vコマンドに0が指定された時に、無音とするか、指数計算をするか（無音にならない）のモード名称を指定します。
設定できるモード名称は、mute または calc です。
　　mute の場合は、音量0で無音
　　calc の場合は、音量0で他の音量同様の指数計算
となります。

vl：
vlコマンドで指定する最大値を指定します。
設定範囲は 4 ～ 10000 の整数です。

@la：
vlコマンドに対応する音量ＬＦＯ（@la）の計算方法の名称を指定します。
設定できる名称は、mul または add です。
　　mulの場合は、@la音量ＬＦＯは vl 設定値への乗算
　　addの場合は、@la音量ＬＦＯは vl 設定値への加算
となります。
※@la=addでは、@laの値を vl の分母に合わせて vl 設定値に加算します。
（@laのカレント値は、０以下の数値になる想定です。０の時が vl の音量そのもの、負数になる程に小さい音量になります。正数になった場合は０に矯正されます）

vl0act：
vlコマンドに０が指定された時に、無音とするか、指数計算をするか（無音にならない）のモード名称を指定します。
設定できるモード名称は、mute または calc です。
　　mute の場合は、音量０で無音
　　calc の場合は、音量０で他の音量同様の指数計算
となります。

v/mul：　　(@ea:0.0～+1.0)

　　v-mul:

　　音声振幅 * (mix_master*mv) * exp_by_v(vDenom - (vNum*@ea))

v/add：　　(@ea:0.0～+1.0)

　　v-add:

　　音声振幅 * (mix_master*mv) * exp_by_v((vDenom-vNum) + (vDenom-(@ea*vDenom)))

vl/mul：　　(@la:0.0～-1.0(sin/saw/tri))

　　vl-mul:

　　* exp_by_vl(vlDenom - (vlNum*(1+(@la*depth/vlDenom))))

vl/add：　　(@la:0.0～-1.0(sin/saw/tri))

　　vl-add:

　　* exp_by_vl((vlDenom-vlNum) + (vlDenom-((1+(@la*depth/vlDenom))*vlDenom)))

4.9. v[1]：ボリューム（for @ea）

【記述例】

```
v15 cde v12 fga
```

この場合、ボリューム15で cde を演奏し、
ボリューム12で fga を演奏します。

【解説】

ボリューム（音量エンベロープ用の振幅倍率）を設定します。

引数[1]にはボリューム値を数値で指定します。
数値の範囲や、数値に対応する振幅倍率は、vsコマンドの設定内容に従います。

トラック先頭における初期設定は、v13 です。
（指数モード、v範囲0～15、減衰3dB単位、v0は無音、の設定）

vコマンドの振幅倍率には、音量エンベロープ(@ea)の時間変化が掛かります。

線形音量、指数音量では、指定数値が大きくなるほど音量が大きくなります。
また、vsコマンドにて、vコマンドの小数以下を丸めない設定では、vコマンドの小数以下の指定も有効になります。

テーブル定義音量では、テーブル設定内容に従った音量変化になります。
この場合、vコマンドの指定値はテーブルの何番目を参照するかの数値になりますので、vコマンドの小数以下の指定は必ず丸められます。小数以下の丸め方は、vsコマンドの設定に従います。

【備考】
音量設定の前提事項

4.10.）または（：相対ボリューム（for @ea）

【記述例】

```
l8 v10 c)d)e)f)g ((a (3 b
```

この場合、各音符の音量設定は次のようになります。
c:v10
d:v11
e:v12
f:v13
g:v14
a:v12
b:v9

【解説】
当コマンド指定時の vコマンド（ボリューム）の設定状態から、
相対的に数値変更したボリュームに設定します。
)　　ボリューム値を１上げます。（v10だった場合v11にする）
(　　ボリューム値を１下げます。（v10だった場合v9にする）

２以上変更したい場合は、複数個記述するか、数字を添えて記述します。
)5　ボリューム値を５上げます。
(5　ボリューム値を５下げます。

数字を添える記述では、小数以下の指定も受け付けますが、演算を行った結果のボリューム設定（最大・最小制限、小数以下の処理）は、vsコマンドの設定による制限に従います。

小カッコの向きと上下の対応を反転したい場合は、
「#MB:CONFIG {...}」の「relative_sign」項目によって反転できます。

【例】

```
#MB:CONFIG { relative_sign: oct_updown=><: vol_updown=)(, }
```

この場合、「)」で音量UP、「(」で音量DOWNになります。

4.11. vl[1]：ボリュームＬモード（for @la）

【記述例】

```
#MB:ENV_A @ea=1 { peak=15,init=&,| n:1:15, r:1:0, }
#MB:LFO_A @la=1 { depth=4, width=24, delay=0, form=sin_a, }
t70 v15 @ea1 @la1
vl15 c4.r8 vl13 c4.r8 vl11 c4.r8 vl9 c4.r8 vl7 c4.r8;
```

この場合、ボリュームＬモードが15から 2 ずつ下がって音符が演奏されます。
@la1の音量ＬＦＯにはdelayが掛かっていないため vlコマンドがノートオン直後から効いています。

【解説】
ボリュームＬモード（音量ＬＦＯ用の振幅倍率）を設定します。

引数[1]には音量を数値で指定します。
数値の範囲や、数値に対応する振幅倍率は、vsコマンドの設定内容に従います。

トラック先頭における初期設定は、vl15 です。
（デシベルモード、vl範囲0～15、減衰3dB単位、vl0は無音、の設定）

vlコマンドの振幅倍率には、音量ＬＦＯ(@la)の時間変化が掛かります。

【注】音量ＬＦＯ(@la)が掛かっている間だけ、vlコマンドが有効です。
（@laのdelay時間の最中は、@laが無効のためvlも無効です）

線形音量、指数音量では、指定数値が大きくなるほど音量が大きくなります。
また、vsコマンドにて、vlコマンドの小数以下を丸めない設定では、vlコマンドの小数以下の指定も有効になります。

テーブル定義音量では、テーブル設定内容に従った音量変化になります。
この場合、vlコマンドの指定値はテーブルの何番目を参照するかの数値になりますので、vlコマンドの小数以下の指定は必ず丸められます。小数以下の丸め方は、vsコマンドの設定に従います。

【備考】
音量設定の前提事項

4.12. vf[1],[2]：ボリュームファンクション

【記述例】

```
#MB:CONFIG {
   env_clock: unit=sec: rate=1/60,
   env_resol: unit=sec: rate=1/60,
   lfo_clock: unit=sec: rate=1/60,
   lfo_resol: unit=sec: rate=1/60,
}
#MB:VOLUME_SCALE vs=1 { type=linear, v=16, vr=2, vl=16, vlr=2, vh=15 }
#MB:ENV_A @ea=1 { peak=15, init=& | n:0:15, r:0:15, n:600:15, n:1:0, }
#MB:LFOTBL_ATK 4 {
   loop=0, cmpl=0, offset=-15, denom=1, width_mode=step, shift_reso=1,|
   15,
}
#MB:LFOTBL_REL 120 {
   loop=0, cmpl=0, offset=-15, denom=1, width_mode=step, shift_reso=1,|
   2,
}
#MB:LFO_A @la=4 { depth=1, width=1, delay=0, form=table, tbl_atk=4, tbl_rel=120 }

t80 @ea1 q8,16 vs1 v16 vl8 vf21,1 @la4 ccccc z;
```

この場合、vl8によって音量８で演奏しますが、エンベロープがリリース中のエンベロープレベル「２」の演奏は、vf21,1により、vl8ではなく最大vl値での演奏になります。

【解説】

音量に関する特殊機能の設定を行います。

指定する引数は次の通りです。
引数[1]...特殊機能の種別番号（funcType）
引数[2]...設定値（param）
引数はカンマで区切って指定します。
トラック先頭における初期設定は、どのfuncTypeも無効に設定されています。

引数[1]（funcType）

現状サポートされているfuncTypeは、次の通りです。
・20（エンベロープがリリース中のみ、vを強制的に最大として処理）
・21（エンベロープがリリース中のみ、vlを強制的に最大として処理）

引数[2]（param）

funcTypeによって指定内容や効果が変わります。

funcType＝20の場合のparam

エンベロープがリリース中のみ、v コマンドの音量を最大値として扱う特殊機能の有効無効を切り替えます。

vf20,0　　機能の無効（初期設定）
vf20,1　　機能の有効

funcType＝21の場合のparam

エンベロープがリリース中のみ、vl コマンドの音量を最大値として扱う特殊機能の有効無効を切り替えます。

vf21,0　　機能の無効（初期設定）
vf21,1　　機能の有効

4.13. @fo[1],[2]：フェードアウト

【記述例】

```
@fo3,-48 cdefg
```

@fo3,-48記述以降の演奏に対し、3 秒間かけて、デシベルスケールで 0dB から -48dB まで減衰させます。

【解説】

フェードアウト機能です。
当コマンドを使用したトラックのみ、音量を徐々に減衰させます。

指定する引数は次の通りです。
引数[1]...フェードアウト時間（time）
引数[2]...音量変化させる範囲（range）
引数はカンマで区切って指定します。

引数[1]（time）

フェードアウトを行う時間を 0 以上の数値で指定します。
単位は秒です。小数以下の指定も受け付けます。
0 を指定した場合は特例で、フェードアウト処理中であっても、フェードアウト機能を中断して、通常音量(0dB)に戻ります。この場合、range 指定は無効です。

引数[2]（range）

フェードアウトによって減衰する音量範囲を指定します。
範囲は -110 ～ 0 の整数で、単位はデシベルです。
減衰量の指定なのでマイナス数値であることに注意してください。
0 未満の場合、指定デシベルまで指数変化で減衰します。
0 の場合は特例で、無音まで線形変化で減衰します。
0 より大きい指定は、0 を指定したものとみなされます。
range は省略可能で、省略した場合は 0 を指定したものとみなされます。

【備考１】
最後に指定したフェードコマンドが優先されます。
例えば、長いフェードアウトの途中で、フェードインに切り替えて動作させることもできます。

【備考２】

フェードアウトのゼロ時間指定と、フェードインのゼロ時間指定（@fo0 と @fi0）の内部処理は同一です。

4.14. @fi[1],[2]：フェードイン

【記述例】

```
@fi0.5,-48 cdefg
```

@fi0.5,-48記述以降の演奏に対し、0.5 秒間かけて、デシベルスケールで、
-48dB から 0dB まで音量を増加させます。

【解説】

フェードイン機能です。
当コマンドを使用したトラックのみ、音量を徐々に増加させます。
・増加する音量に利得は含みません。
・最低音量から通常音量までの増加です。

指定する引数は次の通りです。
引数[1]...フェードイン時間（time）
引数[2]...フェードイン開始時点の最低音量（range）
引数はカンマで区切って指定します。

引数[1]（time）

フェードインを行う時間を 0 以上の数値で指定します。
単位は秒です。小数以下の指定も受け付けます。
0 を指定した場合は特例で、フェードイン処理中であっても、フェードイン機能を中断して、通常音量(0dB)に戻ります。この場合、range 指定は無効です。

引数[2]（range）

フェードイン開始時点の最低音量を指定します。
指定範囲は -110 ～ 0 の整数で、単位はデシベルです。
最低音量の指定なので、マイナス数値であることに注意してください。
0 未満の場合、指定デシベルから通常音量まで指数変化で増加します。
0 の場合は特例で、無音から通常音量まで線形変化で増加します。
0 より大きい指定は、0 を指定したものとみなされます。
range は省略可能で、省略した場合は 0 を指定したものとみなされます。

【備考１】
最後に指定したフェード機能が優先されます。
例えば、長いフェードアウトの途中で、フェードインに切り替えて動作させることもできます。

【備考２】
フェードアウトのゼロ時間指定と、フェードインのゼロ時間指定（@FO0 と @FI0）の内部処理は同一です。

5.制御関連

5.1. プリプロセッサの処理順序

【解説】
ＭＭＬの演奏要求（play）を受け付けると、ＭＭＬテキストがコンパイルされて演奏が始まります。

ここで言うコンパイルとは、ＭＭＬテキストを演奏できる形式に翻訳することですが、この翻訳の直前に、ＭＭＬテキスト全体に対し前処理を行なっていて、この前処理を行う機能のことをプリプロセッサと呼んでいます。

プリプロセッサの処理順序は次の通りです。

（１）コメント処理（コメントブロック処理後、コメントラインを処理）
（２）インクルード処理（インクルード対象内のコメント処理も含む）
（３）メタデータ処理
（４）マクロ処理
（５）繰り返しの展開処理

上記の処理を順に経てからＭＭＬコンパイルに入ります。

そのため、
・コメントブロック中のコメントライン記述は無効。
・メタデータ定義内では、マクロ参照が出来ない。
・メタデータ定義内では、繰り返し記述が使えない。
など、解釈の挙動に特徴があります。

5.2. t[1]：テンポ

【記述例】

・通常のテンポ指定

```
t100 cdefg;
```

BPMを100に設定します。

・除算のあるテンポ指定

```
t3600/28 cdefg;
```

BPMを128.5714...に設定します。
これは、tickカウント１あたり１／６０秒の前提で、
１６分音符のtickカウントが７になるようなテンポにしたい場合、
４分音符は２８ticksになるので、秒数は２８×１／６０。
よって、
tempo = 60 / (28/60) = 3600/28 という計算です。

・だんだんゆっくりにするテンポ指定

```
l4 t120 cde t#-20 def t#-20 efg t#+40 c1;
```

t120 で開始しているので、
t#-20 def は、この場合 t100 で演奏、
t#-20 efg は、この場合 t80 で演奏、
t#+40 c1 は、この場合 t120 で演奏されます。
（「#」は前回 t に設定した値として解釈されます。トラック先頭で「#」を使用した場合は、初期設定値になります。「#」は他のトラックで指定したテンポ値を参照出来ないので注意してください）

・だんだんゆっくりにするテンポ指定（２）

```
l4 t120 cde t#*0.5 def t#*0.5 efg t#*4 c1;
```

t120 で開始しているので、
t#*0.5 def は、この場合 t60 で演奏、
t#*0.5 efg は、この場合 t30 で演奏、
t#*4 c1 は、この場合 t120 で演奏されます。

【解説】

テンポを設定します。
引数[1]には、１分間のビート数（BPM）を指定します。
小数以下の指定も受け付けます。

トラック先頭における初期設定は「t120」、１ビートは４分音符です。
１ビートの音長設定は、

```
#MB:CONFIG {
     tempo_unit: note_ticks=[]: beat_length=[],
}
```

による設定（beat_length）に従います。
また、前回テンポ設定値に対する相対指定を行いたい場合は、数式解釈機能における「#」コマンドを利用します。
ただし、この相対指定における「前回」とは同一トラック内での指定値を指します。ＭＭＬコンパイラの都合、他のトラックにおける過去指定を認識できないので、利用シーンは同一トラック内で徐々にゆっくりさせる指定を行う場合などです。

【備考１】
テンポコマンドでは、同時点の全てのトラックに対し影響します。
テンポ同期の内部手順は次の通りです。
- ＭＭＬコンパイル中に受け取ったテンポコマンドは、テンポ専用のシステム管理トラックに、一旦全て記録します（全トラック分を集約）。
- 全てのトラックのＭＭＬコンパイルが終わった後、テンポ専用トラックを読み、各トラックの同時点に対し、テンポコマンドを挿入する。

以上です。
同時点かどうかの管理には、再生開始からの累積tickカウント数を使用しています。

【備考２】
テンポの精度は、内部におけるtickカウント数あたりの再生サンプル数（整数）をいくつとするかの計算に依存しています。

【備考３】
複数トラックで、同時点に、指定値の違うテンポコマンドを記述した場合、後のトラックに記述したテンポコマンドが使われます。
内部的に、同時点のテンポコマンドは、先に定義されたトラックのものから順に各トラックに配信されるため、結果的に最後のトラックに記述したテンポコマンドが有効になります。

【備考４】
トラックごとにテンポの同期を厳守しようとする設計思想になっています。この設計は、無限リピートの仕様にも影響しています。

5.3. ;（セミコロン）：トラック区切り

【記述例】

```
cdefg;
```

ドレミファソを演奏するだけのトラックを定義しています。

【解説】

セミコロンは、次の用途で使用します。
（１）トラック定義終端
（２）マクロ定義終端

（１）の用途は、複数トラックを定義したい場合の区切りです。
トラック終端にセミコロンを設けると、それまでのＭＭＬ記述が１つのトラックとして認識され、以降を次のトラックの先頭として扱います。
（トラックとトラックが区切られます）

最終トラックには、セミコロンはあってもなくても構いませんが、トラックがいつでも追加できるよう準備する観点から、全てのトラックの終端にはセミコロンを記述しておくことをお勧めします。

【備考１】
最終トラックを記述後、セミコロン無しでテキストが終わった場合は、テキスト終端がトラック終端として扱われます。

【備考２】
最終トラック記述後、セミコロン有りでテキストが終わった場合、テキスト終端にＭＭＬイベントの無いトラックが存在する扱いになりますが、ＭＭＬイベントの無いトラックは定義されずに破棄されます。

【備考３】
セミコロンを連続記述するなどして、空のトラックを定義しても、空のトラックは全て破棄されます。

（２）の用途は、マクロ定義、
$...=.....;
の終端記述です。

5.4. /* ... */：コメントブロック

【記述例】

```
/* ----- // ----- */
```

この場合、コメントブロックが優先的に処理されるため、行の途中のコメントライン開始記述は、コメントブロック内のコメントとして処理されます。

【解説】

コメントブロックを設定します。
/* と */ に囲まれた文字列を、コメントエリアとして読み飛ばします。
コメントエリア内には、全角文字や改行を含んでいても問題ありません。

【備考】

プリプロセッサの処理順序により、コメントライン処理よりも、コメントブロック処理が優先されます。

5.5. //：コメントライン

【記述例】

```
//cdefg;
```

この場合、「cdefg;」はコメント扱いになり、演奏されません。

```
#ML:TITLE テストタイトル//test title
```

この場合、「//test title」の文字列は、コメントライン処理により、
「#ML:TITLE」によるタイトル文字列には含まれないことになります。

【解説】

コメントラインを設定します。
「//」を記述して以降、改行までをコメントエリアとして読み飛ばします。
コメントエリア内には全角文字を含んでいても問題ありません。

【備考】

プリプロセッサの処理順序により、メタデータ処理よりも、コメントライン処理が優先されます。

5.6. |:[1] ... :|：繰り返し

【記述例】

```
|:3 cde | fg :|
```

この場合「cde fg cde fg cde」と同じになります。

```
|: cde | fg :|
```

この場合「cde fg cde」と同じになります。

```
|:1 cdefg :|
```

この場合、「cdefg」と同じになります。

```
|:0 cdefg :|
```

この場合、繰り返しの中身が空となります（コメントアウト状態）。
繰り返し回数を０、１に書き換えることで、部分的にＭＭＬ記述の有効、無効を切り替えることに利用できます。

【解説】

有限回数の繰り返し演奏に使用します。
引数[1]には、繰り返し回数を０以上の整数で指定します。
繰り返し記述を行うと、「|:」から「:|」の間が、引数[1] の回数だけ繰り返しになります。
引数[1]の繰り返し回数は、省略可能です。省略時の繰り返し回数は２です。

ただし、繰り返し内に「|」がある場合は、繰り返し最終回のとき、
「|」から「:|」の間がスキップされます（繰り返し脱出）。
繰り返し脱出の「|」は、省略可能です。

【備考】

この繰り返し機能は、プリプロセッサにより処理されます。
つまり単純な文字列のコピーペーストによる実装なので、無限リピートは出来ません。

5.7. @rp：無限リピートエントリ

【記述例】

```
cegec
@rp
dfafd dfafd;
```

この場合、「dfafd dfafd」が無限リピート演奏されます。

【解説】
無限リピートエントリを登録します。

このコマンドを記述したトラックでは、演奏がトラック終端に達したら、@rp を記述した時点に演奏ポイントが無条件ジャンプします。そのため、結果的に無限リピートになります。
無限リピートコマンドを使用する場合は、対象となるトラック内で「１回だけ」記述してください。

無限リピートコマンドは、テンポコマンドの使用状況と大きく関わっています。
理由は、各トラック間でのテンポ同期を確保するためです。
テンポ同期の都合、無限リピートの処理モードを、テンポコマンドの配置の仕方によって変えています。

（１）規制緩和モード
このモードは、テンポコマンドが、演奏先頭（0 tick時点）にのみ存在する場合に該当します。

【規制緩和モードの場合の @rp 使用条件】
（１.１）各トラックの tickカウント合計の状況は問われません。
（１.２）@rp コマンドは、どのトラックでも自由な時点に配置できます。
以上により、無限リピートの自由度が高まりますが、音楽的におかしなリピートも可能になるため、ユーザー側でリピート状況を確認する必要があります。

（２）厳密モード
このモードは、テンポコマンドが、演奏先頭だけでなく、それ以外の時点にも存在する場合に該当します。厳密モードでは、テンポ同期を厳守する目的で、無限リピートに関する制限を厳しくしています。

【厳密モードの場合の @rp 使用条件】
（２.１）無限リピート対象の、全てのトラックは、演奏tickカウント数の合計が一致している必要があります。
（２.２）無限リピート対象の、全てのトラックにおいて、@rp を配置する時点（演奏開始からのtickカウント数）が一致している必要があります。
（２.３）無限リピートしないトラックが混在する場合、無限リピートしないトラックの演奏tickカウント数の合計は、無限リピートするトラックのもの以下でなければなりません。

厳密モードによる制限の理由：

テンポの変更は、演奏サンプリング周波数(384kHz)に依存した、内部音長である「絶対tickカウント数」の設定変更により実装されています。絶対tickカウント数の操作は、テンポ計算にのみ任されているので、ユーザーは普段意識する必要はありません。

複数回テンポ変更する無限リピート対象トラックで、厳密モードによって制限しないと、見かけ上テンポのつじつまを合わせたとしても、使用する音長の種類と個数の違いによって、簡単に内部音長である絶対tickカウント数が合わなくなって、テンポを同期させることができなくなります。（システム都合のテンポ非同期が発生します）

そのため、厳密モードでは、無限リピートで何周しても同期するよう、テンポ変更のタイミング、無限リピートエントリの位置、トラックの演奏tickカウント数の合計を合わせることを条件にしています。

（２.３）の制限は、２周目以降のリピートに対応する同期が、無限リピートしないトラックに対して行えないために設けています。

【備考１】@rpの優先配置
@rp コマンドは、ＭＭＬコンパイル時、同時点（演奏開始からの累積tickカウント数が同じ時点）に存在する他のコマンドより前に配置されます。
例えば、

```
@@"pls" v8 cdefg @@"saw" v12 @rp gfedc;
```

という記述の場合、見た目では @@"saw" v12 が無限リピート開始時に実行されないように見えますが、内部的には、

```
@@"pls" v8 cdefg @rp @@"saw" v12 gfedc;
```

このように同時点コマンド群の先頭に自動配置されるため、同時点のコマンドはすべて無限リピートエントリ時に実行の対象になります。
この、自動配置の目的は、テンポコマンドの実行もれ防止です。

@rpと同時点にテンポコマンドが存在した場合の、実行もれを防止することで、テンポ同期が取れない状況が発生しないようにしています。

【備考２】２周目以降に向けたＭＭＬ擬似コマンドは無効

無限リピート開始時点において、ＭＭＬ擬似コマンドによる演奏設定は、常に無限リピート初回演奏の条件での適用になります。
ＭＭＬ擬似コマンドとは、例えば l4 などです。
【例１】音長指定

```
l4 cdef @rp gab>c l1;
```

この場合、リピート始めの g は、常に l4 で演奏されてしまいます。
これは、ＭＭＬコンパイル時点で g が l4 でコンパイルされて音長が確定していて、トラック終端に l1 指定があったとしても、無限リピートでジャンプした先にある、既に生成済みの音符の音長に作用できないためです。
同様に、他のＭＭＬ擬似コマンドも作用できない性質を持っています。
これは無限リピート機能特有の問題です。

【例２】相対ボリューム

```
v13 l4 cdef @rp gab>c (2;
```

相対ボリューム（小かっこ）は、vコマンドの固定値に変換されているため、トラック終端に置いて無限リピートする場合、置いた１回分のみ有効です。
この例では記述上、繰り返すたびにボリュームが小さくなっていくかのように見えますが、実際には、

```
v13 l4 cdef @rp gab>c v11;
```

このように記述したのと同様の動作になります。

【例３】音程の補助指定
　　オクターブ関連：o　<　>
　　ノートシフト：ns
これら音程関連のコマンドは、音符生成時に参照され、音程番号に変換されて演奏データになるため、トラック終端に置いても、音長指定同様、リピート開始時の音符に作用できません。

【例４】指定値の分母設定関連
q の分母指定
@w の分母指定
これらの分母指定も、トラック終端で分母を設定変更しても無限リピート開始時に作用できません。

5.8. #MB:CONFIG：初期設定関係

【記述例】

```
#MB:CONFIG {
   tempo_unit: note_ticks=192: beat_length=4,
   mixing_vol: master_rate=0.5,
   relative_sign: oct_updown=><: vol_updown=)(,
   env_clock: unit=sec: rate=1/60,
   env_resol: unit=sec: rate=1/60,
   lfo_clock: unit=sec: rate=1/60,
   lfo_resol: unit=sec: rate=1/60,
   lfo_table: process_mode=0,
   damper_rate: env=0.03: fms=15,
   fms_master: clock=4000000: prescale=64,
   fms_hlfo: opm_spd=1.0: opna_spd=1.0,
   wvm_over_sampling: rate=2,
   nzw_noise: base_cycle=3993600/32,
   nzp_noise: clock=3993600: prescale=32,
   pls_noise: clock=3993600: prescale=32: width=17,
   nzc_noise: step_adj=1.0,
}
```

【解説】

#MB:CONFIG では、ＭＭＬコンパイル前の各種初期設定が可能です。
定義方法の概要は、
・行頭からの記述で、キーワード「#MB:CONFIG」を書く。
・続けて、スペースを置き、中括弧「{}」で括られた中に定義データを書く。
・定義データは、「,」で区切って記述します。
・「,」区切りの中身は、「:」で区切られます。
・「:」で区切られた中身は、
　　機能名: 項目名(1)=定義値: ... 項目名(n)=定義値:,
　というような形式で、項目名の個数は１個以上です。
・スペースや改行は読み飛ばされます。
以上です。
複数の #MB:CONFIG {} を記述してしまい、中身の設定がかち合った場合は、後に記述した方が有効になります。

・機能名：tempo_unit：テンポコマンドの単位設定

【記述例】

```
#MB:CONFIG {
   tempo_unit: note_ticks=192: beat_length=4,
}
```

【解説】

項目名(1)：note_ticks
定義値には、１分音符の tickカウント数を、いくつにするか設定します。
初期設定は 192 です（１tick = 192分音符）。

設定条件は次の通りです。
・４８以上の整数であること。
・beat_lengthの定義値で割り切れること。

【備考１】
４分音符などの音長は、tickカウント数への換算が自然数になるように作られています。これは、テンポずれ起こさないようにするためです。

「tickカウント換算が自然数」についてもう少し考えてみます。
例えば、１９２分音符単位での tickカウント換算は、このようになります。

音符	192	96	64	48	32	24	16	12	8	6	4	3	2	1
tick	1	2	3	4	6	8	12	16	24	32	48	64	96	192

これらは tickカウント換算が自然数になる物のうち、付点音符を含まないものです。付点音符においても、tickカウント換算が自然数になるものを使います。

【備考２】
連符の考え方について。例えば４分音符の３連符の場合。
まず４分音符は、全音符の１／４です。
さらに、４分音符の３分割は、
（１／４）×（１／３）＝（１／１２）
となるので１２分音符が作れれば良い、と考えていきます。

例えば、４分音符の長さの９連符が欲しい場合、３６分音符が必要になります。
しかし、初期設定の１９２分音符単位では３６分音符は作れません。
そこで、１４４分音符単位を考えてみます。

１４４分音符単位の音長ラインナップは、
144, 72, 48, 36, 24, 18, 16, 12, 9, 8, 6, 4, 3, 2, 1
ですので、３６分音符が使えます。

このように、特別な音長単位を使いたい場合、曲目で使用する音長ラインナップを全て含む「note_ticks」を設定します。

項目名(2)：beat_length
定義値には、１ビートを何分音符とするかを設定します。
beat_length=4 とした場合、１ビートは４分音符になります。

・機能名：mixing_vol：ミキシングボリュームの基準設定

【記述例】

```
#MB:CONFIG {
   mixing_vol: master_rate=0.5,
}
```

【解説】

ミキシングボリュームの音量基準となる倍率を指定します。
この設定は、全トラックへの初期設定となります。

項目名(1)：master_rate

定義値には、ミキシングボリュームの音量基準を数値で指定します。
指定値に数値（計算式）のみを書いた場合、音量基準への倍数として扱われます。この場合、０より大きい値である必要があります。
数値の末尾に「dB」と記述した場合、数値がデシベル値として扱われます。
数値の末尾に「dBB」と記述した場合、数値がデシベルバイナリモード値として扱われます。
dB値 または dBB値の場合、数値が正の場合は増幅になり、負の場合は減衰になります。

【備考】

音量設定の前提事項

・機能名：relative_sign：相対記号<>()の方向設定

【記述例】

```
#MB:CONFIG {
   relative_sign: oct_updown=><: vol_updown=)(,
}
```

【解説】

相対オクターブ記号、相対ボリューム記号の方向を設定します。
この設定は、全トラックへの初期設定となります。

項目名(1)：oct_updown

相対オクターブ記号の方向として、定義値には次のいずれかを指定します。

【相対オクターブの方向設定表】

設定文字列	設定内容
><	「>」でUP、「<」でDOWN
<>	「<」でUP、「>」でDOWN

項目名(2)：vol_updown

相対ボリューム記号の方向として、定義値には次のいずれかを指定します。

【相対ボリュームの方向設定表】

設定文字列	設定内容
)(	「)」でUP、「(」でDOWN
()	「(」でUP、「)」でDOWN

・機能名：env_clock：エンベロープ時間単位設定

【記述例】

```
#MB:CONFIG {
   env_clock: unit=sec: rate=1/60,
}
```

【解説】

エンベロープ全種類への、エンベロープの時間単位を設定します。
この設定は、全トラックへの初期設定となります。

当設定を使用しなかった場合の初期設定は次の通りです。

```
#MB:CONFIG {
   env_clock: unit=sec: rate=1/127,
}
```

項目名(1)：unit

定義値には、時間単位を文字列で指定します。

【エンベロープ時間単位設定】

設定文字列	設定内容
sec	時間単位は「１秒」
tick	時間単位は「１tickカウント」

項目名(2)：rate

定義値には、時間単位に対する倍率を数値（計算式）で指定します。
時間単位が「sec」だった場合、rateの有効範囲は、1/2400 ～ 1 になります。

【備考】

時間単位が「１tickカウント」の場合、テンポにより時間単位が変動します。
エンベロープ指定後にテンポを変更した場合、テンポ変更後のノートオンからエンベロープの時間計算が自動追従します。
つまり、ノートオン中にテンポ変更が掛かった場合は、そのノートオン中は時間計算の自動追従はできませんが、次のノートオンから自動追従します。

・機能名：env_resol：エンベロープ解像度設定

【記述例】

```
#MB:CONFIG {
   env_resol: unit=sec: rate=1/300,
}
```

【解説】

エンベロープ全種類への、エンベロープの解像度を設定します。
エンベロープ解像度とは、振幅計算更新の最小時間の単位です。
この設定は、全トラックへの初期設定となります。

当設定を使用しなかった場合の初期設定は次の通りです。

```
#MB:CONFIG {
   env_resol: unit=smp,
}
```

項目名(1)：unit

定義値には、時間単位を文字列で指定します。

【エンベロープ解像度単位設定】

設定文字列	設定内容
smp	時間単位は「384kHzの１サンプル」
sec	時間単位は「１秒」
tick	時間単位は「１tickカウント」

項目名(2)：rate

定義値には、時間単位に対する倍率を数値（計算式）で指定します。
時間単位が「smp」だった場合、rateは不要です。
時間単位が「sec」だった場合、rateの有効範囲は、1/2400 ～ 1 になります。

【備考】
時間単位が「１tickカウント」の場合、テンポにより時間単位が変動します。
エンベロープ指定後にテンポを変更した場合、テンポ変更後のノートオンからエンベロープの時間計算が自動追従します。

つまり、ノートオン中にテンポ変更が掛かった場合は、そのノートオン中は時間計算の自動追従はできませんが、次のノートオンから自動追従します。

・機能名：lfo_clock：ＬＦＯ時間単位設定

【記述例】

```
#MB:CONFIG {
   lfo_clock: unit=sec: rate=1/60,
}
```

【解説】

ＬＦＯ全種類への、時間単位を設定します。
時間単位は、width の周期時間と、delay に影響します。
この設定は、全トラックへの初期設定となります。

当設定を使用しなかった場合の初期設定は次の通りです。

```
#MB:CONFIG {
   lfo_clock: unit=tick: rate=1,
}
```

項目名(1)：unit

定義値には、時間単位を文字列で指定します。

【ＬＦＯ時間単位設定】

設定文字列	設定内容
sec	時間単位は「１秒」
tick	時間単位は「１tickカウント」

項目名(2)：rate

定義値には、時間単位に対する倍率を数値（計算式）で指定します。
時間単位が「sec」だった場合、rateの有効範囲は、1/2400 ～ 1 になります。

【備考】
時間単位が「１tickカウント」の場合、テンポにより時間単位が変動します。

・機能名：lfo_resol：エンベロープ解像度設定

【記述例】

```
#MB:CONFIG {
   lfo_resol: unit=sec: rate=1/300,
}
```

【解説】

ＬＦＯ時間軸の解像度（ＬＦＯ変位計算の最少時間単位）を設定します。
この設定は全トラックへの初期値となります。
この設定が有効になる対象は、
　　ピッチＬＦＯ
　　yコマンドＬＦＯ
　　フィルタＬＦＯ
の３種類です。

ＬＦＯ解像度は、ポルタメント機能による周波数変更の最小時間単位にもなっています。

当設定を使用しなかった場合の初期設定は次の通りです。

```
#MB:CONFIG {
   lfo_resol: unit=sec: rate=1/600,
}
```

項目名(1)：unit

定義値には、時間単位を文字列で指定します。
【ＬＦＯ解像度単位設定】

設定文字列	設定内容
sec	時間単位は「１秒」
tick	時間単位は「１tickカウント」

項目名(2)：rate

定義値には、時間単位に対する倍率を数値（計算式）で指定します。
時間単位が「smp」だった場合、rateは不要です。
時間単位が「sec」だった場合、rateの有効範囲は、1/2400 ～ 1 になります。

【備考１】
時間単位が「１tickカウント」の場合、テンポにより時間単位が変動します。

【備考２】
音量とパンポットのＬＦＯ（@la / @lb）は、常に1サンプル単位で動作するので、本機能の対象外になります。

・機能名：lfo_table：ＬＦＯテーブル処理モード

【記述例】

```
#MB:CONFIG {
   lfo_table: process_mode=0,
}
```

【解説】

ＬＦＯにおける、ユーザー定義テーブルのプロセスモード（処理モード）を設定します。
この設定は、全トラックへの初期設定となります。

項目名(1)：process_mode

定義値には、0 または 1 でモード番号を指定します。

0 のとき、テーブル値を処理してから、ポインタを進めます。
1 のとき、ポインタを進めてから、テーブル値を処理します。

特に必要性が無い場合、0 を使用してください。
初期設定は 0 です。

・機能名：damper_rate：ダンパー初期設定

【記述例】

```
#MB:CONFIG {
   damper_rate: env=0.05: fms=15,
}
```

【解説】

音量エンベロープダンパー（zコマンド）の、初期設定を行います。
この設定は、全トラックへの初期設定となります。
音量エンベロープダンパーは、内部で音量エンベロープ（@ea）向けとＦＭ音源（@@"fms"）向けの２種類に分かれており、初期設定はそれぞれに行います。
当設定を使用しなかった場合の初期設定は次の通りです。

```
#MB:CONFIG {
   damper_rate: env=0.03: fms=15,
}
```

項目名(1)：env

定義値には、0 以上の数値を指定します。小数以下の指定も受け付けます。
指定値は音量エンベロープ（@ea）用のエンベロープダンパーに使われます。
また、指定値は、エンベロープ定義内における変化率（ Ｉ のモード）によるレートとして解釈されます。減衰時間はエンベロープクロック（時間単位）の設定に依存します。

項目名(2)：fms

定義値には、0 ～ 15 の整数を指定します。
指定値は、ＦＭ音源モジュール（@@"fms"）用のエンベロープダンパーに使われます。
ダンパー実行時に、全てのオペレータのリリースレートが、指定値に書き換えられます。

・機能名：fms_master：@@"fms"用マスタークロック設定

【記述例】

```
#MB:CONFIG {
   fms_master: clock=4000000: prescale=64,
}
```

【解説】

ＦＭ音源モジュール（@@"FMS"）の、マスタークロック初期設定を行います。
この設定は、全トラックへの初期設定となります。
当設定を使用しなかった場合の初期設定は次の通りです。

```
#MB:CONFIG {
   fms_master: clock=3579545: prescale=64,
}
```

項目名(1)：clock

定義値には、仮想ＦＭ音源のマスタークロックを数値で指定します。
少数以下の指定も受け付けます。
設定可能範囲は、clock÷prescaleが、32000～96000 となる範囲です。

項目名(2)：prescale

定義値には、分周比を数値で指定します。マスタークロックを割る数です。
少数以下の指定も受け付けます。
設定可能範囲は、clock÷prescaleが、32000～96000 となる範囲です。

【備考１】
ＯＰＭ想定の場合の例：

```
   fms_master: clock=3579545: prescale=64,    //namco  (55930.39..)
   fms_master: clock=4000000: prescale=64,    //X68000 (62500)
```

ＯＰＮ系想定の場合の例：

```
   fms_master: clock=3993600: prescale=72,    //PC88   (55466.66..)
```

【備考２】
当設定（fms_master）の影響範囲は、
・ＦＭ音源のエンベロープシーケンス速度
・ＯＰＭ互換ハードウェアＬＦＯシーケンス速度（@mh）
・ＯＰＮＡ互換ハードウェアＬＦＯシーケンス速度（@mha）
・デチューン１計算
・@m0 と指定した場合の波形生成レート
です。
音程計算は使用中のピッチスケールに従って計算するため、当設定には依存しません。

・機能名：fms_hlfo：@@"fms"用ＨＬＦＯ速度調整

【記述例】

```
#MB:CONFIG {
   fms_hlfo: opm_spd=1.0: opna_spd=1.0,
}
```

【解説】

ＦＭ音源モジュール（@@"fms"）の、ハードＬＦＯ（ＨＬＦＯ）速度調整の初期設定を行います。
この設定は、全トラックへの初期設定となります。
当設定を使用しなかった場合の初期設定は次の通りです。

```
#MB:CONFIG {
   fms_hlfo: opm_spd=1.0: opna_spd=1.0,
}
```

項目名(1)：opm_spd

定義値には、ＯＰＭ互換ＨＬＦＯの速度倍率を数値で指定します。
設定範囲は 0.5 ～ 2.0 です。1.0の場合速度変化なし（等倍）です。
小数点以下の桁数は制限していません。

項目名(1)：opna_spd

定義値には、ＯＰＮＡ互換ＨＬＦＯの速度倍率を数値で指定します。
設定範囲は 0.5 ～ 2.0 です。1.0の場合速度変化なし（等倍）です。
小数点以下の桁数は制限していません。

【備考】

トラックごとに速度変更したい場合は、yコマンドによる指定で行います。
例えば、ＯＰＭの 3.58MHz モードで、4MHz モードで作り込まれたのＨＬＦＯパラメータを使用する際は、ＦＭ音源モジュール（@@"fms"）を選択した状態で、

```
y"@mh_spd_adj",1.117
```

などと指定します。（4000000÷3579545 = 1.117460459360058）

・機能名：wvm_over_sampling：オーバーサンプリング設定

【記述例】

```
#MB:CONFIG {
   wvm_over_sampling: rate=4,
}
```

【解説】

波形メモリ音源（@@"WVM"）では、エイリアシングノイズ対策のために、簡易的なオーバーサンプリング処理を行うことが出来ます。
当設定により、波形メモリ音源のオーバーサンプリングの初期設定を、何倍オーバーサンプリングにするか設定できます。
この設定は、全トラックへの初期設定となります。
当設定を使用しなかった場合の初期設定は次の通りです。

```
#MB:CONFIG {
   wvm_over_sampling: rate=2,
}
```

項目名(1)：wvm_over_sampling

定義値には、オーバーサンプリング倍率を整数で指定します。
設定範囲は 1 ～ 8 です。

【備考１】

波形メモリ音源のオーバーサンプリング設定は、ＭＭＬ記述でトラックごとに変更することもできます。その場合は、波形メモリ音源（@@"wvm"）を選択した状態で、

```
    y"oversmp",4
```

などと指定します。

【備考２】

多くのトラック定義などにより、波形メモリ音源演奏で処理が重い場合は、倍率を下げてみてください。

・機能名：nzw_noise：@@"nzw"用ベースサイクル設定

【記述例】

```
#MB:CONFIG {
   nzw_noise: base_cycle=3993600/32,
}
```

【解説】

ホワイトノイズ音源（@@"nzw"）では、変位がランダムに変更されます。
変位が更新される間隔を、@nコマンドで変更することによって、ノイズの質感を変えることができますが、当設定では、変位更新間隔の基本単位を設定することが出来ます。
この設定は、全トラックへの初期設定となります。

当設定を使用しなかった場合の初期設定は次の通りです。

```
#MB:CONFIG {
   nzw_noise: base_cycle=3993600/32,
}
```

この初期設定は、@nコマンドによる設定の音が、ＰＳＧノイズのＰＣ８８モードとよく似た質感になります。

項目名(1)：base_cycle

定義値には、ノイズ変位の更新間隔の基本単位を数値で設定します。
少数以下の指定も受け付けます。（計算式記述も可）
設定範囲は 1.0 ～ 10000000.0(10MHz) です。

【備考】
ホワイトノイズの更新間隔は、最小で1/384000秒、最大で１秒に制限されます。
そのため、384000を超える設定を行った場合の @n1 では 1/384000秒 を下回りますが、はみ出した部分の更新結果は読み捨てされます。

・機能名：nzp_noise：@@"nzp"用ベースサイクル設定

【記述例】

```
#MB:CONFIG {
   nzp_noise: clock=3993600: prescale=32,
}
```

【解説】

ＰＳＧノイズ音源（@@"nzp"）では、変位は上下の２値に限られますが、それぞれの変位を維持する時間（パルス幅）がランダムに変更されます。

ＰＳＧ音源のノイズは、パルス幅が更新される間隔を、@nコマンドで変更することによって、ノイズの質感を変えることができます。
当設定では、パルス幅間隔の基本単位を設定することが出来ます。
この設定は、全トラックへの初期設定となります。

当設定を使用しなかった場合の初期設定は次の通りです。

```
#MB:CONFIG {
   nzp_noise: clock=3993600: prescale=32,
}
```

この初期設定は、@nコマンドによる設定の音が、ＰＣ８８モードとよく似た質感になります。

項目名(1)：clock

定義値には、仮想ＰＳＧのマスタークロックを数値で設定します。
少数以下の指定も受け付けます。
設定範囲は 1.0 ～ 10000000.0(10MHz) です。

項目名(2)：prescale

定義値には、分周比を数値で指定します。マスタークロックを割る数です。
少数以下の指定も受け付けます。
設定範囲は 1.0 ～ 65536.0 です。

【備考】
パルス幅間隔の基本単位は、clock÷prescale になります。

・機能名：pls_noise：@@"pls"用ノイズベースサイクル設定

【記述例】

```
#MB:CONFIG {
   pls_noise: clock=3993600: prescale=32: width=17,
}
```

【解説】

パルス音源（@@"pls"）には、２値ノイズモジュールも含まれており、仕組みはＰＳＧノイズ音源と同様です。
パルス音源のノイズは、変位の維持時間が更新される間隔を、@nコマンドで変更することによって、ノイズの質感を変えることができます。
当設定では、変位維持時間間隔の基本単位を設定することが出来ます。
この設定は、全トラックへの初期設定となります。

当設定を使用しなかった場合の初期設定は次の通りです。

```
#MB:CONFIG {
   pls_noise: clock=3993600: prescale=32: width=17,
}
```

この初期設定は、@nコマンドによる設定の音が、ＰＣ８８モードとよく似た質感になります。

項目名(1)：clock

定義値には、仮想ＰＳＧのマスタークロックを数値で設定します。
少数以下の指定も受け付けます。
設定範囲は 1.0 ～ 10000000.0(10MHz) です。

項目名(2)：prescale

定義値には、分周比を数値で指定します。マスタークロックを割る数です。
少数以下の指定も受け付けます。
設定範囲は 1.0 ～ 65536.0 です。

項目名(3)：width

定義値には、ノイズ生成シフトレジスタのビット幅を選ぶインデックス値を指定します。
設定範囲は 0 ～ 31 です。

【備考１】
変位維持時間間隔の基本単位は、clock÷prescale になります。

【備考２】
widthに指定するビット幅について。２値ノイズは、線形帰還シフトレジスタ(LFSR)を使って生成しています。（LFSRの詳しい説明はWikipedia参照）
ＶＳＭＭＬで使用しているビット幅は次の通りです。

```
        static inline const int NOISE_REGTAP[MAX_NZTAP] = {
//                  21098765432109876543210987654321
                  0b00000000000000000100000100000000,          //width  0: XNOR from:15,9  (FC short 93 cycle,width:4)
                  0b00000000000000001001000000000000,          //width  1: -
                  0b00000000000000001001000000000000,          //width  2: -
                  0b00000000000000001001000000000000,          //width  3: -
//                  21098765432109876543210987654321
                  0b00000000000000001001000000000000,          //width  4: -
                  0b00000000000000000000000000010100,          //width  5: XNOR from:5,3   (31 cycle)
                  0b00000000000000000000000000110000,          //width  6: XNOR from:6,5   (63 cycle)
                  0b00000000000000000000000001100000,          //width  7: XNOR from:7,6   (127 cycle)
//                  21098765432109876543210987654321
                  0b00000000000000000000000010111000,          //width  8: XNOR from:8,6,5,4  (255 cycle)
                  0b00000000000000000000000100010000,          //width  9: XNOR from:9,5   (511 cycle)
                  0b00000000000000000000001001000000,          //width 10: XNOR from:10,7  (1023 cycle)
                  0b00000000000000000000010100000000,          //width 11: XNOR from:11,9  (2047 cycle)
//                  21098765432109876543210987654321
                  0b00000000000000000000100000101001,          //width 12: XNOR from:12,6,4,1  (4095 cycle)
                  0b00000000000000000001000000001101,          //width 13: XNOR from:13,4,3,1  (8191 cycle)
                  0b00000000000000000010000000010101,          //width 14: XNOR from:14,5,3,1  (16383 cycle)
                  0b00000000000000000110000000000000,          //width 15: XNOR from:15,14 (FC long 32767 cycle)
//                  21098765432109876543210987654321
                  0b00000000000000001101000000001000,          //width 16: XNOR from:16,15,13,4  (65535 cycle)
                  0b00000000000000001001000000000000,          //width 17: XNOR from:17,14 (PSG 131071 cycle)
                  0b00000000000000010000010000000000,          //width 18: XNOR from:18,11 (262143 cycle)
                  0b00000000000001000000000000100011,          //width 19: XNOR from:19,6,2,1  (524287 cycle)
//                  21098765432109876543210987654321
                  0b00000000000010010000000000000000,          //width 20: XNOR from:20,17 (1048575 cycle)
                  0b00000000000101000000000000000000,          //width 21: XNOR from:21,19 (2097151 cycle)
                  0b00000000001100000000000000000000,          //width 22: XNOR from:22,21 (4194303 cycle)
                  0b00000000010000100000000000000000,          //width 23: XNOR from:23,18 (8388607 cycle)
//                  21098765432109876543210987654321
                  0b00000000111000010000000000000000,          //width 24: XNOR from:24,23,22,17  (16777215 cycle)
                  0b00000001001000000000000000000000,          //width 25: XNOR from:25,22 (33554431 cycle)
                  0b00000010000000000000000000100011,          //width 26: XNOR from:26,6,2,1  (67108863 cycle)
                  0b00000100000000000000000000010011,          //width 27: XNOR from:27,5,2,1  (134217727 cycle)
//                  21098765432109876543210987654321
                  0b00001001000000000000000000000000,          //width 28: XNOR from:28,25 (268435455 cycle)
                  0b00010100000000000000000000000000,          //width 29: XNOR from:29,27 (536870911 cycle)
                  0b00100000000000000000000000101001,          //width 30: XNOR from:30,6,4,1  (1073741823 cycle)
                  0b01001000000000000000000000000000,          //width 31: XNOR from:31,28 (2147483647 cycle)
        };
        static inline const int NOISE_WIDTH[MAX_NZTAP] = {
                  4,17,17,17, 17, 5, 6, 7,  8, 9,10,11, 12,13,14,15,
                 16,17,18,19, 20,21,22,23, 24,25,26,27, 28,29,30,31,
        };
```

概ねインデックス値とビット幅は一致していますが、インデックス値０は特例でＦＣノイズのショートモードになっています。インデックス値１～４は未使用扱いです。

・機能名：nzc_noise：@@"nzc"用の変位更新間隔の倍率調整

【記述例】

```
#MB:CONFIG {
   nzc_noise: step_adj=0.5,
}
```

波形メモリノイズ音源の、変位更新間隔の倍率が、
通常（1.0）の場合と比べて、同じノイズサイクル指定（@nコマンド）でも、
半分（0.5）のノイズサイクルになります。

【解説】

波形メモリノイズ音源（@@"nzc"）の、変位更新間隔（ノイズサイクル）を、
初期設定の何倍にするかを設定します。
当設定は、波形メモリノイズ音源への @nコマンドによるノイズサイクル指定の
結果に影響を与えます。
この設定は、全トラックへの初期設定となります。

当設定を使用しなかった場合の初期設定は次の通りです。

```
#MB:CONFIG {
   nzc_noise: step_adj=1.0,
}
```

項目名(1)：step_adj

定義値には、変位更新間隔の倍率を数値で指定します。
設定範囲は 0.0625 ～ 16.0 です。

5.9. #ML:INCLUDE：外部テキスト読み込み機能

【記述例】

```
#ML:INCLUDE test.txt
```

【解説】

指定されたファイル名のテキストファイルを読み込んで、現在のテキストの先頭に追加します。

複数の「#ML:INCLUDE」指定があった場合は、指定された順番に連結されて、連結された結果が、現在のテキストの先頭に追加されます。

この機能は、他のメタライン（#ML:...）、メタブロック（#MB...{...}）の解釈よりも前に処理されます。

外部テキストからメタデータ定義を読み込んで使う用途を想定しています。
例えば、外部に共通利用目的のＦＭ音源音色テキストを用意しておいて、読み込んで使う、などです。

用意する外部テキストの場所は、PlayListウィンドウのカレントフォルダ以下の場所である必要があります。相対パスで親「..」に遡ることは出来ない作りなので注意してください。

5.10. #ML:REPORT_TICKS：総TICKカウント数の表示

【記述例】

```
#ML:REPORT_TICKS
```

【解説】

定義済みトラックの tickカウント合計情報を表示します。
メタデータとして、

```
#ML:REPORT_TICKS
```

と記述すると、集計された tickカウント合計情報を、Warningウィンドウに表示します。

表示内容は次の通りです。
・演奏tickカウント数の合計。
・無限リピート時の、リピートエントリ時点の tickカウント数。
・無限リピート時の、リピート対象部分の tickカウント数の合計。
・無限リピートするトラックについては、無限リピートエントリとトラック終端におけるコマンド状態の差分表示。
以上が、定義済みトラックごとに表示されます。

5.11.！：ＭＭＬステータス表示関係

【解説】
ＭＭＬ編集中のデバッグ表示として、各種情報を Warningウィンドウ に表示します。

- !n[1]：音符ログ

【記述例】

```
!n1 cdefgr
!n0 gfedcr;
```

この場合、前半のcdefgrの音符部分のログのみが表示されます。

【解説】
ＭＭＬ編集中のデバッグ表示として、音符の履歴情報（ログ）を、Warningウィンドウに表示します。
引数[1]は、０または１で指定します。
指定値が１のとき、音符ログの開始します。
指定値が０のとき、音符ログの停止します。

・!r[1]：休符ログ

【記述例】

```
!r1 cdefgr
!r0 gfedcr;
```

この場合、前半のcdefgrの休符部分のログのみが表示されます。

【解説】

ＭＭＬ編集中のデバッグ表示として、休符の履歴情報（ログ）を、Warningウィンドウに表示します。
引数[1]は、０または１で指定します。
指定値が１のとき、休符ログの開始します。
指定値が０のとき、休符ログの停止します。

・!s[str]：現時点のステータス表示

【記述例】

```
t70 l4 @q3 v15
a(2 a(2
!s"stat-a" a(2 a(2
!s"stat-b" a(2 a(2
!s"stat-c" a(2 !s
```

どの時点かの判別を容易にするため、ラベル"stat-a"、"stat-b"、"stat-c"を使っています。最後の「!s」記述のように、ラベル無しでも受け付けます。

【解説】

!s[str]コマンド記述時点での、各種ＭＭＬコマンド発行状態が、Warningウィンドウに表示されます。
引数[str]はダブルクォーテーションで括った文字列を指定します。文字列は単なるラベルですので内容は自由ですが、半角文字のみで記述してください。
引数を省略する場合は「!s」とだけ記述します。

・!c[1],[str]：回数条件によるステータス表示

【記述例】

```
t70 l4 @q3 v15 a(2 a(2 a(2 a(2 a(2 a(2 a(2
!c3,"ge";
```

!c3,"ge"によって、その時点で３回以上指定されているコマンドのステータスを表示します。

【解説】

!c[1],[str]コマンド記述時点での、その時点で指定条件に合うＭＭＬコマンドの発行状態が、Warningウィンドウに表示されます。
引数[1]はＭＭＬコマンドの指定回数、引数[str]はダブルクォーテーションで括った文字列で、条件を指定します。

【!c[1],[str]の条件文字とその内容】

条件文字	gt	ge	lt	le	eq	ne
内容	より多い	以上	より少ない	以下	等しい	等しくない

- !m[1]：ステータスを指定メモリ番号へ記憶
- !d[1],[2]：記憶したステータス同士の差分を表示

【記述例】

```
t70 l4 @q3 v15 !m1 a(2 a(2 a(2 !m2 a(2 a(2 a(2 a(2
!d1,2;
```

!m1 と !m2 によってメモリ番号１番と２番にその場所のステータスを記憶。
!d1,2 によって、メモリ番号１番と２番の状態比較をし、差分を表示。

【解説】

ステータスメモリコマンド（!m[1]）：
!m[1]コマンド記述時点でのステータスを記憶します。
引数[1]は記憶するメモリ番号です。0 ～ 9 までの整数で指定します。

ステータスdiffコマンド（!d[1],[2]）：
記憶済みメモリ番号同士でステータス内容を比較し、差分を表示します。
引数[1]と引数[2]はメモリ番号です。双方 0 ～ 9 までの整数で指定します。

- ![str]：注意喚起の通告を表示

【記述例】

```
t70 l4 @q3 v15 !"start" a(2 a(2 a(2 a(2 a(2 a(2 a(2
!"end";
```

【解説】

![str]コマンド記述時点での通告が、Warningウィンドウに表示されます。
引数[str]はダブルクォーテーションで括った文字列で、通告内容を半角文字で記述します。
主な用途は、後で修正を促す内容のコメントを設定しておく、などです。

・！：ステータス表示関係機能の簡易ヘルプ表示

【記述例】

```
t70 l4 @q3 v15 ! a(2 a(2 a(2 a(2 a(2 a(2 a(2;
```

【解説】

ステータス表示関係コマンドの概要を簡易的に Warningウィンドウ に表示します。

5.12. @dsp[1],[2],[3]：KeyDispゲージへの表示指示

【記述例】

```
@dsp2,13,15 cr
@dsp2,10 cr
```

左から２番目のゲージ表示を、全体の 13/15 の大きさで表示し、その後 10/15 の大きさで表示します。

```
@dsp2,-1
```

左から２番目のゲージ表示の解除を行い、本来の表示に戻します。

【解説】

KeyDispの表示領域における、各トラックの音量関連ゲージ、左から順に
　　ボリューム（v）ゲージ
　　エンベロープ（@ea）ゲージ
　　ボリュームＬ（vl）ゲージ
　　音量ＬＦＯ（@la）ゲージ
に対して、強制的に上書きで表示指示を行います。
これは表示のみで、音には全く影響を及ぼしません。

この機能の用途は、例えば、yコマンドでＦＭ音源モジュールのＴＬを操作した後、手動で音量ゲージを表示させる手段として利用することです。

指定する引数は次の通りです。
引数[1]...ゲージ表示の宛先（dest）
引数[2]...ゲージ表示レベルの分子（num）
引数[3]...ゲージ表示レベルの分母（denom）
引数はカンマで区切って指定します。

引数[1]（dest）

ゲージ表示の宛先を、KeyDisp表示における左から何番目のゲージにするかを数値で指定します。
設定範囲は 1 ～ 4 です。

引数[2]（num）

ゲージ表示させたい大きさの分子を指定します。
ゲージ表示させるには、分子÷分母の結果を 0〜1 の範囲に収めてください。
ゲージ表示を解除するには、分子÷分母が負数になるよう指定してください。
（解除になると、通常通りの表示を行います）

引数[3]（denom）

ゲージ表示させたい大きさの分母を指定します。
分母値に 0 は指定できません。
１度分母を指定すると、以降の@dspコマンドで分母を省略した時の分母値に採用されます。分母値の初期設定は 1 です。

5.13. $名前=内容;：マクロ定義

【解説】
マクロ機能は、頻繁に使うＭＭＬコマンド群に「名前」をつける機能です。
一度「名前」つければ（マクロ定義）、後はその名前を書くだけ（マクロ参照）でＭＭＬコマンド群を呼び出せるので、記述が楽になります。

マクロの定義方法：
トラック先頭（ＭＭＬコマンドを１つも記述してない状態のトラック）で、

```
$名前=内容;
```

このように、「$」で始めてマクロ名を書き、次に「=」を書いてからＭＭＬコマンド群を書いて、最後に「;」で締めくくることでマクロを定義します。
マクロの内容には改行を含んでも良いので、長いフレーズの定義も可能です。
また、トラック先頭であれば、複数のマクロを定義できます。

すでに定義されたマクロを、以降のマクロ定義で参照するような、入れ子的なマクロ定義も可能です。

マクロ定義はセミコロン「;」で締め括りますが、トラック定義とは別物と考えてください。トラック先頭において「$」で始まる部分は、マクロ定義モードと解釈されます。しかし、マクロ定義後に、１つでもＭＭＬコマンドを記述すると、トラック定義モードとなり、「$」は定義済みマクロの参照開始とみなされますので、ＭＭＬ記述中にマクロを定義することは出来ません。

マクロの名前の付け方には次の制限があります。
・名前の１文字目はアルファベット又はアンダーライン(_)。
・名前の２文字目以降は、
　　アルファベット、アンダーライン(_)、
　　数字、４種の記号【 + # ( ) 】
　のいずれかです。

マクロの参照方法：
マクロ定義されたトラック以降で、

```
$名前
```

のように記述します。こうすると、「$名前」が「内容」に置き換えられます。
この置き換えは、プリプロセッサにより行われます。

【マクロの定義と参照の例】

トラック先頭の状態で、次のように書いた場合、

```
$C=ecgc;
$C $C $C;
```

上記のトラック定義は、

```
ecgc ecgc ecgc;
```

と同じことになります。
ＭＭＬコンパイル時、プリプロセッサにより、マクロ参照部分がマクロ定義内容に置き換えられています。
マクロ定義部分は、プリプロセッサによる置き換え処理の際に、除去される格好です。

入れ子的にマクロを利用する場合は、

```
$C=ecgc;
$D=fdad;
$M= $C $D;
t150 l16 o5 $M;
```

といった感じになります。

マクロ機能は単純な文字列の置き換えなので、ちょっと変わった使い方もできます。例えば、次のように書いた場合、

```
$C=cd;
l16 $C2$C4$C8;
```

上記のトラック定義は、

```
l16 cd2cd4cd8;
```

と同じこととなり、dの音長だけを変えて記述しています。

マクロの定義は、トラック定義の先頭部分に集まっていなければならない制限があります。

使用できる記述例：

```
$T1=@5 v10;
$T2=@8 v12;
$T1 cgeg $T2 cgeg $T1 egcgc;
$T3=@10 v7;
$T4=@11 v12;
$T1 cccc $T3 cccc $T4 ccccc;
```

上記では、マクロの定義が１トラック目と２トラック目の先頭部分に集められていて、２トラック目では、それまでに定義した全てのマクロが参照できることが確認できます。

エラーになる例：

```
@0 cgeg
$T1=@5 v10;
$T1 egcgc;
```

上記では、ＭＭＬ定義が始まってからマクロ定義をしようとしており、エラーになります。
これは、ＭＭＬ記述が始まった時点で、そのトラックでは「$」文字がマクロ定義開始ではなく、マクロ参照開始と認識されるからです。
ＭＭＬ記述が始まると、セミコロンもマクロ内容終端ではなく、ＭＭＬトラック終端として認識されます。

・引数つきマクロ定義

引数つきマクロの定義方法：

```
$名前{引数1,引数2,...}=内容;
```

このように、マクロ名につづけて中カッコを書き、その中に「%」で始まる引数名を書いて、「=」後に内容を書くと、引数つきマクロが定義できます。
引数名の書式は、マクロ名と同じです。
そして、内容の定義内で、引数を使いたいところ（引数参照）では、
「%引数名」と書くことで引数が参照できます。

引数つきマクロの参照方法：

```
$名前{引数1,引数2...}
```

このように、マクロ名につづけて中カッコを書き、その中に引数を書くことで、引数つきマクロが参照できます。

【引数つきマクロの定義と参照の例】

トラック先頭の状態で、次のように書いた場合、

```
$m{%note}=v13 %note16 v8 %note16 r8;
$m{c} $m{d} $m{e} $m{f} $m{g};
```

上記のトラック定義は、

```
v13 c16 v8 c16 r8 v13 d16 v8 d16 r8 v13 e16 v8 e16 r8 v13 f16 v8 f16 r8
v13 g16 v8 g16 r8;
```

と同じこととなります。同じマクロを %note の部分を変更しながら使い回すことができます。

引数は複数指定することもできます。

```
$m{%vol,%note}=v%vol %note;
$m{13,cde} $m{10,efg};
```

上記のトラック定義は、

```
v13 cde v10 efg;
```

と同じこととなります。

6.音色関連

6.1. @@[str]：使用音源モジュール指定

【記述例】

```
@@"sin"
```

【解説】

使用する音源モジュールを指定します。
引数[str]には、選択したい音源モジュールを指す文字列を指定します。
トラック先頭における初期設定は、@@"pls"（パルス音源）です。

音源モジュールを指す文字列一覧

sin	サイン波音源
saw	ノコギリ波音源
tri	三角波音源
pls	パルス波音源
cpx	複合波（@0:sin/@1:saw/@2:tri/@3:pls）音源
wvm	波形メモリ音源
fms	ＦＭ音源
nzw	ホワイトノイズ音源
nzp	ＰＳＧノイズ音源
nzf	ＦＣノイズ音源
nzg	ＧＢノイズ音源
nzc	波形メモリノイズ音源
pcm	ＰＣＭ音源

6.2. @[1]：サブフォーム番号指定

【記述例】

```
@@"fms" @18 efg @7 gab;
```

音源モジュールをＦＭ音源に設定して、
サブフォーム番号（音色番号）18 でefgを演奏し、
その後サブフォーム番号 7 でgabを演奏します。

【解説】

当コマンド指定時、既に選択されている音源モジュールに対し、
サブフォーム番号の変更を指示します。

引数[1]には、0 以上の整数を指定します。
トラック先頭における初期設定は、@0 です。

6.3. @@"sin"：サイン波音源

【記述例】

```
@@"sin"
```

【解説】
使用する音源モジュールを、サイン波音源にします。

【yコマンドにおける、音源モジュールへの設定】

y"subform",[1]：
全てのオペレータの音源サブフォーム番号を、[1] に設定します。
サイン波音源では、指定できるサブフォーム番号は０のみです。

y"opnum",[1]：
オペレータモードを [1] に設定します。
設定範囲は １ ～ ９ です。１ならば１声、９ならば９声で発声します。

設定値	1	2	3	4	5	6	7	8	9
有効OP	OP1	OP1-OP2	OP1-OP3	OP1-OP4	OP1-OP5	OP1-OP6	OP1-OP7	OP1-OP8	OP1-OP9

y"clipmode",[1]：
クリップモードの値を [1] に設定します。
0 の場合、クリップモード無効（初期設定）
1 の場合、クリップモード有効
クリップモードとは、全てのオペレータの発音を重ねた結果、
下限-1.0～上限+1.0の範囲の振幅から はみ出す部分を、下限と上限に制限する機能です。用途は、ディストーション効果を得ることです。

y"ampdenom",[1]：
音量比のオートバランサー機能の値を [1] に設定します。
0 の場合、オートバランサー無効
1 の場合、オートバランサー有効（初期設定）
音量比のオートバランサーは、マルチオペレータモードにおける音割れ防止が目的の機能です。
オートバランサーによる合計振幅への倍率計算は、
　　倍率＝全有効オペレータの音量倍率の合計の逆数

により求められます。

y"detune",[1]：

マルチオペレータモード時の音程比を、コーラスモード [1] cent に設定します（初期設定は 8、設定範囲は -100～100）。

【sin:コーラスモードの音程比】

OP番号	OP1	OP2	OP3	OP4	OP5	OP6	OP7	OP8	OP9
centずれ	0	[1]	-[1]	[1]/2	-[1]/2	[1]/4	-[1]/4	[1]/8	-[1]/8

その他のオペレータごとの設定：

項目名	挙動
y"***subform",[1]	オペレータの音源サブフォーム番号を、 [1] に設定します。
y"***phase",[1]	オペレータの位相を、[1] に設定します。 設定範囲は 0.0 ～ 1.0未満です。 -1 を指定した場合は、0.0 ～ 1.0未満の範囲の値がランダムで設定されます。
y"***dB",[1]	オペレータの音量比を、[1] dBに設定します。 設定範囲は -120.0 ～ 48.0 です。
y"***mul",[1]	オペレータの周波数倍率を、[1] に設定します。 設定範囲は 0.0 以上です。
y"***cent",[1]	オペレータの周波数比を、[1] centに設定します。 設定範囲は、-9600.0 ～ 9600.0 です。
y"***freq",[1]	オペレータの周波数への加算値を、 [1] Hzに設定します。

「***」に当てはまる文字が「all」の場合、全てのオペレータが対象
「***」に当てはまる文字が「op1」の場合、オペレータ１のみが対象
「***」に当てはまる文字が「op2」の場合、オペレータ２のみが対象
「***」に当てはまる文字が「op3」の場合、オペレータ３のみが対象
「***」に当てはまる文字が「op4」の場合、オペレータ４のみが対象
「***」に当てはまる文字が「op5」の場合、オペレータ５のみが対象
「***」に当てはまる文字が「op6」の場合、オペレータ６のみが対象
「***」に当てはまる文字が「op7」の場合、オペレータ７のみが対象
「***」に当てはまる文字が「op8」の場合、オペレータ８のみが対象
「***」に当てはまる文字が「op9」の場合、オペレータ９のみが対象

6.4. @@"saw"：ノコギリ波音源

【記述例】

```
@@"saw"
```

【解説】

使用する音源モジュールを、ノコギリ波音源にします。

【yコマンドにおける、音源モジュールへの設定】

y"subform",[1]：
全てのオペレータの音源サブフォーム番号を、[1] に設定します。
ノコギリ波音源では、指定できるサブフォーム番号は０のみです。

y"opnum",[1]：
オペレータモードを [1] に設定します。
設定範囲は １ ～ ９ です。１ならば１声、９ならば９声で発声します。

設定値	1	2	3	4	5	6	7	8	9
有効OP	OP1	OP1-OP2	OP1-OP3	OP1-OP4	OP1-OP5	OP1-OP6	OP1-OP7	OP1-OP8	OP1-OP9

y"clipmode",[1]：
クリップモードの値を [1] に設定します。
0 の場合、クリップモード無効（初期設定）
1 の場合、クリップモード有効
クリップモードとは、全てのオペレータの発音を重ねた結果、
下限-1.0～上限+1.0の範囲の振幅から はみ出す部分を、下限と上限に制限する機能です。用途は、ディストーション効果を得ることです。

y"ampdenom",[1]：
音量比のオートバランサー機能の値を [1] に設定します。
0 の場合、オートバランサー無効
1 の場合、オートバランサー有効（初期設定）
音量比のオートバランサーは、マルチオペレータモードにおける音割れ防止が目的の機能です。
オートバランサーによる合計振幅への倍率計算は、
　　倍率＝全有効オペレータの音量倍率の合計の逆数

により求められます。

y"detune",[1]：

マルチオペレータモード時の音程比を、コーラスモード [1] cent に設定します（初期設定は 8、設定範囲は -100～100）。

【saw:コーラスモードの音程比】

OP番号	OP1	OP2	OP3	OP4	OP5	OP6	OP7	OP8	OP9
centずれ	0	[1]	-[1]	[1]/2	-[1]/2	[1]/4	-[1]/4	[1]/8	-[1]/8

その他のオペレータごとの設定：

項目名	挙動
y"***subform",[1]	オペレータの音源サブフォーム番号を、 [1] に設定します。
y"***phase",[1]	オペレータの位相を、[1] に設定します。 設定範囲は 0.0 ～ 1.0未満です。 -1 を指定した場合は、0.0 ～ 1.0未満の範囲の値がランダムで設定されます。
y"***dB",[1]	オペレータの音量比を、[1] dBに設定します。 設定範囲は -120.0 ～ 48.0 です。
y"***mul",[1]	オペレータの周波数倍率を、[1] に設定します。 設定範囲は 0.0 以上です。
y"***cent",[1]	オペレータの周波数比を、[1] centに設定します。 設定範囲は、-9600.0 ～ 9600.0 です。
y"***freq",[1]	オペレータの周波数への加算値を、 [1] Hzに設定します。

「***」に当てはまる文字が「all」の場合、全てのオペレータが対象
「***」に当てはまる文字が「op1」の場合、オペレータ１のみが対象
「***」に当てはまる文字が「op2」の場合、オペレータ２のみが対象
「***」に当てはまる文字が「op3」の場合、オペレータ３のみが対象
「***」に当てはまる文字が「op4」の場合、オペレータ４のみが対象
「***」に当てはまる文字が「op5」の場合、オペレータ５のみが対象
「***」に当てはまる文字が「op6」の場合、オペレータ６のみが対象
「***」に当てはまる文字が「op7」の場合、オペレータ７のみが対象
「***」に当てはまる文字が「op8」の場合、オペレータ８のみが対象
「***」に当てはまる文字が「op9」の場合、オペレータ９のみが対象

6.5. @@"tri"：三角波音源

【記述例】

```
@@"tri"
```

【解説】

使用する音源モジュールを、三角波音源にします。

【yコマンドにおける、音源モジュールへの設定】

y"subform",[1]：

全てのオペレータの音源サブフォーム番号を、[1] に設定します。
三角波音源では、指定できるサブフォーム番号は０のみです。

y"opnum",[1]：

オペレータモードを [1] に設定します。
設定範囲は １ ～ ９ です。１ならば１声、９ならば９声で発声します。

設定値	1	2	3	4	5	6	7	8	9
有効OP	OP1	OP1-OP2	OP1-OP3	OP1-OP4	OP1-OP5	OP1-OP6	OP1-OP7	OP1-OP8	OP1-OP9

y"clipmode",[1]：

クリップモードの値を [1] に設定します。
0 の場合、クリップモード無効（初期設定）
1 の場合、クリップモード有効
クリップモードとは、全てのオペレータの発音を重ねた結果、
下限-1.0～上限+1.0の範囲の振幅から はみ出す部分を、下限と上限に制限する機能です。用途は、ディストーション効果を得ることです。

y"ampdenom",[1]：

音量比のオートバランサー機能の値を [1] に設定します。
0 の場合、オートバランサー無効
1 の場合、オートバランサー有効（初期設定）
音量比のオートバランサーは、マルチオペレータモードにおける音割れ防止が目的の機能です。
オートバランサーによる合計振幅への倍率計算は、
　　倍率＝全有効オペレータの音量倍率の合計の逆数

により求められます。

y"detune",[1]：

マルチオペレータモード時の音程比を、コーラスモード [1] cent に設定します（初期設定は 8、設定範囲は -100～100）。

【tri:コーラスモードの音程比】

OP番号	OP1	OP2	OP3	OP4	OP5	OP6	OP7	OP8	OP9
centずれ	0	[1]	-[1]	[1]/2	-[1]/2	[1]/4	-[1]/4	[1]/8	-[1]/8

その他のオペレータごとの設定：

項目名	挙動
y"***subform",[1]	オペレータの音源サブフォーム番号を、 [1] に設定します。
y"***phase",[1]	オペレータの位相を、[1] に設定します。 設定範囲は 0.0 ～ 1.0未満です。 -1 を指定した場合は、0.0 ～ 1.0未満の範囲の値がランダムで設定されます。
y"***dB",[1]	オペレータの音量比を、[1] dBに設定します。 設定範囲は -120.0 ～ 48.0 です。
y"***mul",[1]	オペレータの周波数倍率を、[1] に設定します。 設定範囲は 0.0 以上です。
y"***cent",[1]	オペレータの周波数比を、[1] centに設定します。 設定範囲は、-9600.0 ～ 9600.0 です。
y"***freq",[1]	オペレータの周波数への加算値を、 [1] Hzに設定します。

「***」に当てはまる文字が「all」の場合、全てのオペレータが対象
「***」に当てはまる文字が「op1」の場合、オペレータ１のみが対象
「***」に当てはまる文字が「op2」の場合、オペレータ２のみが対象
「***」に当てはまる文字が「op3」の場合、オペレータ３のみが対象
「***」に当てはまる文字が「op4」の場合、オペレータ４のみが対象
「***」に当てはまる文字が「op5」の場合、オペレータ５のみが対象
「***」に当てはまる文字が「op6」の場合、オペレータ６のみが対象
「***」に当てはまる文字が「op7」の場合、オペレータ７のみが対象
「***」に当てはまる文字が「op8」の場合、オペレータ８のみが対象
「***」に当てはまる文字が「op9」の場合、オペレータ９のみが対象

6.6. @@"pls"：パルス波音源

【記述例】

```
@@"pls"
```

【解説】

使用する音源モジュールを、パルス波音源にします。

【yコマンドにおける、音源モジュールへの設定】

y"subform",[1]：

全てのオペレータの音源サブフォーム番号を、[1] に設定します。
設定範囲は 0 ～ 2 です。
0 の場合、オシレータ波形がパルス波になります。
1 の場合、オシレータ波形が２値ノイズになります。
2 の場合、オシレータ波形がパルス波と２値ノイズのミックスになります。

y"opnum",[1]：

オペレータモードを [1] に設定します。
設定範囲は 1 ～ 9 です。１ならば１声、９ならば９声で発声します。

設定値	1	2	3	4	5	6	7	8	9
有効OP	OP1	OP1-OP2	OP1-OP3	OP1-OP4	OP1-OP5	OP1-OP6	OP1-OP7	OP1-OP8	OP1-OP9

y"clipmode",[1]：

クリップモードの値を [1] に設定します。
0 の場合、クリップモード無効（初期設定）
1 の場合、クリップモード有効
クリップモードとは、全てのオペレータの発音を重ねた結果、
下限-1.0～上限+1.0の範囲の振幅から はみ出す部分を、下限と上限に制限する機能です。用途は、ディストーション効果を得ることです。

y"ampdenom",[1]：

音量比のオートバランサー機能の値を [1] に設定します。
0 の場合、オートバランサー無効
1 の場合、オートバランサー有効（初期設定）

音量比のオートバランサーは、マルチオペレータモードにおける音割れ防止が目的の機能です。
オートバランサーによる合計振幅への倍率計算は、
　　倍率＝全有効オペレータの音量倍率の合計の逆数
により求められます。

y"detune",[1]：
マルチオペレータモード時の音程比を、コーラスモード [1] cent に設定します（初期設定は 8、設定範囲は -100～100）。

【pls:コーラスモードの音程比】

OP番号	OP1	OP2	OP3	OP4	OP5	OP6	OP7	OP8	OP9
centずれ	0	[1]	-[1]	[1]/2	-[1]/2	[1]/4	-[1]/4	[1]/8	-[1]/8

y"pwm",[1]：
全てのオペレータのデューティ比を、[1] に設定します。
設定範囲は 0.005 ～ 0.995 です。

y"setfrqtype",[1]：
効果音対策用の周波数設定機能の有効無効を数値で [1] に設定します。
（当音源宛の @pfコマンドと同様です）
初期設定は -1 です。
設定値が -1 の時、効果音対策用の周波数設定機能が無効になります。
設定値が　0 ～ 100 の時、効果音対策用の周波数設定機能が有効になります。
そしてその設定値は分周比の内部カウンタを微調整として何回分進めるかの整数として扱われます。通常は設定値に 1 を指定します。100以上の値は100として扱われます。

y"nz_sreg",[1]：
全てのオペレータの２値ノイズ用シフトレジスタの値を [1] に設定します。
（当音源宛の @nrコマンドと同様です）
次の場合によって指定された値の使われ方が変わります。
※ノイズ用シフトレジスタのビット幅の設定が bitw だったとします
（１）０以上、１未満の値
（２）１以上、(2^bitw)-1 未満の値
（３）上記（１）（２）以外の値

（１）の場合、指定値に (2^bitw)-1 を乗算し、結果が１未満なら１、(2^bitw)-2 以上なら (2^bitw)-2 に制限してシフトレジスタに設定されます。
（２）の場合、値が整数に四捨五入されてシフトレジスタに設定されます。
（３）の場合（負の値など）、０以上１未満のランダム値を作成し、（１）同様の処理をします。

y"nz_cycle",[1]：
全てのオペレータの２値ノイズ更新サイクル（変位更新間隔）を、
[1] に設定します。（当音源宛の @nコマンドと同様です）
設定範囲は 1 ～ clk です。
※clk = ノイズ用マスタークロック値

y"nz_mclock",[1]：
全てのオペレータの２値ノイズのマスタークロックを、[1] に設定します。
設定範囲は 1 ～ 10000000（10MHz）です。

y"nz_pscale",[1]：
全てのオペレータの２値ノイズのプリスケールを、[1] に設定します。
（プリスケールは、マスタークロックを割る数です）
設定範囲は 1 ～ 65536 です。

y"nz_bwidth",[1]：
全てのオペレータの２値ノイズ用シフトレジスタのビット幅を選ぶインデックス値を [1] に設定します。
設定範囲は 0 ～ 31 です。
ビット幅の説明についてはこちら

y"nz_clkmode",[1]：
全てのオペレータの２値ノイズのクロックモードを、[1] に設定します。
設定範囲は 0、1、2、3、10 のうちのいずれかです。
0 の場合、PC88モード（clock=3993600, prescale=32, width=17）
1 の場合、 X1 モード（clock=4000000, prescale=32, width=17）
2 の場合、MSX モード（clock=3579545, prescale=32, width=17）
3 の場合、FM-7モード（clock=1228800, prescale=16, width=17）
10の場合、オリジナルモード（clock=4000000, prescale=32, width=31）

その他のオペレータごとの設定：

項目名	挙動
y"***subform",[1]	オペレータの音源サブフォーム番号を、 [1] に設定します。
y"***phase",[1]	オペレータの位相を、[1] に設定します。 設定範囲は 0.0 ～ 1.0未満です。 -1 を指定した場合は、0.0 ～ 1.0未満の範囲の値がランダムで設定されます。
y"***dB",[1]	オペレータの音量比を、[1] dBに設定します。 設定範囲は -120.0 ～ 48.0 です。

y"***mul",[1]	オペレータの周波数倍率を、[1] に設定します。 設定範囲は 0.0 以上です。
y"***cent",[1]	オペレータの周波数比を、[1] centに設定します。 設定範囲は、-9600.0 ～ 9600.0 です。
y"***freq",[1]	オペレータの周波数への加算値を、 [1] Hzに設定します。
y"***pwm",[1]	オペレータのデューティ比を、[1] に設定します。 設定範囲は 0.005 ～ 0.995 です。
y"***nz_sreg",[1]	オペレータの２値ノイズ用シフトレジスタの値を、 [1] に設定します。 設定範囲は、y"nz_sreg",[1] と同様です。
y"***nz_cycle",[1]	全てのオペレータの２値ノイズ更新サイクル（変位更新間隔）を、[1] に設定します。 設定範囲は、y"nz_cycle",[1] と同様です。
y"***nz_mclock",[1]	全てのオペレータの２値ノイズのマスタークロックを、[1] に設定します。 設定範囲は 1 ～ 10000000（10MHz）です。
y"***nz_pscale",[1]	全てのオペレータの２値ノイズのプリスケールを、[1] に設定します。 設定範囲は 1 ～ 65536 です。
y"***nz_bwidth",[1]	全てのオペレータの２値ノイズ用シフトレジスタのビット幅を選ぶインデックス値を [1] に設定します。 設定範囲は 0 ～ 31 です。

「***」に当てはまる文字が「all」の場合、全てのオペレータが対象
「***」に当てはまる文字が「op1」の場合、オペレータ１のみが対象
「***」に当てはまる文字が「op2」の場合、オペレータ２のみが対象
「***」に当てはまる文字が「op3」の場合、オペレータ３のみが対象
「***」に当てはまる文字が「op4」の場合、オペレータ４のみが対象
「***」に当てはまる文字が「op5」の場合、オペレータ５のみが対象
「***」に当てはまる文字が「op6」の場合、オペレータ６のみが対象
「***」に当てはまる文字が「op7」の場合、オペレータ７のみが対象
「***」に当てはまる文字が「op8」の場合、オペレータ８のみが対象
「***」に当てはまる文字が「op9」の場合、オペレータ９のみが対象

6.7. @@"cpx"：複合波音源

【記述例】

```
@@"cpx"
```

【解説】
使用する音源モジュールを、複合波音源にします。

【yコマンドにおける、音源モジュールへの設定】

y"subform",[1]：
全てのオペレータの音源サブフォーム番号を、[1] に設定します。
複合波音源で指定できるサブフォーム番号は 0 ～ 3 です。
0 の場合、正弦波になります。
1 の場合、ノコギリ波になります。
2 の場合、三角波になります。
3 の場合、パルス波になります。

y"opnum",[1]：
オペレータモードを [1] に設定します。
設定範囲は 1 ～ 9 です。１ならば１声、９ならば９声で発声します。

設定値	1	2	3	4	5	6	7	8	9
有効OP	OP1	OP1-OP2	OP1-OP3	OP1-OP4	OP1-OP5	OP1-OP6	OP1-OP7	OP1-OP8	OP1-OP9

y"clipmode",[1]：
クリップモードの値を [1] に設定します。
0 の場合、クリップモード無効（初期設定）
1 の場合、クリップモード有効
クリップモードとは、全てのオペレータの発音を重ねた結果、
下限-1.0～上限+1.0の範囲の振幅から はみ出す部分を、下限と上限に制限する機能です。用途は、ディストーション効果を得ることです。

y"ampdenom",[1]：
音量比のオートバランサー機能の値を [1] に設定します。
0 の場合、オートバランサー無効
1 の場合、オートバランサー有効（初期設定）

音量比のオートバランサーは、マルチオペレータモードにおける音割れ防止が目的の機能です。
オートバランサーによる合計振幅への倍率計算は、
　　倍率＝全有効オペレータの音量倍率の合計の逆数
により求められます。

y"detune",[1]：

マルチオペレータモード時の音程比を、コーラスモード [1] cent に設定します（初期設定は 8、設定範囲は -100～100）。

【cpx:コーラスモードの音程比】

OP番号	OP1	OP2	OP3	OP4	OP5	OP6	OP7	OP8	OP9
centずれ	0	[1]	-[1]	[1]/2	-[1]/2	[1]/4	-[1]/4	[1]/8	-[1]/8

y"pwm",[1]：

全てのオペレータのデューティ比を、[1] に設定します。
設定範囲は 0.005 ～ 0.995 です。

y"setfrqtype",[1]：

効果音対策用の周波数設定機能の有効無効を数値で [1] に設定します。
初期設定は -1 です。
設定値が -1 の時、効果音対策用の周波数設定機能が無効になります。
設定値が　0 以上の時、効果音対策用の周波数設定機能が有効になります。そしてその設定値は分周比の内部カウンタを微調整として何回分進めるかの整数として扱われます。通常は設定値に 1 を指定します。

その他のオペレータごとの設定：

項目名	挙動
y"***subform",[1]	オペレータの音源サブフォーム番号を、 [1] に設定します。
y"***phase",[1]	オペレータの位相を、[1] に設定します。 設定範囲は 0.0 ～ 1.0未満です。 -1 を指定した場合は、0.0 ～ 1.0未満の範囲の値がランダムで設定されます。
y"***dB",[1]	オペレータの音量比を、[1] dBに設定します。 設定範囲は -120.0 ～ 48.0 です。
y"***mul",[1]	オペレータの周波数倍率を、[1] に設定します。 設定範囲は 0.0 以上です。
y"***cent",[1]	オペレータの周波数比を、[1] centに設定します。 設定範囲は、-9600.0 ～ 9600.0 です。

y"***freq",[1]	オペレータの周波数への加算値を、 [1] Hzに設定します。
y"***pwm",[1]	オペレータのデューティ比を、[1] に設定します。 設定範囲は 0.005 ～ 0.995 です。

「***」に当てはまる文字が「all」の場合、全てのオペレータが対象
「***」に当てはまる文字が「op1」の場合、オペレータ１のみが対象
「***」に当てはまる文字が「op2」の場合、オペレータ２のみが対象
「***」に当てはまる文字が「op3」の場合、オペレータ３のみが対象
「***」に当てはまる文字が「op4」の場合、オペレータ４のみが対象
「***」に当てはまる文字が「op5」の場合、オペレータ５のみが対象
「***」に当てはまる文字が「op6」の場合、オペレータ６のみが対象
「***」に当てはまる文字が「op7」の場合、オペレータ７のみが対象
「***」に当てはまる文字が「op8」の場合、オペレータ８のみが対象
「***」に当てはまる文字が「op9」の場合、オペレータ９のみが対象

6.8. @@"wvm"：波形メモリ音源

【記述例】

```
@@"wvm"
```

【解説】

使用する音源モジュールを、波形メモリ音源にします。

【yコマンドにおける、音源モジュールへの設定】

y"subform",[1]：

全てのオペレータの音源サブフォーム番号を、[1] に設定します。
設定範囲は、0 ～ 1023 です。
ただし、メタデータ定義（#MB:WVM_WAVE）にて波形データを定義した番号に限られます。

y"opnum",[1]：

オペレータモードを [1] に設定します。
設定範囲は 1 ～ 9 です。１ならば１声、９ならば９声で発声します。

設定値	1	2	3	4	5	6	7	8	9
有効OP	OP1	OP1-OP2	OP1-OP3	OP1-OP4	OP1-OP5	OP1-OP6	OP1-OP7	OP1-OP8	OP1-OP9

y"clipmode",[1]：

クリップモードの値を [1] に設定します。
0 の場合、クリップモード無効（初期設定）
1 の場合、クリップモード有効
クリップモードとは、全てのオペレータの発音を重ねた結果、
下限-1.0～上限+1.0の範囲の振幅から はみ出す部分を、下限と上限に制限する機能です。用途は、ディストーション効果を得ることです。

y"ampdenom",[1]：

音量比のオートバランサー機能の値を [1] に設定します。
0 の場合、オートバランサー無効
1 の場合、オートバランサー有効（初期設定）
音量比のオートバランサーは、マルチオペレータモードにおける音割れ防止が目的の機能です。

オートバランサーによる合計振幅への倍率計算は、
　　倍率＝全有効オペレータの音量倍率の合計の逆数
により求められます。

y"detune",[1]：
マルチオペレータモード時の音程比を、コーラスモード [1] cent に設定します（初期設定は 8、設定範囲は -100～100）。

【wvm:コーラスモードの音程比】

OP番号	OP1	OP2	OP3	OP4	OP5	OP6	OP7	OP8	OP9
centずれ	0	[1]	-[1]	[1]/2	-[1]/2	[1]/4	-[1]/4	[1]/8	-[1]/8

y"oversmp",[1]：
オーバーサンプリングモードを、[1] に設定します。
設定範囲は 1 ～ 8 です。
初期設定は 2 です（２倍オーバーサンプリング）。

その他のオペレータごとの設定：

項目名	挙動
y"***subform",[1]	オペレータの音源サブフォーム番号を、 [1] に設定します。
y"***phase",[1]	オペレータの位相を、[1] に設定します。 設定範囲は 0.0 ～ 1.0未満です。 -1 を指定した場合は、0.0 ～ 1.0未満の範囲の値がランダムで設定されます。
y"***dB",[1]	オペレータの音量比を、[1] dBに設定します。 設定範囲は -120.0 ～ 48.0 です。
y"***mul",[1]	オペレータの周波数倍率を、[1] に設定します。 設定範囲は 0.0 以上です。
y"***cent",[1]	オペレータの周波数比を、[1] centに設定します。 設定範囲は、-9600.0 ～ 9600.0 です。
y"***freq",[1]	オペレータの周波数への加算値を、 [1] Hzに設定します。

「***」に当てはまる文字が「all」の場合、全てのオペレータが対象
「***」に当てはまる文字が「op1」の場合、オペレータ１のみが対象
「***」に当てはまる文字が「op2」の場合、オペレータ２のみが対象
「***」に当てはまる文字が「op3」の場合、オペレータ３のみが対象
「***」に当てはまる文字が「op4」の場合、オペレータ４のみが対象
「***」に当てはまる文字が「op5」の場合、オペレータ５のみが対象
「***」に当てはまる文字が「op6」の場合、オペレータ６のみが対象
「***」に当てはまる文字が「op7」の場合、オペレータ７のみが対象

「***」に当てはまる文字が「op8」の場合、オペレータ８のみが対象
「***」に当てはまる文字が「op9」の場合、オペレータ９のみが対象

6.9. @@"fms"：ＦＭ音源

【記述例】

```
@@"fms"
```

【解説】

使用する音源モジュールを、ＦＭ音源にします。

【備考１】
ＦＭ音源の音色作成に関わる用語説明

【備考２】
ＦＭ音源では、メタデータ定義

```
#MB:FMS_6OP      （６オペレータモード）
#MB:FMS_4OP      （４オペレータモード）
#MB:FMS_2OP      （２オペレータモード）
#MB:FMS_OPM      （ＯＰＭ互換の４オペレータ）
#MB:FMS_OPNA     （ＯＰＮＡ互換の４オペレータ）
#MB:FMS_OPN      （ＯＰＮ互換の４オペレータ）
```

のいずれかによって、音色データに音色番号を割り当てて登録したものを使用します。音色データの選択は、音源サブフォーム番号（@番号）によって、音色番号を指定して行います。

【備考３】
当ＦＭ音源モジュールは、ＯＰＭの上位互換になっていて、内部にオペレータを６個持っています。
音色定義の書式は複数ありますが、複数の独立した音色定義領域を持っている訳ではなく、実体は６オペレータモード用の記憶領域があるだけです。
どの音色記述方法でも、６オペレータモード用の記憶領域に設定されます。
６オペ以外の記述方法では、使わないオペレータを未使用状態に設定しています。そのため、音色番号３番は６オペモード、音色番号８番は４オペモードというように、オペレータモードが混在した定義になります。
同じ番号に複数の音色定義をした場合、最後の定義が有効になるので、音色番号の重複定義には注意してください。

【備考４】
有効な音程の範囲は、実際のＯＰＭでは 3.58MHz 駆動の場合、

「o0C+ から o8C まで」ですが、当ＦＭ音源モジュールでは、ＭＭＬオクターブ指定における範囲と同等の範囲で受付可能です。
ただし、音色データ内のKS（キースケーリング）は、実際のＯＰＭ相当における最低音程以下には最低音程、最高音程以上には最高音程とみなして動作します。

・yコマンド（１）音源モジュールへの設定

y"subform",[1]：
全てのオペレータの音源サブフォーム番号を、[1] に設定します。
設定範囲は、0 ～ 1023 です。
ただし、メタデータ定義にて音色データを定義した番号に限られます。

y"reso_mode",[1]：
フェーズジェネレータ部の波形生成モードを、[1] に設定します。
設定範囲は、0 ～ 8 です。
0 の場合、ネイティブモード(※)にします。
1 の場合、384000Hz で波形生成を行います。
2 の場合、192000Hz で波形生成を行います。
3 の場合、128000Hz で波形生成を行います。
4 の場合、 96000Hz で波形生成を行います。
5 の場合、 76800Hz で波形生成を行います。
6 の場合、 64000Hz で波形生成を行います。
7 の場合、 54587.1248...Hz で波形生成を行います。
8 の場合、 48000Hz で波形生成を行います。
（1～8の場合、384000÷設定値の生成レートとなります）
※ネイティブモードとは

```
#MB:CONFIG {
   fms_master: clock=[1]: prescale=[2],
}
```

上記の設定によるレート（clock÷prescale）で波形生成を行うモードです。

y"eg_reset",[1]：
ノートオン時の初期エンベロープレベル（IEL）の動作モードを、[1] に設定します。
設定範囲は、-1 または 0 ～ 128 です。初期設定は -1 です。
-1 の場合、IELは、音色設定内の IEL の値に従います。
0 ～ 128 の場合、IELは、全てのオペレータが指定値に従います（IELが強制的に上書き設定されます）。128に近づくほど、無音からのエンベロープ開始になります。

【備考】

音色を OPM / OPNA / OPN モードで定義している場合は、いずれもＡＲ初期レベルを変更できませんが、内部では全てのオペレータで「ノートオン直前までのレベルを引き継ぐモード」に設定されています。

y"fb_bias",[1]：
フィードバックレベルへのバイアス調整値を [1] に設定します。
設定範囲は 0.75 ～ 1.35 です。初期設定は 1.0（バイアス無し）です。
指定値が大きくなるほど音色が明るくなり、小さくなるほど曇った感じになります。

y"set_hlfo",[1]：
ハードＬＦＯの設定を行います。
この設定は、@mh または @mha により内部で呼び出すために使用されますので、通常は使用しないでください。

y"set_dampfunc",[1]：
エンベロープダンパー（消音機能）の設定、または、実行を行います。
エンベロープダンパーでは、休符途中でのリリースレートを一時的に操作することで消音を行います。
ダンパー実行後は、次のノートオンで元のリリースレートに自動復帰します。
指定値による動作は次の通りです。
-1 の場合、設定済み内容でダンパーを実行します。
0 ～ 15 の場合、ダンパー実行の際、この指定値を全てのオペレータのリリースレートに適用されるよう設定します（設定のみで消音は実行しません）。

y"reset_addv",[1]：
[1] による宛先の、全オペレータへの加算値設定を 0 にリセットします。
0 の場合、"*_add_dt3"、"*_add_tl"、"add_el_each"を 0 にリセットします。
1 の場合、"*_add_dt3"を 0 にリセットします。
2 の場合、"*_add_tl"を 0 にリセットします。
3 の場合、"add_el_each"を 0 にリセットします。
※「*」に当てはまる文字は"op1"～"op6"。全てのオペレータが対象。

y"add_tl_ame1",[1]：
音色設定内の AME が 1 になっているオペレータのみ、
TL値が、音色設定の TL に [1] を加算した値になります。
このコマンドは、"*_add_tl"の設定内容を、AME が 1 になっているオペレータのみを対象に上書きするものです。
※「*」に当てはまる文字は"op1"～"op6"。

y"add_el",[1]：

全オペレータの、現在のエンベロープ出力レベルに対し、[1] を直ちに加算します。（@mzコマンドと同様）

y"add_el_each",[1]：
全オペレータの、現在のエンベロープ出力レベルに対し、ノートオンの都度、ノートオンの直前に [1] を加算します。（@mzeコマンドと同様）

y"hlfo_mode",[1]：
ＦＭ音源のハードＬＦＯモードを、[1] に設定します。
設定範囲は、-1, 0, 1 のいずれかです。初期設定は -1 です。
-1 のとき、ハードＬＦＯ処理を行いません。
0 のとき、ＯＰＭモードでハードＬＦＯ処理を行います。
1 のとき、ＯＰＮＡモードでハードＬＦＯ処理を行います。
ハードＬＦＯ処理を行わない場合、その分処理が軽くなります。

y"@mh_wf",[1]：
OPM HARD LFO wave form（0～3）

y"@mh_lfrq",[1]：
OPM HARD LFO frequency（0～255）

y"@mh_pmd",[1]：
OPM HARD LFO pmd（0～127）

y"@mh_amd",[1]：
OPM HARD LFO amd（0～127）

y"@mh_pmsams",[1]：
OPM HARD LFO pms/ams
pmsは0～7で上位4bit、amsは0～3で下位4bit の書式で指定します。例えば、pms=7、ams=3のときは、0x73なので、10進数では115で指定します。

y"@mh_sync",[1]：
OPM HARD LFO sync-mode（0～1）

y"@mh_spd_adj",[1]：
OPM HARD LFO SpeedAdj.
ＯＰＭ互換ＨＬＦＯの速度倍率を、[1] に設定します。
設定範囲は 0.5 ～ 2.0 です。初期設定は 1.0（等倍）です。

y"@mha_lfrq",[1]：
OPNA HARD LFO frequency（0～7）

y"@mha_amspms",[1]：
OPNA HARD LFO ams/pms
amsは0～3で上位4bit、pmsは0～7で下位4bit の書式で指定します。例えば、
ams=3、pms=7のときは、0x37なので、10進数では55で指定します。

y"@mha_sync",[1]：
OPNA HARD LFO sync-mode（0～1）

y"@mha_spd_adj",[1]：
OPNA HARD LFO SpeedAdj.
ＯＰＮＡ互換ＨＬＦＯの速度倍率を、[1] に設定します。
設定範囲は 0.5 ～ 2.0 です。初期設定は 1.0（等倍）です。

その他のオペレータごとの設定：

項目名	挙動
y"***_add_dt3",[1]	オペレータの detune3 への加算値を、 [1] に設定します。（この加算値は記憶されます） ※この加算値に現在設定されている音色データから対応する detune3 値を呼び出して加算した結果を音源モジュールにセットします。ただし、加算結果は -9600～9600 に制限されます。
y"***_add_tl",[1]	オペレータの TL への加算値を、 [1] に設定します。（この加算値は記憶されます） ※この加算値に現在設定されている音色データから対応する TL 値を呼び出して加算した結果を音源モジュールにセットします。ただし、加算結果は 0～127 に制限されます。
y"***_iph",[1]	オペレータの IPH を、[1] に設定します。
y"***_iel",[1]	オペレータの IEL を、[1] に設定します。
y"***_ar",[1]	オペレータの AR を、[1] に設定します。
y"***_d1r",[1]	オペレータの D1R を、[1] に設定します。
y"***_d2r",[1]	オペレータの D2R を、[1] に設定します。
y"***_rr",[1]	オペレータの RR を、[1] に設定します。
y"***_d1l",[1]	オペレータの D1L を、[1] に設定します。
y"***_tl",[1]	オペレータの TL を、[1] に設定します。
y"***_ks",[1]	オペレータの KS を、[1] に設定します。
y"***_mul",[1]	オペレータの MUL を、[1] に設定します。
y"***_dt1",[1]	オペレータの DT1 を、[1] に設定します。
y"***_dt2",[1]	オペレータの DT2 を、[1] に設定します。
y"***_dt3",[1]	オペレータの DT3 を、[1] に設定します。
y"***_dt4",[1]	オペレータの DT4 を、[1] に設定します。
y"***_fb",[1]	オペレータの FB を、[1] に設定します。
y"***_ame",[1]	オペレータの AME を、[1] に設定します。
y"***_msk",[1]	オペレータの MSK を、[1] に設定します。

「***」に当てはまる文字が「op1」の場合、オペレータ１のみが対象
「***」に当てはまる文字が「op2」の場合、オペレータ２のみが対象
「***」に当てはまる文字が「op3」の場合、オペレータ３のみが対象
「***」に当てはまる文字が「op4」の場合、オペレータ４のみが対象
「***」に当てはまる文字が「op5」の場合、オペレータ５のみが対象
「***」に当てはまる文字が「op6」の場合、オペレータ６のみが対象

・yコマンド（２）特殊動作対応

y"sys1_msvchv",[1]：
system1 opm-volume support
master volumeとch.volumeの設定です。引数からTL値を計算してキャリアのTLに音量の書き込みを行います。
[1]: 16進数で「0xaabbccdd」と記述した場合、
　　aa: 未使用
　　bb: 未使用
　　cc: master volume 0-255
　　dd: ch.volume 0-255
master volume及びch.volumeは、書き込んだ値が記憶され、"sys1_msv"もしくは"sys1_chv"の yコマンドで利用されます。
master volume及びch.volumeの初期値は双方255です。

[1]に負数を指定した場合、
現状のmaster volume及びch.volumeにてTL値を再計算して設定します。
負数指定は、音色の変更があった直後の使用を想定しています。

y"sys1_msv",[1]：
system1 opm-volume support
master volume の設定です。引数からTL値を計算してキャリアのTLに音量の書き込みを行います。
[1]: master volume 0-255
ch.volumeは、以前書き込んだ値が読み出されて設定されます。ch.volumeの初期値は255です。

[1]に負数を指定した場合、
現状のmaster volume及びch.volumeにてTL値を再計算して設定します。
負数指定は、音色の変更があった直後の使用を想定しています。

y"sys1_chv",[1]：
system1 opm-volume support
ch.volume の設定です。引数からTL値を計算してキャリアのTLに音量の書き込みを行います。
[1]: ch.volume 0-255
master volume は、以前書き込んだ値が読み出されて設定されます。master volume の初期値は255です。

[1]に負数を指定した場合、

現状のmaster volume及びch.volumeにてTL値を再計算して設定します。
負数指定は、音色の変更があった直後の使用を想定しています。

y"sys1_pmdamd",[1]：
system1 opm HLFO support
HLFOパラメータのPMDとAMDの２つを１度に設定します。
[1]: 16進数で「0xaabb」と記述した場合、
　　aa: PMDパラメータ
　　bb: AMDパラメータ

6.10. @@"nzw"：ホワイトノイズ音源

【記述例】

```
@@"nzw" @0 @n8
```

【解説】

使用する音源モジュールを、ホワイトノイズ音源にします。
ホワイトノイズ音源では、変位がランダムに変更されます。

ホワイトノイズ音源では、変位の更新間隔を @nコマンド で変更することで、
ノイズの質感を変えることができます。

また、@コマンドで変位の変更解像度を変更することができます。
@0　　：従来通りの無段階相当のホワイトノイズ
@1～@999：２段階（high/low）～1000段階までの、有段階ノイズ
想定している主な設定は、@0（無段階）、@1（２段階）、@15（16段階）です。

【yコマンドにおける、音源モジュールへの設定】

y"subform",[1]：
音源サブフォーム番号を、[1] に設定します。
（当音源宛の @コマンドと同様の効果）
設定範囲は、0 ～ 999 です。

y"cycle",[1]：
ホワイトノイズの変位が更新される間隔を、[1] に設定します。
（当音源宛の @nコマンドと同様の効果）

y"base_cycle",[1]：
ホワイトノイズの変位更新間隔の基本単位を、[1] に設定します。
当設定は、

```
#MB:CONFIG { nzw_noise: base_cycle=[1], }
```

この設定と機能は同じですが、yコマンドで記述するため、トラック別に設定変更したい場合に利用できます。

6.11. @@"nzp"：ＰＳＧノイズ音源

【記述例】

```
@@"nzp"
```

【解説】
使用する音源モジュールを、ＰＳＧノイズ音源にします。

ＰＳＧノイズは、変位は上下の２値に限られますが、それぞれの変位を維持する時間（パルス幅）がランダムに変更されます。
ただし、更新間隔が非常に短い２値ノイズは再生サンプリング周波数を超えた密度になるため、過密部分は相加平均処理しています。そのため、Ｖ３ＭＭＬでは相加平均部分がホワイトノイズのように出力されます。

ＰＳＧノイズ音源では、パルス幅の更新間隔を @nコマンド で変更することで、ノイズの質感を変えることができます。

【yコマンドにおける、音源モジュールへの設定】

y"sreg",[1]：
２値ノイズ用シフトレジスタの値を [1] に設定します。
（当音源宛の @nrコマンドと同様です）
次の場合によって指定された値の使われ方が変わります。
※ＰＳＧノイズ用シフトレジスタのビット幅は 17bit 固定です
（１）０以上、１未満の値
（２）１以上、131072-1 未満の値
（３）上記（１）（２）以外の値

（１）の場合、指定値に 131072-1 を乗算し、結果が１未満なら１、131072-2 以上なら 131072-2 に制限してシフトレジスタに設定されます。
（２）の場合、値が整数に四捨五入されてシフトレジスタに設定されます。
（３）の場合（負の値など）、０以上１未満のランダム値を作成し、（１）同様の処理をします。

y"cycle",[1]：
ＰＳＧノイズのパルス幅が更新される間隔を、[1] に設定します。
（当音源宛の @nコマンドと同様です）

設定範囲は 1 ～ clk です。
※clk = ＰＳＧノイズのマスタークロック値

y"m_clock",[1]：
ＰＳＧノイズのマスタークロックを、[1] に設定します。
設定範囲は 1 ～ 10000000（10MHz）です。

y"prescale",[1]：
ＰＳＧノイズのプリスケールを、[1] に設定します。
（プリスケールは、マスタークロックを割る数です）
設定範囲は 1 ～ 65536 です。

y"clk_mode",[1]：
ＰＳＧノイズのクロックモードを、[1] に設定します。
設定範囲は 0、1、2、3 のうちのいずれかです。
0 の場合、PC88モード（clock=3993600, prescale=32, width=17）
1 の場合、 X1 モード（clock=4000000, prescale=32, width=17）
2 の場合、MSX モード（clock=3579545, prescale=32, width=17）
3 の場合、FM-7モード（clock=1228800, prescale=16, width=17）

6.12. @@"nzf"：ＦＣノイズ音源

【記述例】

```
@@"nzf"
```

【解説】

使用する音源モジュールを、ＦＣノイズ音源にします。

ＦＣノイズは、変位は上下の２値に限られますが、それぞれの変位を維持する時間（パルス幅）がランダムに変更されます。
ただし、更新間隔が非常に短い２値ノイズは再生サンプリング周波数を超えた密度になるため、過密部分は相加平均処理しています。そのため、Ｖ３ＭＭＬでは相加平均部分がホワイトノイズのように出力されます。

ＦＣノイズ音源では、パルス幅の更新間隔を @nコマンド で変更することで、ノイズの質感を変えることができます。

【yコマンドにおける、音源モジュールへの設定】

y"subform",[1]：
ＦＣノイズの音源サブフォーム番号を、[1] に設定します。
（当音源宛の @コマンドと同様です）
音源サブフォーム指定で、長周期モードと短周期モードの変更ができます。
設定範囲は、0 ～ 1 です。初期設定は 0 です。
0 の場合、長周期モードに設定します。
1 の場合、短周期モードに設定します。

y"sreg",[1]：
２値ノイズ用シフトレジスタの値を [1] に設定します。
（当音源宛の @nrコマンドと同様です）
次の場合によって指定された値の使われ方が変わります。
※ＦＣノイズ用シフトレジスタのビット幅は、15bit（ロング）または、
4bit（ショート）です。
（１）０以上、１未満の値
（２）１以上、32768-1 未満の値（ロングモード時）
（３）１以上、16-1 未満の値（ショートモード時）
（４）上記（１）（２）（３）以外の値

（１）の場合、ロングモードでは指定値に 32768-1 を乗算し、結果が１未満なら１、32768-2 以上なら 32768-2 に制限してシフトレジスタに設定されます。
ショートモードでは、ロングモードの時の「32768」の部分が「16」の制限に変わります。
（２）または（３）の場合、値が整数に四捨五入されて、シフトレジスタに設定されます。
（４）の場合（負の値など）、０以上１未満のランダム値を作成し、（１）同様の処理をします。

y"cycle",[1]：
ＦＣノイズのパルス幅が更新される間隔を、[1] に設定します。
（当音源宛の @nコマンドと同様です）
設定範囲は 0 ～ 15 の整数です。
指定値は、あらかじめ決められた１６通りのサイクルの呼び出し番号に使用されます。値が大きくなるほど、低いサイクルになります。

6.13. @@"nzg"：ＧＢノイズ音源

【記述例】

```
@@"nzg"
```

【解説】
使用する音源モジュールを、ＧＢノイズ音源にします。

ＧＢノイズは、変位は上下の２値に限られますが、それぞれの変位を維持する時間（パルス幅）がランダムに変更されます。
ただし、更新間隔が非常に短い２値ノイズは再生サンプリング周波数を超えた密度になるため、過密部分は相加平均処理しています。そのため、Ｖ３ＭＭＬでは相加平均部分がホワイトノイズのように出力されます。

ＧＢノイズ音源では、パルス幅の更新間隔を @nコマンド で変更することで、ノイズの質感を変えることができます。

【yコマンドにおける、音源モジュールへの設定】

y"subform",[1]：
ＧＢノイズの音源サブフォーム番号を、[1] に設定します。
音源サブフォーム指定で、長周期モードと短周期モードの変更ができます。
設定範囲は、0 ～ 1 です。初期設定は 0 です。
0 の場合、長周期モードに設定します。
1 の場合、短周期モードに設定します。

y"cycle",[1]：
ＧＢノイズのパルス幅が更新される間隔を、[1] に設定します。
（当音源宛の @nコマンドと同様です）
設定範囲は 0 ～ 157 の整数です。
ＧＢノイズ音源では、サイクルのパラメータを、0～7 のレートと、0～15 のシフト数の、２つのパラメータで決定します。
10進数で、一の位の桁をレート、十から百の位の桁をシフト数に見立てて指定します。一の位の、8～9 は、7 としてみなされます。
例えば、指定値 105 は、レート 5、シフト数 10 です。

6.14. @@"nzc"：波形メモリノイズ音源

【記述例】

```
@@"nzc"
```

【解説】

使用する音源モジュールを、波形メモリノイズ音源にします。

波形メモリノイズ音源は、メタデータ定義（#MB:WVM_WAVE）によって、
波形データに波形番号を割り当てて登録したものを利用したノイズ音源です。
波形データの選択は、音源サブフォーム番号（@番号）によって、波形番号を指定して行います。

波形メモリノイズ音源で使用する波形データは、定義されている内容の先頭３２サンプルのみ使用されます。３２サンプルに満たない波形定義の場合は、残りの部分が変位０（中心点）で埋められて使用されます。
波形メモリノイズでは、線形帰還シフトレジスタの演算を利用して、擬似ランダムに波形サンプルを取り出して変位に採用します。

波形メモリノイズ音源では、変位を更新する間隔を @nコマンド で変更することで、ノイズの質感を変えることができます。

【yコマンドにおける、音源モジュールへの設定】

y"subform",[1]：
波形メモリノイズの音源サブフォーム番号を、[1] に設定します。
設定範囲は、0 ～ 1023 です。
ただし、メタデータ定義にて波形データを定義した番号に限られます。

y"sreg",[1]：
波形メモリノイズ用シフトレジスタの値を [1] に設定します。
（当音源宛の @nrコマンドと同様です）
次の場合によって指定された値の使われ方が変わります。
※波形メモリノイズ用シフトレジスタのビット幅は、16bit 固定です。
（１）０以上、１未満の値
（２）１以上、65536-1 未満の値
（３）上記（１）（２）以外の値

（１）の場合、指定値に 65536-1 を乗算し、結果が１未満なら１、
65536-2 より大きければ 65536-2 に制限してシフトレジスタに設定されます。
（２）の場合、値がそのままシフトレジスタに設定されます。
（３）の場合（負の値など）、０以上１未満のランダム値を作成し、（１）同様の処理をします。

y"cycle",[1]：
波形メモリノイズの変位が更新される間隔を、[1] に設定します。
（当音源宛の @nコマンドと同様です）
設定範囲は 0.25 ～ baseCycle です。
baseCycleは通常 2048.0 ですが、「y"step_r_mgnf",[1]」の設定に依存します。
「y"step_r_mgnf",[1]」の設定値を MGNF とした場合、
baseCycle = 2048.0 ÷ MGNF になります。

※2048 = (384000/24000)*128

y"step_r_mgnf",[1]：
波形メモリノイズの変位更新間隔の倍率を、[1] に設定します。
設定範囲は 0.0625 ～ 16.0 です。初期設定は 1.0 です。
この yコマンドは、当音源宛の @nコマンド による設定に影響を与えます。
当設定は、

```
#MB:CONFIG { nzc_noise: step_adj=[1], }
```

この設定と機能は同じですが、yコマンドで記述するため、トラック別に設定変更したい場合に利用できます。

6.15. @@"pcm"：ＰＣＭ音源

【記述例】

```
@@"pcm"
```

【解説】
使用する音源モジュールを、ＰＣＭ音源にします。

ＰＣＭ音源では、メタデータ定義（#MB:LOAD_SAMPLING）によって、ＰＣＭ波形データに波形番号を割り当てて登録したものを使用します。
波形データの選択は、音源サブフォーム番号（@番号）によって、波形番号を指定して行います。

【yコマンドにおける、音源モジュールへの設定】

y"subform",[1]：
ＰＣＭ音源サブフォーム番号（波形番号）を [1] に設定します。
設定範囲は、0 ～ 2047 です。
ただし、メタデータ定義にて波形データを定義した番号に限られます。

y"cmpl_mode",[1]：
ＰＣＭ再生における補完モードの選択を行います。
設定範囲は、0 ～ 1 です。初期設定は 1 です。
0 の場合、再生を０次補完モードで行います。
1 の場合、再生を１次補完モードで行います。
（当音源宛の @cコマンドと同様です）

6.16. @m[1]：@@"fms"用：波形生成モード

【記述例】

```
@m0
```

この場合、ネイティブモード(※)でＦＭ音源の波形生成を行います。

【解説】

ＦＭ音源モジュール（@@"fms"）のフェーズジェネレータ部の、波形生成モード（解像度）を設定します。
引数[1]には、モード番号を指定します。
設定範囲は、0 ～ 8 です。

0 の場合、ネイティブモード(※)にします。
1 の場合、384000Hz で波形生成を行います。
2 の場合、192000Hz で波形生成を行います。
3 の場合、128000Hz で波形生成を行います。
4 の場合、 96000Hz で波形生成を行います。
5 の場合、 76800Hz で波形生成を行います。
6 の場合、 64000Hz で波形生成を行います。
7 の場合、 54587.1248...Hz で波形生成を行います。
8 の場合、 48000Hz で波形生成を行います。
（1～8の場合、384000÷設定値の生成レートとなります）

※ネイティブモードとは・・

```
#MB:CONFIG {
   fms_master: clock=[1]: prescale=[2],
}
```

上記設定によるレート（clock÷prescale）で波形生成を行うモードです。
例えば、fms_masterが、3579545÷64 = 55930.390625 に設定されていて、
コマンド「@m0」が記述されていれば、3.579545MHzのＯＰＭ互換の波形生成モードとなります。

【備考】

ネイティブモードよりも高いレートで波形生成を行う場合の利点は、
・ノイズ低減。
・計算精度向上。
それに対し欠点は、
・FBの強い音色の互換性低下。

・折り返しノイズを見込んだ音色の互換性低下。
・処理が重くなる。

などがあります。

6.17. @mh：@@"fms"用：ＯＰＭ互換ＨＬＦＯ

【記述例】

```
@mh2,80,127,0,3,0,0
```

【解説】

ＦＭ音源モジュール（@@"fms"）用の、ＯＰＭ互換ＨＬＦＯ
（Hardware Low Frequency Oscillator）機能を設定します。

@mhコマンド と @mhaコマンド は排他制御が掛かります。
排他制御により、@mh を使うと、それまでの @mha 指定は無効になり、
@mha を使うと、それまでの @mh 指定は無効になります。
（両方同時にＨＬＦＯは掛かりません）

【書式】@mh[1],[2],[3],[4],[5],[6],[7]

指定する引数は次の通りです。

```
引数[1]...波形（wf）            [wave form]
引数[2]...周波数（lfrq）        [lfo frequency]
引数[3]...音程深度（pmd）       [pitch modulation depth]
引数[4]...音量深度（amd）       [amplitude modulation depth]
引数[5]...音程感度（pms）       [pitch modulation sensitivity]
引数[6]...音量感度（ams）       [amplitude modulation sensitivity]
引数[7]...同期モード（sync）    [synchronization]
```

引数はカンマで区切って指定します。
引数[1]の wf 以外は省略可能で、省略した場合は後ろから順に省略したものとみなされます。省略時には初期値（0）が採用されます。
トラック先頭における初期設定は、当機能無効（@mhz指定状態）です。

引数[1]（wf）

ＬＦＯで使用する波形を指定します。
範囲は 0～3。初期設定は 0 です。
0: のこぎり波
1: 矩形波
2: 三角波
3: ランダム波（ノイズ）

引数[2]（lfrq）
ＬＦＯの周波数を指定します。
範囲は 0～255。初期設定は 0 です。

引数[3]（pmd）
音程ＬＦＯの深度を指定します。
範囲は 0～127。初期設定は 0 です。

引数[4]（amd）
音量ＬＦＯの深度を指定します。
範囲は 0～127。初期設定は 0 です。

引数[5]（pms）
音程ＬＦＯの感度を指定します。
範囲は 0～7。初期設定は 0 です。

引数[6]（ams）
音量ＬＦＯの感度を指定します。
範囲は 0～3。初期設定は 0 です。

引数[7]（sync）
ノートオン同期モードを指定します。
範囲は 0 または 1 。初期設定は 0 です。
0 を指定すると、ノートオンでＬＦＯ波形の位相はリセットされません。
1 を指定すると、ノートオンの都度、ＬＦＯ波形の位相が最初からスタートします。

6.18. @mha：@@"fms"用：ＯＰＮＡ互換ＨＬＦＯ

【記述例】

```
@mha3,4,0,0
```

【解説】

ＦＭ音源モジュール（@@"fms"）用の、ＯＰＮＡ互換ＨＬＦＯ
（Hardware Low Frequency Oscillator）機能を設定します。

@mhaコマンド と @mhコマンド は排他制御が掛かります。
排他制御により、@mha を使うと、それまでの @mh 指定は無効になり、
@mh を使うと、それまでの @mha 指定は無効になります。
（両方同時にＨＬＦＯは掛かりません）

【書式】@mha[1],[2],[3],[4]

指定する引数は次の通りです。
引数[1]...周波数（lfrq）　　[lfo frequency]
引数[2]...音程感度（pms）　　[pitch modulation sensitivity]
引数[3]...音量感度（ams）　　[amplitude modulation sensitivity]
引数[4]...同期モード（sync）　[synchronization]
引数はカンマで区切って指定します。
引数[1]の lfrq 以外は省略可能で、省略した場合は後ろから順に省略したものとみなされます。省略時には初期値（0）が採用されます。
トラック先頭における初期設定は、当機能無効（@mhz指定状態）です。

引数[1]（lfrq）

ＬＦＯの周波数を指定します。
範囲は 0～7。初期設定は 0 です。

引数[2]（pms）

音程ＬＦＯの感度を指定します。
範囲は 0～7。初期設定は 0 です。

引数[3]（ams）

音量ＬＦＯの感度を指定します。
範囲は 0～3。初期設定は 0 です。

引数[4]（sync）

ノートオン同期モードを指定します。
範囲は 0 または 1 。初期設定は 0 です。
0 を指定すると、ノートオンでＬＦＯ波形の位相はリセットされません。
1 を指定すると、ノートオンの都度、ＬＦＯ波形の位相が最初からスタートします。

6.19. @mhz：@@"fms"用：ＨＬＦＯ無効

【記述例】

```
@mhz
```

【解説】

ＦＭ音源モジュール（@@"fms"）用の、ＨＬＦＯ
（Hardware Low Frequency Oscillator）機能を無効に設定します。

ＨＬＦＯ機能が無効になると、ＦＭ音源の動作がその分軽くなります。
トラック開始時の初期設定では、ＨＬＦＯ機能は無効になっています。

ＨＬＦＯ機能が無効の状態では、yコマンドを使用して直接ＨＬＦＯパラメータ設定を行っても動作しないので注意してください。

ＨＬＦＯ機能を利用する場合、@mhコマンド または @mhaコマンド を使用すると、必ずＨＬＦＯ機能が有効化されますので確実です。

6.20. @mr[1]：@@"fms"用：IEL強制設定

【記述例】

```
@mr127 cdefg
@mr64 cdefg;
```

【解説】
ＦＭ音源モジュール（@@"fms"）用の、IEL強制設定を行います。

IELとは、ノートオンによるエンベロープ開始時の出力レベルを設定するパラメータです。

引数[1]には、IEL強制設定値を指定します。
設定範囲は 0 ～ 128 です。

当コマンドを使用すると、IEL強制設定が有効になり、
全てのオペレータの IEL は、強制的に引数[1]の値になります。
（音色設定内のIELが無視されます）
128に近づくほど、無音からのエンベロープ開始になります。

初期設定は、IEL強制設定が無効です（@mrzコマンドを指定した状態）。
IEL強制設定が無効の場合、全てのオペレータの IEL は、音色設定内の指定に従います。

【備考】
音色を OPM / OPNA / OPN モードで定義している場合は、いずれも IELを変更できませんが、全てのオペレータで「ノートオン直前までのレベルを引き継ぐモード」に設定されています。

6.21. @mrz：@@"fms"用：IEL強制設定の停止

【記述例】

```
@mrz
```

【解説】

ＦＭ音源モジュール（@@"fms"）用の、IEL強制設定を解除（無効化）します。
@mrコマンドが有効な場合に無効化したい時に使用します。

6.22. @mz[1]：@@"fms"用：ENV-LVの即時減衰機能

【記述例】

```
ccc @mz20 g @mz20 a @mz20 @mz20 b;
```

このように書くと、gは20、aは20、bは40、エンベロープ出力レベルを減衰してノートオンします。
（@mz20を２回書くことで、20+20=40の減衰になっている点に注意）

【解説】

ＦＭ音源モジュール（@@"fms"）用の、エンベロープ出力レベル（ENV-LV）を、全オペレータで即時減衰させます。
似た機能のエンベロープダンパー（zコマンド）は、リリースレートを操作する方式のため、実際の出力減衰には、ダンパー実行から、わずかに休符時間を必要とします。
当コマンドでは、ENV-LV を即時減衰させます（減衰に休符時間は不要）。

初期エンベロープレベル（IEL）がレベル引き継ぎモードになっていて、なおかつ休符やゲートによる隙間が全くない場合でも、当コマンドをノートオン直前に使用すれば、減衰させた ENV-LV でノートオンさせることができます。

引数[1]には、減衰量を 0 ～ 128 で指定します。
小数以下の指定も受け付けますが、小数以下１１ビットの固定小数点数の精度に丸められて応答します。これは内部の振幅値テーブルサイズ（解像度）の都合によるものです。

減衰量のスケールは、トータルレベル（TL）のスケールと同様です。
例えば @mz20 とした場合、ENV-LV が TL 換算で 20 加算（出力が減衰）したのと同様の効果が、即時適用されます。

6.23. @mze[1]：@@"fms"用：@mz機能の継続適用

【記述例】

```
@mze20 cccggg;
```

このように書くと、cccgggの音符がいずれも、ノートオン直前に20、エンベロープ出力レベルを減衰してから、ノートオンします。

【解説】

ＦＭ音源モジュール（@@"fms"）用の @mzコマンド を、ノートオンの都度、ノートオン直前に適用させます。

初期エンベロープレベル（IEL）がレベル引き継ぎモードになっていて、なおかつ休符やゲートによる隙間が全くない場合でも、当コマンドを使用すれば、以降のノートオンの都度、減衰させたエンベロープ出力レベルでノートオンさせることができます。

引数[1]には、減衰量を 0 ～ 128 で指定します。
小数以下の指定も受け付けますが、小数以下１１ビットの固定小数点数の精度に丸められて応答します。これは内部の振幅値テーブルサイズ（解像度）の都合によるものです。

減衰量のスケールは、トータルレベル（TL）のスケールと同様です。
例えば @mze20 とした場合、エンベロープ出力レベルが TL 換算で 20 加算（出力が減衰）したのと同様の効果が、即時適用されます。

6.24. @oa[1]：オペレータ振幅の自動調整

【記述例】

```
@oa1
```

この場合、振幅調整を行い、レベルオーバーを防ぎます。

【解説】

対象となる音源モジュール、

```
@@"sin"
@@"saw"
@@"tri"
@@"pls"
@@"cpx"
@@"wvm"
```

が選択されている場合に、選択中の音源モジュールに対し、
有効になっているオペレータ数や各オペレータのdB設定に応じて、
レベルオーバーを防ぐ振幅調整を行うか否かを設定します。

@oa1の場合、振幅調整を行い、レベルオーバーを防ぎます。
@oa0の場合、振幅調整を行いません。

【備考】

振幅調整の具体的な処理は、複数オペレータ有効時に、各オペレータの振幅倍率の合計を、振幅の分母に設定する処理です。

```
m_ampDenom = m_op[0].op_amp + m_op[1].op_amp + m_op[2].op_amp ...
```

6.25. @oc[1]：オペレータクリッピング機能

【記述例】

```
@oc1
```

この場合、オペレータクリッピング機能を行います。

【解説】

対象となる音源モジュール、

```
@@"sin"
@@"saw"
@@"tri"
@@"pls"
@@"cpx"
@@"wvm"
```

が選択されている場合に、選択中の音源モジュールに対し、
有効になっている全オペレータの合計出力に対し、
標準振幅範囲の -1.0 ～ +1.0 からはみだす部分につき、
+1.0以上を+1.0、-1.0以下を-1.0に制限（クリッピング）
するか否かを設定します。

@oc1の場合、クリッピングを行います。
@oc0の場合、クリッピングを行いません。

この機能の目的は、ディストーション効果を得ることです。

6.26.@ph[1],[2]：位相リセットモード設定

【記述例】

```
@@"wvm" @0 @ph1,0 cde;
```

この場合、ノートオンの都度、位相が０スタートします。

【解説】

ノートオンの都度、音源モジュールの位相をリセットするかどうか、の設定を行います。ただし、位相リセットをサポートする音源モジュールが選択されている場合に限ります。

【位相リセットをサポートする音源モジュールとその初期設定】

音源名	初期設定	音源名	初期設定
@@"sin"	@ph0,-1	@@"pls"	@ph0,-1
@@"saw"	@ph0,-1	@@"cpx"	@ph0,-1
@@"tri"	@ph0,-1	@@"wvm"	@ph0,-1

指定する引数は次の通りです。
引数[1]...モード番号（mode）
引数[2]...位相値（phase）
引数はカンマで区切って指定します。

@phコマンドを受け付けた直後のノートオンは、mode に関わらず、
必ず位相が phase の内容でリセットされます。

引数[1]（mode）

ノートオン時の、位相リセットモードを整数で指定します。

mode が 0 のとき

ノートオンの都度、位相をリセットしません。

mode が 1 のとき

ノートオンの都度、位相をリセットします。

mode が 2 のとき

今回ノートオンが直前ノートオフと同時刻の場合（休符やゲートでの間隔が無い場合）、位相をリセットしません。
今回ノートオンが直前ノートオフと同時刻でない場合（休符やゲートでの間隔がある場合）、位相をリセットします。

mode が 3 のとき

無音状態（エンベロープが完全に終了した状態）からのノートオンの場合、位相をリセットします。
無音状態でない場合でのノートオンの場合、位相をリセットしません。

mode が 4 のとき

ノートオンの都度、位相をリセットしませんが、「@番号」による音源サブフォーム指定コマンドを使用した直後のノートオンのみ、位相をリセットします。

mode に 0 以上の値を指定した場合で、上記 mode 以外の値を指定した場合、mode に 1 が指定されたものとみなされます。

mode が -1 のとき

既に設定済みの mode と phase を変更せずに、強制的に、既に設定済みの phase に位相をリセットします。この位相リセットは即時実行の単発です（ノートオンの都度には実行しません）。
mode指定が -1 のときは、phase の指定は不要で、指定したとしても無視されます。
-1 以外の負数の指定の場合は、-1 が指定されたものとみなされます。

引数[2]（phase）

modeに従ったタイミングで位相リセットする際の、位相値を指定します。

書式１：0 以上 1 未満の浮動小数点数

ノートオン時にリセットする際の位相を、0 以上 1 未満の浮動小数点数で指定します。省略した場合は、0 を指定したものとみなされます。
（0〜1は、0〜2*pi に対応）

書式２：負数（-1）

phase に -1 と指定すると、ノートオン時の位相をランダムな位置にリセットします。

【備考１】

phase に「?番号」と指定すると、ノートオン時の位相にランダム値パレットを利用できます。
ランダム値パレットとは、ＭＭＬコンパイル開始時に毎回１回初期化される、０～１の浮動少数点数のランダム値を格納する配列で、０番～９９番の１００個が用意されています。例えば、

```
@ph0,?0
```

と指定すると、直後のノートのみ、０番のランダム値パレットで位相リセットします。
これは、複数トラックで開始時の位相のランダム値を一致させたい場合を想定しています。

【備考２】

「?番号」と、引数[2]での「-1」の違いについて。
「?番号」では、番号ごとにコンパイル開始時に数値が決定され、全てのトラック、全ての位置で、番号が同じであれば同じ値が適用されます。
「-1」では、ノートオンなどの位相リセットのタイミングでその都度ランダム値が生成されます。

6.27. @phr：位相リセット即時実行

【記述例】

```
@phr
```

【解説】

選択中の音源モジュールの位相をリセットします。

- この位相リセットは即時実行の単発です（ノートオンの都度の位相リセットには影響しません）。
- 位相のリセット位置は、@phコマンドにより既に設定済みの位相値です。
- @phコマンドによる、既に設定済みの位相リセットモードと位相値の変更は行いません。
- 当コマンドは、位相リセットをサポートする音源モジュールが選択されている場合に限り使用できます。

【位相リセットをサポートする音源モジュール】

@@"sin"	@@"saw"	@@"tri"	@@"pls"	@@"cpx"	@@"wvm"

6.28. @pf[1]：@@"pls"用：効果音対策用周波数設定

【記述例】

```
@@"pls" @pf1 cde;
```

この場合、パルス音源の効果音対策用の周波数設定が有効になり、cde が演奏されます。

【解説】

パルス音源の効果音対策用の周波数設定機能の設定を行います。
引数[1]には、動作パラメータを整数で指定します。
パルス音源内の全てのオペレータが設定対象となります。

【@pfコマンドの設定値と動作内容】

引数[1]設定値	動作内容
-1	効果音対策用周波数設定は無効です。 周波数設定時に特殊処理は行いません。 トラック先頭における初期設定はこのモードになります。 （負数は全て-1として扱われます）
0 ～ 100	効果音対策用周波数設定は有効です。 周波数設定時に特殊処理を行います。 設定値は分周比の内部カウンタを微調整として何回分進めるかの整数として扱われます。通常は設定値に 1 を指定します。 （100以上の値は100として扱われます）

【備考】

PSG(AY-3-8910)におけるパルス音源の発音周波数は、マスタークロック由来の周波数に対し分周比の値を設定することで得られます。
PSGにおいては、内部にマスタークロック由来のアップカウンタがあり、分周比の値になると発音の変位の HIGH / LOW を切り替えていると考えられます。
PSGの効果音の発音においては、分周比が、ごく短時間に連続して書き換えが行われる場合があります。この場合、アップカウンタが増加中に書き換え先の分周比が、アップカウンタ値よりも小さかったり大きかったりした場合の挙動が問題になってきます。この、分周比の書き換えによる挙動の特徴をシミュレートする機能が、「効果音対策用周波数設定」になります。

6.29. @pfz：@pfコマンドの機能を停止

【記述例】

```
@pfz
```

【解説】

@pfコマンド（@@"pls"用の効果音対策用の周波数設定機能）の機能を無効状態にします。「@pf-1」と記述したのと同様の効果になります。

6.30. @w[1],[2]：@@"pls"用：デューティ比設定

【記述例】

```
@@"pls" @w3,8 cdefg
@w1 gfedc;
```

この場合、デューティ比 37.5% で cdefg を演奏してから、
次にデューティ比 12.5% で gfedc を演奏します。

【解説】

パルス波音源（@@"pls"）に対して、デューティ比（パルス幅）を設定します。

指定する引数は次の通りです。
引数[1]...デューティ比の分子（num）
引数[2]...デューティ比の分母（denom）
引数はカンマで区切って指定します。
トラック先頭における初期設定は、@w4,8（デューティ比５０％）です。

引数[1]（num）

デューティ比の分子を数値で指定します。

引数[2]（denom）

デューティ比の分母を１以上の数値で指定します。
カンマとdenomの組は省略可能で、省略した場合はそれまでに設定されたdenomの値が採用されます。

【備考１】
当コマンドの設定範囲は、分子÷分母が、0.005 以上 0.995 以下の範囲です。
範囲外だった場合は、最小未満は最小に、最大超えは最大に制限されます。

【備考２】
一度 denom 付きで設定したら、以降は denom を省略する使い方を想定しています。分母が 8 で構わない場合は、トラック先頭から denom を省略してください。

6.31. @n[1]：ノイズサイクル設定

【記述例】

```
@@"nzw" @n10 ccccc
@n20 ccccc;
```

この場合、選択中の @@"nzw" のノイズサイクルを 10 に設定して ccccc を演奏し、さらにノイズサイクルを 20 に変更して ccccc を演奏します。

【解説】

選択中の音源モジュール宛に、ノイズの更新サイクルを設定します。
宛先になりうる音源モジュールは、

```
@@"pls"
@@"nzw"
@@"nzp"
@@"nzf"
@@"nzg"
@@"nzc"
```

以上の６種類です。
これ以外の宛先に当コマンドを使用した場合、エラーとなります。
引数[1]の有効範囲は、宛先のノイズ音源モジュールによって変わります。

@@"pls" が宛先の場合

引数[1]の設定範囲は、1.0 ～ master_cycle です。
引数[1]が master_cycle と等しい場合、ノイズサイクルは１秒になります。

master_cycle は、次の #MB:CONFIG の初期設定

```
#MB:CONFIG { pls_noise: clock=3993600: prescale=32: width=17, }
```

または、@@"pls"宛の yコマンド

```
y"nz_mclock",[]  および  y"nz_pscale",[]
```

による設定に依存します。（master_cycle = clock/prescale）
#MB:CONFIG は全体指定、yコマンドはトラック別指定になります。

引数[1]の指定値が 1 ～ 31 の整数だった場合、AY-3-8910互換(width=17の場合)です。
指定値が 31 を超える設定値だったり、少数を含む値だった場合は、
1 ～ 31 の整数における計算の延長線上での値で動作します。
指定値が大きくなるほど、低いサイクル（更新サイクルが長い）になります。

@@"nzw" が宛先の場合

引数[1]の設定範囲は、1.0 ～ base_cycle です。

base_cycleの初期設定は、次の #MB:CONFIG の初期設定によります。

```
#MB:CONFIG { nzw_noise: base_cycle=3993600/32, }
```

base_cycleの設定は、#MB:CONFIG による全体指定か、yコマンドによるトラック別指定か、２通りの方法があります。

引数[1]の指定値は、ベースサイクル何回分ごとに更新するかを意味します。
ノイズ変位の更新間隔は、
　　（音声生成レート(384kHz)÷base_cycle）× @n値
で計算されますが、この間隔が、
１サンプル ～ 384000サンプル（1/384000秒～１秒）
の範囲外だった場合、最小未満は最小値に、最大超えは最大値に制限されます。

@@"nzp" が宛先の場合

引数[1]の設定範囲は、1.0 ～ master_cycle です。
引数[1]が master_cycle と等しい場合、ノイズサイクルは１秒になります。

master_cycle は、次の #MB:CONFIG の初期設定

```
#MB:CONFIG { nzp_noise: clock=3993600: prescale=32, }
```

または、@@"nzp"宛の yコマンド

```
y"m_clock",[]  および  y"prescale",[]
```

による設定に依存します。（master_cycle = clock/prescale）
#MB:CONFIG は全体指定、yコマンドはトラック別指定になります。

引数[1]の指定値が 1 ～ 31 の整数だった場合、AY-3-8910互換です。
指定値が 31 を超える設定値だったり、少数を含む値だった場合は、
 1 ～ 31 の整数における計算の延長線上での値で動作します。
指定値が大きくなるほど、低いサイクル（更新サイクルが長い）になります。

@@"nzf" が宛先の場合

引数[1]の設定範囲は、0 ～ 15 の整数です。
指定値は、あらかじめ用意された１６通りのノイズサイクルテーブルの
呼び出し番号に使用されます。
指定値が大きくなるほど、低いノイズサイクルになります。

@@"nzg" が宛先の場合

引数[1]の設定範囲は、0 ～ 157 の整数です。
ＧＢノイズ音源では、サイクルのパラメータを、
0～7 のレートと、0～15 のシフト数の、
２つのパラメータで決定します。
10進数で、一の位の桁をレート、十から百の位の桁をシフト数に見立てて指定します。
一の位の、8～9 は、7 としてみなされます。
例えば、指定値 105 は、レート 5、シフト数 10 です。
傾向として、指定値が大きくなるほど、おおむね低いサイクルのノイズになります。

@@"nzc" が宛先の場合

引数[1]の設定範囲は 0.25 ～ baseCycle です。

baseCycleは通常 2048.0 ですが、次の #MB:CONFIG の初期設定

```
#MB:CONFIG { nzc_noise: step_adj=[], }
```

または、yコマンド

```
y"step_r_mgnf",[]
```

による設定に依存します。
これら設定への指定値を MGNF とした場合、
baseCycle = 2048.0 ÷ MGNF になります。
#MB:CONFIG は全体指定、yコマンドはトラック別指定になります。

※2048 = (384000/24000)*128

6.32. @nr[1]：ノイズ音源用のシフトレジスタ設定

【記述例】

```
@@"nzp" @n64 p1  @nr?1 ccc;
@@"nzp" @n64 p-1 @nr?1 ccc;
```

この場合、ランダム値パレットの１番でシフトレジスタ値を設定したノイズを、別トラックで左右に振って鳴らしていますが、モノラルに聞こえます。
@nrコマンドが無い（シフトレジスタ値が違う）と、ステレオに聞こえます。

【備考】ランダム値パレット

【解説】
シフトレジスタを使用するタイプのノイズ音源モジュールに対し、シフトレジスタの値を設定します。ただし、次のノイズ音源を選択中の場合に限ります。

@@"pls"	パルスノイズ	@@"nzp"	ＰＳＧノイズ
@@"nzf"	ＦＣノイズ	@@"nzc"	波形メモリノイズ

通常はこのコマンドを使用する必要はありません。
利用シーンは、例えば、２トラックを使用して、それぞれレガシーパンポットで左右に振り、異なるエンベロープをつけたいけれど、ノイズパターンは左右一致させたい場合、などです。

引数[1]には、シフトレジスタの値を指定します。
選択中である宛先のノイズ音源モジュールによって、指定値の有効範囲が変わります。

@@"pls" が宛先の場合
パルス波音源の全てのオペレータの２値ノイズ用シフトレジスタの値を、
引数[1] に設定します。
※ノイズ用シフトレジスタのビット幅の設定が bitw だったとします
（１）０以上、１未満の値
（２）１以上、(2^bitw)-1 未満の値
（３）上記（１）（２）以外の値

（１）の場合、指定値に (2^bitw)-1 を乗算し、結果が１未満なら１、

(2^bitw)-2 より大きければ (2^bitw)-2 の整数に制限してシフトレジスタに設定されます。
（２）の場合、値が整数に四捨五入されてシフトレジスタに設定されます。
（３）の場合（負の値など）、０以上１未満のランダム値を作成し、（１）同様の処理をします。

@@"nzp" が宛先の場合

シフトレジスタの値を、引数[1] に設定します。
次の場合によって指定された値の使われ方が変わります。
※ＰＳＧノイズのシフトレジスタは 17bit です。
（１）０以上、１未満の値
（２）１以上、131072-1 未満の値
（３）上記（１）（２）以外の値

（１）の場合、指定値に 131072-1 を乗算し、結果が１未満なら１、131072-2 より大きければ 131072-2 の整数に制限して、シフトレジスタに設定されます。
（２）の場合、値が整数に四捨五入されてシフトレジスタに設定されます。
（３）の場合（負の値など）、０以上１未満のランダム値を作成し、（１）同様の処理をします。

@@"nzf" が宛先の場合

シフトレジスタの値を 引数[1] に設定します。
次の場合によって指定された値の使われ方が変わります。
※ＦＣノイズ用シフトレジスタのビット幅は、15bit（ロング）または、4bit（ショート）です。
（１）０以上、１未満の値
（２）１以上、32768-1 未満の値（ロングモード時）
（３）１以上、16-1 未満の値（ショートモード時）
（４）上記（１）（２）（３）以外の値

（１）の場合、ロングモードでは指定値に 32768-1 を乗算し、結果が１未満なら１、32768-2 以上なら 32768-2 に制限してシフトレジスタに設定されます。
ショートモードでは、ロングモードの時の「32768」の部分が「16」の制限に変わります。
（２）または（３）の場合、値が整数に四捨五入されて、シフトレジスタに設定されます。
（４）の場合（負の値など）、０以上１未満のランダム値を作成し、（１）同様の処理をします。

@@"nzc" が宛先の場合

シフトレジスタの値を 引数[1] に設定します。
次の場合によって指定された値の使われ方が変わります。
※波形メモリノイズ用シフトレジスタのビット幅は、16bit です。
（１）０以上、１未満の値
（２）１以上、65536-1 未満の値
（３）上記（１）（２）以外の値

（１）の場合、指定値に 65536-1 を乗算し、結果が１未満なら１、
65536-2 より大きければ 65536-2 に制限してシフトレジスタに設定されます。
（２）の場合、値が整数に四捨五入されてシフトレジスタに設定されます。
（３）の場合（負の値など）、０以上１未満のランダム値を作成し、（１）同様の処理をします。

6.33. @c[1]：ＰＣＭ音源の補完モード設定

【記述例】

```
@c1
```

この場合、ＰＣＭ再生が１次補完（線形補完）モードになります。

【解説】

対象となる音源モジュール、@@"pcm" が選択されている場合に、
ＰＣＭ再生の補完モードを設定します。

@c0の場合、０次補完を行います。（前回値で補完）
@c1の場合、１次補完を行います。（線形補完）

6.34. y[str],[1]：音源モジュール制御

【記述例】

```
y"subform",1
```

この場合、選択済みの音源モジュールに対し、サブフォーム番号を１に設定します。例えば @@"fms" の状態でこのコマンドが記述された場合はＦＭ音源モジュールの音色番号が１に設定されます。

【解説】

選択済みの音源モジュールに対し、制御コマンドを発行します。
指定する引数は次の通りです。
引数[str]：機能名（ダブルクォーテーション括った文字列）
引数[1]　：パラメータ（数値。計算式記述も可）

選択済みの音源モジュールによって、使える機能名のラインナップは変わります。（詳細は各音源モジュールの説明の項を参照）

6.35. ym[str1],[str2],[1]：音源モジュール制御詳細モード

【記述例】

```
ym"fms","subform",1
```

この場合、ＦＭ音源モジュールの音色番号が１に設定されます。

【解説】

指定の音源モジュールに対し、制御コマンドを発行します。
指定する引数は次の通りです。
引数[str1]：宛先音源名（ダブルクォーテーション括った文字列）
引数[str2]：機能名（ダブルクォーテーション括った文字列）
引数[1]　：パラメータ（数値。計算式記述も可）

指定の音源モジュールによって、使える機能名のラインナップは変わります。
（詳細は各音源モジュールの説明の項を参照）

・ＬＦＯ波形モジュールへの適用

ymコマンドの宛先は、各種ＬＦＯの波形モジュールも対象になります。

【ＬＦＯ波形モジュール宛の識別子】

波形名称　/　LFO宛先	LFO_P	LFO_A	LFO_B	LFO_F	LFO_Y
サイン波	lpsin	lasin	lbsin	lfsin	lysin
ノコギリ波	lpsaw	lasaw	lbsaw	lfsaw	lysaw
三角波	lptri	latri	lbtri	lftri	lytri
パルス波	lppls	lapls	lbpls	lfpls	lypls
ホワイトノイズ	lpnzw	lanzw	lbnzw	lfnzw	lynzw
ユーザー定義テーブル	lptbl	latbl	lbtbl	lftbl	lytbl
ノンリニアベンド	lpnlb	非対応	lbnlb	lfnlb	lynlb

現状有効なＬＦＯ波形モジュールへの制御は次の通りです。

（１）パルス波パターンのデューティ比設定

宛先名：lppls / lapls / lbpls / lfpls / lypls
機能名：pwm
設定範囲：0.05～0.995（初期設定 0.5）
機能内容：
パルス波パターンへのデューティ比を設定します。
【例】

```
ym"lppls","pwm",0.25
```

ピッチＬＦＯのパルス波パターンのデューティ比を、25%に指定します。

（２）ノートオフ同期のユーザー定義テーブル番号指定

宛先名：lptbl / latbl / lbtbl / lftbl / lytbl
機能名：subform2nd
設定範囲：定義済みのユーザー定義テーブル番号（#MB:LFOTBL_REL）
機能内容：
ユーザー定義テーブルでＬＦＯシーケンスを行う場合に、ノートオン同期テーブルによる動作に加え、ノートオフに同期して「#MB:LFOTBL_REL」で定義済みのテーブルに切り替えます（ノートオン同期とノートオフ同期でテーブルが交互に切り替わるようになります）。
ここで指定するテーブル番号を -1 とすると、ノートオフ同期のテーブル切り替え機能が無効になります。
【例】

```
ym"latbl","subform2nd",1
```

（３）ユーザー定義テーブルの処理モード指定

宛先名：lptbl / latbl / lbtbl / lftbl / lytbl
機能名：p_mode
設定範囲：0 または 1（初期設定 0）
機能内容：
ユーザー定義テーブルでLFOシーケンスを行う場合の、テーブル処理モード（#MB:CONFIG {lfo_table: process_mode=[1],}相当）を指定します。この設定により、トラックごとにprocess_modeを変更することが出来ます。

【例】

```
ym"lptbl","p_mode",1
```

6.36. yp[1]：音源制御コマンド群の実行

6.37. #MB:Y_PACK：音源制御コマンド群の定義

【記述例】

```
#MB:ENV_A @ea=1 { peak=15, init=&, | n:1:15, r:8:0 }
#MB:Y_PACK yp=1 {
   cpx:opnum:3,           //オペレータ数を３個にする(OP1...OP3)

   cpx:op1subform:1,      //OP1の波形＝ノコギリ波
   cpx:op1cent:0.0,       //OP1の周波数＝ずれ無し
   cpx:op1dB:0.0,         //OP1の音量＝等倍

   cpx:op2subform:3,      //OP2の波形＝パルス波
   cpx:op2cent:1200+0,    //OP2の周波数＝+1200cent
   cpx:op2dB:-6.0,        //OP2の音量＝-6dB

   cpx:op3subform:3,      //OP3の波形＝パルス波
   cpx:op3cent:1200+7.5,  //OP3の周波数＝+1207.5cent
   cpx:op3dB:-6.0,        //OP3の音量＝-6dB
}
t120 l8 q11,16 @ea1 @@"cpx" yp1
o4 cegefdgg cegefdg&g;
```

このように書くと、

- #MB:Y_PACK によって、定義番号１番に音源制御コマンド群の内容を割り当てています。
- yp1によって、定義番号１番の #MB:Y_PACK の内容を適用。

といった定義と適用を記述できます。

【解説】
音源制御コマンド群の実行コマンド（yp[1]）：
音源制御コマンド群の定義（#MB:Y_PACK）の定義番号を整数で指定します。
定義番号は 0 ～ 1023 の整数です。
未定義の番号を指定するとエラーになります。

音源制御コマンド群の定義（#MB:Y_PACK）：
音源制御コマンド群の実行コマンド（yp[1]）で使用する、音源制御コマンド群を定義します。

音量エンベロープ定義方法の概要は、
・行頭からの記述で、キーワード「#MB:Y_PACK」を書く。
・続けて、スペースを置き、「yp=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られた設定データを書く。
です。

定義番号：
定義番号は、ypコマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 1023 です。

設定データ：

【記述例】

```
#MB:Y_PACK yp=1 {
   pls:opnum:2,           //オペレータ数を２個にする(OP1...OP2)
   pls:op1cent:0,         //OP1の周波数＝0cent
   pls:op1dB:0.0,         //OP1の音量＝等倍
   pls:op2cent:7.5,       //OP2の周波数＝7.5cent
   pls:op2dB:0.0,         //OP2の音量＝等倍
}
```

【解説】
フォーマットは次の通りです。

```
#MB:Y_PACK yp=定義番号 {
     [Y_PACK節],[Y_PACK節],...（1個以上、2048個以下のY_PACK節）
}
```

Y_PACK節：

Y_PACK節は、「:」で区切られた３個１組のパラメータを、「,」区切りで１個以上列挙したものです。
Y_PACK節内の要素の名前を、次のように決め、それぞれについて説明します。

```
    [module]:[y_func]:[value],
```

Y_PACK節の要素[module]：

[module]には、当該Y_PACK節の宛先音源モジュール名を、文字列で指定します。
指定できる文字列は次を参照：

音源モジュールを指す文字列一覧

Y_PACK節の要素[y_func]：

[y_func]には、当該Y_PACK節の宛先に対する機能名称を、文字列で指定します。
指定できる文字列は、各音源モジュール（@@"sin"など）の説明内、

【yコマンドにおける、音源モジュールへの設定】

の項目にある、yコマンドの各機能ごとの宛先文字列がそのまま、ここで指定できる文字列になります。

Y_PACK節の要素[value]：

[value]には、[module][y_func]で確定した yコマンド宛先に与える数値を指定します。
数値の指定範囲は、機能ごとに異なりますので、各音源モジュール説明内、

【yコマンドにおける、音源モジュールへの設定】

の項目を参照してください。

6.38. @dl[1]：ディレイエフェクト

6.39. #MB:DELAY：ディレイエフェクト定義

【記述例】

```
#MB:DELAY @dl=1 { type=infinite, msec=300, dB=-8, }
t120 @@"pls" o4 l8 q6,8 @dl1 cegefdgg cegefdg&g;
```

このように書くと、

- #MB:DELAY によって、定義番号１番にディレイエフェクト内容を割り当てています。
- @dl1によって、定義番号１番の #MB:DELAY の内容を適用。

といった定義と適用を記述できます。

【解説】

ディレイエフェクトコマンド（@dl[1]）：
ディレイエフェクト定義（#MB:DELAY）の定義番号を整数で指定します。
定義番号は０～２５５の整数です。
未定義の番号を指定するとエラーになります。
トラック先頭における初期設定には定義番号は無く、ディレイエフェクトは無効です。

ディレイエフェクト定義（#MB:DELAY）：
ディレイエフェクトコマンド（@dl[1]）で使用する、ディレイエフェクト設定を定義します。

ディレイエフェクト定義方法の概要は、
・行頭からの記述で、キーワード「#MB:DELAY」を書く。
・続けて、スペースを置き、「@dl=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られた設定データを書く。
です。

定義番号：
定義番号は、@dlコマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 255 です。

設定データ：
設定データのフォーマットは、まず最初のパラメータで指定する種別定義（type=...）で変わります。種別定義が必ず先頭で行われる必要があります。

・type=infinite のとき（無限回数）

【記述例】

```
#MB:DELAY @dl=1 {
   type=infinite, msec=300, dB=-8,
}
```

この例では、
　　フィードバックによる無限回数加算、遅延300ms、減衰-8dB
のエフェクトを定義しています。

【解説】
フィードバックによる無限回数ディレイ加算を行います。
フォーマットは次の通りです。

```
#MB:DELAY @dl=定義番号 {
     type=infinite, msec=[], dB=[],
}
```

msec：
１次遅延の時間を設定します。単位は「ミリ秒」です。
設定範囲は、0.05 ～ 5000.0 です。

dB：
１次遅延の振幅を、元に比べて何デシベルにするかを設定します。
設定範囲は、-0.2 ～ -96.0 です。

- type=infinite_bqbpf のとき
- type=infinite_bqbpf_r のとき

【記述例】

```
#MB:DELAY @dl=10 {
   type=infinite_bqbpf, msec=300, dB=-8,
   freq=1000, bw=1.0,
}
#MB:DELAY @dl=11 {
   type=infinite_bqbpf_r, msec=300, dB=-8,
   freq=1000, bw=1.0,
}
```

この例では、
　　定義番号１０番：
　　無限回数加算＆バンドパスフィルタ、遅延300ms、減衰-8dB

　　定義番号１１番：
　　無限回数加算＆バンドパスフィルタ（再帰）、遅延300ms、減衰-8dB
のエフェクトを定義しています。

【解説】

フィードバックによる無限回数ディレイ加算を、バンドパスフィルタを伴って行います。
「infinite_bqbpf」と「infinite_bqbpf_r」の違いは、バンドパスフィルタの掛かり方です。
「_r」が付かない方は、元波形をディレイバッファに送る時にだけバンドパスが掛かるので、バンドパス対象が最初のディレイのみになります。
「_r」が付く方は、フィードバック加算の対象にバンドパスが掛かるので、バンドパスが再帰的に掛かります。
フォーマットは次の通りです。

```
#MB:DELAY @dl=定義番号 {
     type=infinite_bqbpf, msec=[], dB=[], freq=[], bw=[],
}
#MB:DELAY @dl=定義番号 {
     type=infinite_bqbpf_r, msec=[], dB=[], freq=[], bw=[],
}
```

msec：
１次遅延の時間を設定します。単位は「ミリ秒」です。
設定範囲は、0.05 ～ 5000.0 です。

dB：
１次遅延の振幅を、元に比べて何デシベルにするかを設定します。
設定範囲は、-0.2 ～ -96.0 です。

freq：
バンドパスフィルタのカットオフ周波数を設定します。単位は「Hz」です。
設定範囲は、40.0 ～ 384000.0 です。

bw：
バンドパスフィルタの帯域幅（band width）を設定します。
設定範囲は、0.0 より大きい値です。
bw=1のとき、カットオフ周波数から±１オクターブが-3dBとなります。
bw=2のとき、カットオフ周波数から±２オクターブが-3dBとなります。
bwの数値が小さくなるほど、カットオフ周波数からの通過幅が狭くなります。

・type=finite* のとき（有限１～６回）

【記述例】

```
#MB:DELAY @dl=1 {
   type=finite1, msec1=300, dB1=-6,
}
```

```
#MB:DELAY @dl=1 {
   type=finite2,
   msec1=200, dB1=-6,
   msec2=400, dB2=-9,
}
```

```
#MB:DELAY @dl=1 {
   type=finite3,
   msec1=200, dB1=-6,
   msec2=400, dB2=-9,
   msec3=600, dB3=-12,
}
```

```
#MB:DELAY @dl=1 {
   type=finite4,
   msec1=200, dB1=-6,
   msec2=400, dB2=-9,
   msec3=600, dB3=-12,
   msec4=800, dB4=-15,
}
```

```
#MB:DELAY @dl=1 {
   type=finite5,
   msec1=200, dB1=-6,
   msec2=400, dB2=-9,
   msec3=600, dB3=-12,
   msec4=800, dB4=-15,
   msec5=1000, dB5=-18,
}
```

```
#MB:DELAY @dl=1 {
   type=finite6,
   msec1=200, dB1=-6,
   msec2=400, dB2=-9,
   msec3=600, dB3=-12,
   msec4=800, dB4=-15,
   msec5=1000, dB5=-18,
   msec6=1200, dB6=-20,
}
```

【解説】

次数１～次数６の定義に従って、遅延加算を行うディレイエフェクトデータ定義です。

項目名	内容
msec1	１次遅延の時間（ミリ秒）。設定範囲は 0.05 ～ 5000.0
dB1	１次遅延の振幅比（dB）。設定範囲は -0.2 ～ -96.0
msec2	２次遅延の時間（ミリ秒）。設定範囲は 0.05 ～ 5000.0
dB2	２次遅延の振幅比（dB）。設定範囲は -0.2 ～ -96.0
msec3	３次遅延の時間（ミリ秒）。設定範囲は 0.05 ～ 5000.0
dB3	３次遅延の振幅比（dB）。設定範囲は -0.2 ～ -96.0
msec4	４次遅延の時間（ミリ秒）。設定範囲は 0.05 ～ 5000.0
dB4	４次遅延の振幅比（dB）。設定範囲は -0.2 ～ -96.0
msec5	５次遅延の時間（ミリ秒）。設定範囲は 0.05 ～ 5000.0
dB5	５次遅延の振幅比（dB）。設定範囲は -0.2 ～ -96.0
msec6	６次遅延の時間（ミリ秒）。設定範囲は 0.05 ～ 5000.0
dB6	６次遅延の振幅比（dB）。設定範囲は -0.2 ～ -96.0

- type=finite*_bqbpf のとき（BPF&有限１～６回）

【記述例】

```
#MB:DELAY @dl=1 {
   type=finite1_bqbpf, msec1=300, dB1=-6,
   freq=1000, bw=1.0,
}
```

```
#MB:DELAY @dl=1 {
   type=finite2_bqbpf,
   msec1=200, dB1=-6,
   msec2=400, dB2=-9,
   freq=1000, bw=1.0,
}
```

```
#MB:DELAY @dl=1 {
   type=finite3_bqbpf,
   msec1=200, dB1=-6,
   msec2=400, dB2=-9,
   msec3=600, dB3=-12,
   freq=1000, bw=1.0,
}
```

```
#MB:DELAY @dl=1 {
   type=finite4_bqbpf,
   msec1=200, dB1=-6,
   msec2=400, dB2=-9,
   msec3=600, dB3=-12,
   msec4=800, dB4=-15,
   freq=1000, bw=1.0,
}
```

```
#MB:DELAY @dl=1 {
   type=finite5_bqbpf,
   msec1=200, dB1=-6,
   msec2=400, dB2=-9,
   msec3=600, dB3=-12,
   msec4=800, dB4=-15,
   msec5=1000, dB5=-18,
   freq=1000, bw=1.0,
}
```

```
#MB:DELAY @dl=1 {
   type=finite6_bqbpf,
   msec1=200, dB1=-6,
   msec2=400, dB2=-9,
   msec3=600, dB3=-12,
   msec4=800, dB4=-15,
   msec5=1000, dB5=-18,
   msec6=1200, dB6=-20,
   freq=1000, bw=1.0,
}
```

【解説】

次数１～次数６の定義に従って、バンドパスフィルタ付きの遅延加算を行うディレイエフェクトデータ定義です。

項目名	内容
msec1	１次遅延の時間（ミリ秒）。設定範囲は 0.05 ～ 5000.0
dB1	１次遅延の振幅比（dB）。設定範囲は -0.2 ～ -96.0
msec2	２次遅延の時間（ミリ秒）。設定範囲は 0.05 ～ 5000.0
dB2	２次遅延の振幅比（dB）。設定範囲は -0.2 ～ -96.0
msec3	３次遅延の時間（ミリ秒）。設定範囲は 0.05 ～ 5000.0
dB3	３次遅延の振幅比（dB）。設定範囲は -0.2 ～ -96.0
msec4	４次遅延の時間（ミリ秒）。設定範囲は 0.05 ～ 5000.0
dB4	４次遅延の振幅比（dB）。設定範囲は -0.2 ～ -96.0
msec5	５次遅延の時間（ミリ秒）。設定範囲は 0.05 ～ 5000.0
dB5	５次遅延の振幅比（dB）。設定範囲は -0.2 ～ -96.0

項目名	内容
msec6	６次遅延の時間（ミリ秒）。設定範囲は 0.05 ～ 5000.0
dB6	６次遅延の振幅比（dB）。設定範囲は -0.2 ～ -96.0
freq	BPFのカットオフ周波数(Hz)。設定範囲は、40.0 ～ 384000.0
bw	BPFの帯域幅。設定範囲は、0.0 より大きい値です。 bw=1のとき、カットオフ周波数から±１オクターブが-3dB。 bwの数値が小さくなるほど、通過幅が狭くなります。

6.40. @dlz：ディレイエフェクト停止

【記述例】

```
@dlz
```

【解説】

ディレイエフェクト機能を停止します。

6.41. @fd[1]：動的フィルタ

6.42. #MB:FILTER_D：動的フィルタ定義

【記述例】

```
#MB:FILTER_D @fd=1 { type=lpf_h, depth_@ef=0, freq=6000, resonance=0 }
t120 l8 q11,16 @@"saw" @fd1 o4 cegefdgg @fdz cegefdg&g;
```

このように書くと、

- #MB:FILTER_D によって、定義番号１番に動的フィルタ内容を割り当てています。
- @fd1によって、定義番号１番の #MB:FILTER_D の内容を適用。

といった定義と適用を記述できます。

【解説】

動的フィルタコマンド（@fd[1]）：
動的フィルタ定義（#MB:FILTER_D）の定義番号を整数で指定します。
定義番号は 0 ～ 255 の整数です。
未定義の番号を指定するとエラーになります。
トラック先頭における初期設定には定義番号は無く、動的フィルタは無効です。
「動的(dynamic)」と付いているのは、フィルタエンベロープコマンド（@ef[]）や、フィルタＬＦＯコマンド（@lf[]）によって、動的にフィルタ効果を変化させられるためです。

動的フィルタ定義（#MB:FILTER_D）：
動的フィルタコマンド（@fd[1]）で使用する、動的フィルタ設定を定義します。

動的フィルタ定義方法の概要は、
・行頭からの記述で、キーワード「#MB:FILTER_D」を書く。
・続けて、スペースを置き、「@fd=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られた設定データを書く。
です。

定義番号：

定義番号は、@fdコマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 255 です。

設定データ：

【記述例】

```
#MB:FILTER_D @fd=1 {
   type=lpf_h, depth_@ef=0, freq=6000, resonance=0,
}
```

【解説】

フォーマットは次の通りです。

```
#MB:DELAY @fd=定義番号 {
     type=lpf_h, depth_@ef=[], freq=[], resonance=[],
}
```

type：

フィルタの種別を設定します。
設定の選択肢と効果は次の通りです。

選択肢	効果
lpf_h	ローパスフィルタ（高次数）
lpf_l	ローパスフィルタ（低次数）
hpf_h	ハイパスフィルタ（高次数）
hpf_l	ハイパスフィルタ（低次数）
off	フィルタ機能停止

depth_@ef：

フィルタエンベロープ（@ef）の影響深度を指定します。
設定範囲は -100.0 ～ 100.0 です。
depth_@ef は、カットオフの内部管理値（0 ～ 1）に対し、@efのエンベロープ結果値（0 ～ 1）を、depth_@ef/100 倍して加算します。
フィルタに @efエンベロープ変化をつける必要が無い場合には、
depth_@ef に 0 を指定してください。

freq：
カットオフ周波数を指定します。
設定範囲は 40.0 ～ 192000.0 です。

resonance：
レゾナンス（カットオフ周波数付近の強調）の量を指定します。
設定範囲は 0 ～ 100 です。
100 のときは発振します。

6.43. @fdz：動的フィルタ停止

【記述例】

```
@fdz
```

【解説】

動的フィルタ機能を停止します。

6.44. @fs[1]：静的フィルタ

6.45. #MB:FILTER_S：静的フィルタ定義

【記述例】

```
#MB:FILTER_S @fs=1 {
   type=bq_lpf, freq=3000, q=1/(2^0.5),
}
t120 @@"pls" o4 l8 q11,16
@fsz cegr
@fs1 cegcegcegr;
```

このように書くと、

- #MB:FILTER_S によって、定義番号１番に静的フィルタ内容を割り当てています。
- @fs1によって、定義番号１番の #MB:FILTER_S の内容を適用。

といった定義と適用を記述できます。

【解説】

静的フィルタコマンド（@fs[1]）：
静的フィルタ定義（#MB:FILTER_S）の定義番号を整数で指定します。
定義番号は 0 ～ 255 の整数です。
未定義の番号を指定するとエラーになります。
トラック先頭における初期設定には定義番号は無く、静的フィルタは無効です。
「静的(static)」と付いているのは、エンベロープやＬＦＯによる変化機能がなく、固定的なためです。

静的フィルタ定義（#MB:FILTER_S）：
静的フィルタコマンド（@fs[1]）で使用する、静的フィルタ設定を定義します。

静的フィルタ定義方法の概要は、
・行頭からの記述で、キーワード「#MB:FILTER_S」を書く。
・続けて、スペースを置き、「@fs=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られた設定データを書く。
です。

定義番号：

定義番号は、@fsコマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 255 です。

設定データ：

設定データのフォーマットは、まず最初のパラメータで指定する種別定義（type=...）で変わります。種別定義が必ず先頭で行われる必要があります。

• BiQuadとは

「BiQuad」とは、BiQuad型のフィルタ、双二次フィルタです。
双二次フィルタとは、入力信号に対して２つ前までの入力信号と２つ前までの出力信号を使用して出力信号を計算するデジタルフィルタです。
Robert Bristow-Johnson という人が書いた有名な
"Cookbook formulae for audio EQ biquad filter coefficients,"
という文書があります。双二次フィルタは、この文書の技術に基づいています。

・type=bq_lpf のとき

【記述例】

```
#MB:FILTER_S @fs=1 {
   type=bq_lpf, freq=3000, q=1/(2^0.5),
}
```

この例では、
　　BiQuadのローパスフィルタ、カットオフ周波数=3000Hz、Q値=1/√2
のフィルタを定義しています。

【解説】

BiQuadのローパスフィルタを定義します。
フォーマットは次の通りです。

```
#MB:FILTER_S @fs=1 {
   type=bq_lpf, freq=[], q=[],
}
```

freq：
カットオフ周波数を設定します。単位は「Hz」です。
設定範囲は、40.0 ～ 384000.0 です。

q：
フィルタのQ値を設定します。
設定範囲は、0 より大きい値です。
カットオフ周波数付近を強調したくない場合は、Q値を1/√2にします。
Q値を大きくすることでカットオフ周波数でのGainの減少を抑えることができますが、大きくしすぎるとカットオフ周波数付近が強調されてしまいます。

・type=bq_lpf2p のとき

【記述例】

```
#MB:FILTER_S @fs=1 {
   type=bq_lpf2p, freq_lpf2p=3000,
}
```

この例では、
　　BiQuadのローパスフィルタ（２直列）、カットオフ周波数=3000Hz
のフィルタを定義しています。

【解説】

BiQuadのローパスフィルタ（２直列）を定義します。
フォーマットは次の通りです。

```
#MB:FILTER_S @fs=1 {
   type=bq_lpf2p, freq=[],
}
```

フィルタのＱ値は、内部で自動定義されます。
このローパスフィルタでは、BiQuadのローパスを２回かけて、フィルタの次数を上げる効果を得ています（通常-12dB/octのところ、-24dB/oct）。
Ｑ値は２つあり、いずれも内部で最適なものが自動定義されるため、Ｑ値の設定項目はありません。最適なＱ値とは、
Ｑ１：0.5411961001461970;　//　1 / (2 * sin(3π/8))
Ｑ２：1.3065629648763766;　//　1 / (2 * sin(π/8))
です。

freq：

カットオフ周波数を設定します。単位は「Hz」です。
設定範囲は、40.0 ～ 384000.0 です。

・type=bq_hpf のとき

【記述例】

```
#MB:FILTER_S @fs=1 {
   type=bq_hpf, freq=2000, q=1/(2^0.5),
}
```

この例では、
　　BiQuadのハイパスフィルタ、カットオフ周波数=2000Hz、Q値=1/√2
のフィルタを定義しています。

【解説】

BiQuadのハイパスフィルタを定義します。
フォーマットは次の通りです。

```
#MB:FILTER_S @fs=1 {
   type=bq_hpf, freq=[], q=[],
}
```

freq：
カットオフ周波数を設定します。単位は「Hz」です。
設定範囲は、40.0 ～ 384000.0 です。

q：
フィルタのQ値を設定します。
設定範囲は、0 より大きい値です。
カットオフ周波数付近を強調したくない場合は、Q値を1/√2にします。
Q値を大きくすることでカットオフ周波数でのGainの減少を抑えることができますが、大きくしすぎるとカットオフ周波数付近が強調されてしまいます。

・type=bq_bpf のとき

【記述例】

```
#MB:FILTER_S @fs=1 {
   type=bq_bpf, freq=1000, bw=0.8,
}
```

この例では、
　　BiQuadのバンドパスフィルタ、カットオフ周波数=1000Hz、バンド幅=0.8
のフィルタを定義しています。

【解説】

BiQuadのバンドパスフィルタを定義します。
フォーマットは次の通りです。

```
#MB:FILTER_S @fs=1 {
   type=bq_bpf, freq=[], bw=[],
}
```

freq：
カットオフ周波数を設定します。単位は「Hz」です。
（カットオフ周波数は、通す帯域の中心となる周波数になります）
設定範囲は、40.0 ～ 384000.0 です。

bw：
フィルタのバンド幅を設定します。単位はオクターブです。
設定範囲は、0 より大きい値です。

バンド幅は、カットオフ周波数からどこまでの周波数幅を通すかを指定する数値になります。（「通す」とは、出力信号が 0dB ～ -3dB の範囲）
例えば、カットオフ周波数が440Hzで帯域幅が１オクターブなら、220Hz～880Hzで、２オクターブなら110Hz～1760Hzが通す周波数となります。

・type=bq_notch のとき

【記述例】

```
#MB:FILTER_S @fs=1 {
   type=bq_bpf, freq=2000, bw=2.0,
}
```

この例では、
　　BiQuadのノッチフィルタ、カットオフ周波数=2000Hz、バンド幅=2.0
のフィルタを定義しています。

【解説】

BiQuadのノッチフィルタ（範囲非通過）を定義します。
フォーマットは次の通りです。

```
#MB:FILTER_S @fs=1 {
   type=bq_notch, freq=[], bw=[],
}
```

freq：
カットオフ周波数を設定します。単位は「Hz」です。
（カットオフ周波数は、通さない帯域の中心となる周波数になります）
設定範囲は、40.0 ～ 384000.0 です。

bw：
フィルタのバンド幅を設定します。単位はオクターブです。
設定範囲は、0 より大きい値です。

バンド幅は、カットオフ周波数からどこまでの周波数幅を通さないかを指定する数値になります。（「通さない」とは、出力信号が 無音 ～ -3dB の範囲）
例えば、カットオフ周波数が440Hzで帯域幅が１オクターブなら、220Hz～880Hzで、２オクターブなら110Hz～1760Hzが通さない周波数となります。

・type=bq_apf のとき

【記述例】

```
#MB:FILTER_S @fs=1 {
   type=bq_apf, freq=1800, q=0.5,
}
```

この例では、
　　BiQuadのオールパスフィルタ、カットオフ周波数=1800Hz、Q値=0.5
のフィルタを定義しています。

【解説】

BiQuadのオールパスフィルタを定義します。
フォーマットは次の通りです。

```
#MB:FILTER_S @fs=1 {
   type=bq_apf, freq=[], q=[],
}
```

freq：
カットオフ周波数を設定します。単位は「Hz」です。
設定範囲は、40.0 ～ 384000.0 です。

q：
フィルタのQ値を設定します。
設定範囲は、0 より大きい値です。

・type=bq_lsf のとき

【記述例】

```
#MB:FILTER_S @fs=1 {
   type=bq_lsf, freq=500, q=1/(2^0.5), dB=12,
}
```

この例では、
　　BiQuadのローシェルフフィルタ、
　　カットオフ周波数=500Hz、Q値=1/√2、利得=12dB
のフィルタを定義しています。

【解説】

BiQuadのローシェルフフィルタ（低域増幅）を定義します。
フォーマットは次の通りです。

```
#MB:FILTER_S @fs=1 {
   type=bq_lsf, freq=[], q=[], dB=[],
}
```

freq：

カットオフ周波数を設定します。単位は「Hz」です。
（カットオフ周波数はそれ以下の周波数を増幅する周波数になります）
設定範囲は、40.0 ～ 384000.0 です。

カットオフ周波数は厳密には出力信号が指定した増幅量の半分となる周波数で、
増幅はカットオフ周波数より高い周波数から始まります。

q：

フィルタのQ値を設定します。
設定範囲は、0 より大きい値です。
カットオフ周波数付近を強調したくない場合は、Q値を1/√2にします。
Q値を大きくすることで増幅のカーブを急にすることができますが、大きくしすぎるとカットオフ周波数付近が強調されてしまいます。

dB：

フィルタの利得を設定します。単位は「dB」です。
設定範囲は、-18.0 ～ 18.0 です。
負数を指定すると、増幅ではなく、減衰させる機能になります。

・type=bq_hsf のとき

【記述例】

```
#MB:FILTER_S @fs=1 {
   type=bq_hsf, freq=7000, q=1/(2^0.5), dB=12,
}
```

この例では、
　　BiQuadのハイシェルフフィルタ、
　　カットオフ周波数=7000Hz、Q値=1/√2、利得=12dB
のフィルタを定義しています。

【解説】

BiQuadのハイシェルフフィルタ（高域増幅）を定義します。
フォーマットは次の通りです。

```
#MB:FILTER_S @fs=1 {
   type=bq_hsf, freq=[], q=[], dB=[],
}
```

freq：
カットオフ周波数を設定します。単位は「Hz」です。
（カットオフ周波数はそれ以上の周波数を増幅する周波数になります）
設定範囲は、40.0 ～ 384000.0 です。

カットオフ周波数は厳密には出力信号が指定した増幅量の半分となる周波数で、
増幅はカットオフ周波数より低い周波数から始まります。

q：
フィルタのQ値を設定します。
設定範囲は、0 より大きい値です。
カットオフ周波数付近を強調したくない場合は、Q値を1/√2にします。
Q値を大きくすることで増幅のカーブを急にすることができますが、大きくしすぎるとカットオフ周波数付近が強調されてしまいます。

dB：
フィルタの利得を設定します。単位は「dB」です。
設定範囲は、-18.0 ～ 18.0 です。
負数を指定すると、増幅ではなく、減衰させる機能になります。

・type=bq_peaking のとき

【記述例】

```
#MB:FILTER_S @fs=1 {
   type=bq_peaking, freq=1000, bw=2.0, dB=12,
}
```

この例では、
　　BiQuadのピーキングフィルタ、
　　カットオフ周波数=1000Hz、バンド幅=2.0、利得=12dB
のフィルタを定義しています。

【解説】

BiQuadのピーキングフフィルタ（ピンポイント増幅）を定義します。
フォーマットは次の通りです。

```
#MB:FILTER_S @fs=1 {
   type=bq_peaking, freq=[], bw=[], dB=[],
}
```

freq：

カットオフ周波数を設定します。単位は「Hz」です。
（カットオフ周波数は、増幅する帯域の中心となる周波数になります）
設定範囲は、40.0 ～ 384000.0 です。
カットオフ周波数は厳密には出力信号が指定した増幅量の半分となる周波数です。

bw：

フィルタのバンド幅を設定します。単位はオクターブです。
設定範囲は、0 より大きい値です。

バンド幅は、カットオフ周波数からどこまでの周波数幅を増幅するかを指定する数値になります。
例えば、カットオフ周波数が440Hzで帯域幅が１オクターブなら、220Hz～880Hzで、２オクターブなら110Hz～1760Hzが通す周波数となります。

dB：

フィルタの利得を設定します。単位は「dB」です。
設定範囲は、-18.0 ～ 18.0 です。
負数を指定すると、増幅ではなく、減衰させる機能になります。

・type=bq_lsflpf のとき

【記述例】

```
#MB:FILTER_S @fs=1 {
   type=bq_lsflpf,
   freq_lsf=200,  q_lsf=1/(2^0.5), dB_lsf=12,  //low shelf
   freq_lpf=7500, q_lpf=1/(2^0.5),             //low pass
}
```

この例では、
　　BiQuadのローシェルフフィルタ、
　　カットオフ周波数=200Hz、Q値=1/√2、利得=12dB
　　BiQuadのローパスフィルタ、
　　カットオフ周波数=7500Hz、Q値=1/√2
のフィルタを定義しています。

【解説】

BiQuadのローシェルフとローパスを組み合わせたものです。
フォーマットは次の通りです。

```
#MB:FILTER_S @fs=1 {
   type=bq_lsflpf,
   freq_lsf=[], q_lsf=[], dB_lsf=[],
   freq_lpf=[], q_lpf=[],
}
```

freq_lsf：

ローシェルフの低域カットオフ周波数を設定します。単位は「Hz」です。
（低域カットオフ周波数はそれ以下の周波数を増幅する周波数になります）
設定範囲は、40.0 ～ 384000.0 です。

q_lsf：

ローシェルフの低域Q値を設定します。
設定範囲は、0 より大きい値です。

dB_lsf：

ローシェルフの低域利得を設定します。単位は「dB」です。
設定範囲は、-18.0 ～ 18.0 です。
負数を指定すると、増幅ではなく、減衰させる機能になります。

freq_lpf：
ローパスのカットオフ周波数を設定します。単位は「Hz」です。
設定範囲は、40.0 ～ 384000.0 です。

q_lpf：
ローパスのＱ値を設定します。
設定範囲は、0 より大きい値です。
カットオフ周波数付近を強調したくない場合は、Ｑ値を1/√2にします。
Ｑ値を大きくすることでカットオフ周波数でのGainの減少を抑えることができますが、大きくしすぎるとカットオフ周波数付近が強調されてしまいます。

【備考】
このローシェルフとローパスの組み合わせが想定している利用シーンは、ローパスフィルタが必要なＰＣＭ音源に対し、低音増強もさせたいような状況です。

・type=bq_lsflpf2p のとき

【記述例】

```
#MB:FILTER_S @fs=1 {
   type=bq_lsflpf2p,
   freq_lsf=200,  q_lsf=1/(2^0.5), dB_lsf=12,  //low shelf
   freq_lpf2p=3000,                            //low pass
}
```

この例では、
　　BiQuadのローシェルフフィルタ、
　　カットオフ周波数=200Hz、Q値=1/√2、利得=12dB
　　BiQuadのローパスフィルタ（２直列）、
　　カットオフ周波数=3000Hz
のフィルタを定義しています。

【解説】

BiQuadのローシェルフとローパス（２直列）を組み合わせたものです。
フォーマットは次の通りです。

```
#MB:FILTER_S @fs=1 {
   type=bq_lsflpf2p,
   freq_lsf=[], q_lsf=[], dB_lsf=[],
   freq_lpf2p=[],
}
```

ローパスフィルタ（２直列）のQ値は、内部で自動定義されます。
このローパスフィルタでは、BiQuadのローパスを２回かけて、フィルタの次数を上げる効果を得ています（通常-12dB/octのところ、-24dB/oct）。
Q値は２つあり、いずれも内部で最適なものが自動定義されるため、Q値の設定項目はありません。最適なQ値とは、
Ｑ１：0.5411961001461970;　//　1 / (2 * sin(3π/8))
Ｑ２：1.3065629648763766;　//　1 / (2 * sin(π/8))
です。

freq_lsf：

ローシェルフの低域カットオフ周波数を設定します。単位は「Hz」です。
（低域カットオフ周波数はそれ以下の周波数を増幅する周波数になります）
設定範囲は、40.0 ～ 384000.0 です。

q_lsf：
ローシェルフの低域Q値を設定します。
設定範囲は、0 より大きい値です。

dB_lsf：
ローシェルフの低域利得を設定します。単位は「dB」です。
設定範囲は、-18.0 ～ 18.0 です。
負数を指定すると、増幅ではなく、減衰させる機能になります。

freq_lpf2p：
ローパス（２直列）のカットオフ周波数を設定します。単位は「Hz」です。
設定範囲は、40.0 ～ 384000.0 です。

【備考】
このローシェルフとローパス（２直列）の組み合わせが想定している利用シーンは、ローパスフィルタが必要なＰＣＭ音源に対し、低音増強もさせたいような状況です。

・type=bq_eq2band のとき

【記述例】

```
#MB:FILTER_S @fs=1 {
   type=bq_eq2band,
   freq_l=250,  q_l=1/(2^0.5), dB_l=12,  //low shelf
   freq_h=7500, q_h=1/(2^0.5), dB_h=12,  //high shelf
}
```

この例では、
　　BiQuadの２バンドイコライザ、
　　低域カットオフ周波数= 250Hz、低域Ｑ値=1/√2、低域利得=12dB、
　　高域カットオフ周波数=7500Hz、高域Ｑ値=1/√2、高域利得=12dB
のフィルタを定義しています。

【解説】

BiQuadの２バンドイコライザを定義します。
２バンドイコライザは、ローシェルフとハイシェルフを組み合わせたものです。
フォーマットは次の通りです。

```
#MB:FILTER_S @fs=1 {
   type=bq_eq2band,
   freq_l=[], q_l=[], dB_l=[],
   freq_h=[], q_h=[], dB_h=[],
}
```

freq_l：

低域カットオフ周波数を設定します。単位は「Hz」です。
（低域カットオフ周波数はそれ以下の周波数を増幅する周波数になります）
設定範囲は、40.0 ～ 384000.0 です。

q_l：

低域Ｑ値を設定します。
設定範囲は、0 より大きい値です。

dB_l：

低域利得を設定します。単位は「dB」です。
設定範囲は、-18.0 ～ 18.0 です。
負数を指定すると、増幅ではなく、減衰させる機能になります。

freq_h：
高域カットオフ周波数を設定します。単位は「Hz」です。
（高域カットオフ周波数はそれ以上の周波数を増幅する周波数になります）
設定範囲は、40.0 ～ 384000.0 です。

q_h：
高域Q値を設定します。
設定範囲は、0 より大きい値です。

dB_h：
高域利得を設定します。単位は「dB」です。
設定範囲は、-18.0 ～ 18.0 です。
負数を指定すると、増幅ではなく、減衰させる機能になります。

・type=bq_eq3band のとき

【記述例】

```
#MB:FILTER_S @fs=1 {
   type=bq_eq3band,
   freq_l=200,   q_l=1/(2^0.5), dB_l=12,  //low shelf
   freq_m=1000, bw_m=1/(2^0.5), dB_m=6,   //peaking
   freq_h=8000,  q_h=1/(2^0.5), dB_h=12,  //high shelf
}
```

この例では、
　　BiQuadの３バンドイコライザ、
　　低域カットオフ周波数= 200Hz、低域Ｑ値=1/√2、低域利得=12dB、
　　中域カットオフ周波数=1000Hz、中域バンド幅=1/√2、中域利得=6dB、
　　高域カットオフ周波数=8000Hz、高域Ｑ値=1/√2、高域利得=12dB
のフィルタを定義しています。

【解説】

BiQuadの３バンドイコライザを定義します。
３バンドイコライザは、ローシェルフ、ピーキング、ハイシェルフを組み合わせたものです。
フォーマットは次の通りです。

```
#MB:FILTER_S @fs=1 {
   type=bq_eq2band,
   freq_l=[],  q_l=[], dB_l=[],
   freq_m=[], bw_m=[], dB_m=[],
   freq_h=[],  q_h=[], dB_h=[],
}
```

freq_l：
低域カットオフ周波数を設定します。単位は「Hz」です。
（低域カットオフ周波数はそれ以下の周波数を増幅する周波数になります）
設定範囲は、40.0 ～ 384000.0 です。

q_l：
低域Ｑ値を設定します。
設定範囲は、0 より大きい値です。

dB_l：
低域利得を設定します。単位は「dB」です。

設定範囲は、-18.0 ～ 18.0 です。
負数を指定すると、増幅ではなく、減衰させる機能になります。

freq_m：
中域カットオフ周波数を設定します。単位は「Hz」です。
（中域カットオフ周波数は、増幅する帯域の中心となる周波数になります）
設定範囲は、40.0 ～ 384000.0 です。

bw_m：
中域バンド幅を設定します。単位はオクターブです。
設定範囲は、0 より大きい値です。

dB_m：
中域利得を設定します。単位は「dB」です。
設定範囲は、-18.0 ～ 18.0 です。
負数を指定すると、増幅ではなく、減衰させる機能になります。

freq_h：
高域カットオフ周波数を設定します。単位は「Hz」です。
（高域カットオフ周波数はそれ以上の周波数を増幅する周波数になります）
設定範囲は、40.0 ～ 384000.0 です。

q_h：
高域Q値を設定します。
設定範囲は、0 より大きい値です。

dB_h：
高域利得を設定します。単位は「dB」です。
設定範囲は、-18.0 ～ 18.0 です。
負数を指定すると、増幅ではなく、減衰させる機能になります。

・type=off のとき

【記述例】

```
#MB:FILTER_S @fs=1 {
   type=off,
}
```

この例では、
　　フィルタ無効
を定義しています。

【解説】

フィルタ無効を定義します。（通常は使用しません）
フォーマットは次の通りです。（引数はありません）

```
#MB:FILTER_S @fs=1 {
   type=off,
}
```

6.46. @fsz：静的フィルタ停止

【記述例】

```
@fsz
```

【解説】

静的フィルタ機能を停止します。

6.47. #MB:WVM_WAVE：@@"wvm"用：波形データ定義

【記述例】

```
#MB:WVM_WAVE @=1 {
   type=seq_uint4, lv_peak=0x0F, |
   EE77EE77EE77EE770077007007700770077
}
#MB:WVM_WAVE @=2 {
   type=csv_signed, lv_peak=16, |
   0,4,8,12,16,12,8,4,0,-4,-8,-12,-16,-12,-8,-4
}
t120 v15 l8 @@"wvm" @1 cegefdgg @2 cegefdg&g;
```

このように書くと、

- #MB:WVM_WAVE によって、定義番号１番と２番に波形メモリ音源用の波形データを割り当てています。
- @1によって定義番号１番の波形データ、@2によって定義番号２番の波形データを適用して演奏。

といった定義と適用を記述できます。

【解説】

波形メモリ音源（@@"wvm"）用の波形データを定義します（#MB:WVM_WAVE）。

波形データ定義方法の概要は、
・行頭からの記述で、キーワード「#MB:WVM_WAVE」を書く。
・続けて、スペースを置き、「@=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られた設定データを書く。
です。

定義番号：

定義番号は、@コマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 1023 です。

設定データ：

設定データのフォーマットは、まず最初のパラメータで指定する種別定義（type=...）で変わります。種別定義が必ず先頭で行われる必要があります。

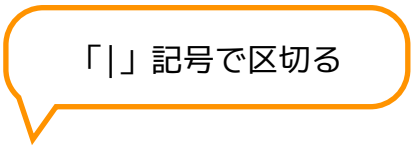
・type=csv_signed のとき

【記述例】

```
#MB:WVM_WAVE @=1 {
   type=csv_signed, lv_peak=16, |
   0,4,8,12,16,12,8,4,0,-4,-8,-12,-16,-12,-8,-4
}
```

【解説】

フォーマットは次の通りです。




```
#MB:WVM_WAVE @=定義番号 {
   type=csv_signed, lv_peak=[], |
   [],[],[]...
}
```

※lv_peak=[]の後の「,」は無くても構いません。
※「|」の後に続くのは、カンマ区切りの数値で表現した波形データです。
※波形データに定義できる要素数の範囲は、２個～４０９６個です。

type=csv_signed：

「type=csv_signed」は、波形データ形式を「カンマ区切り、符号あり」に設定することを意味します。数値の形式は、１０進数の整数または浮動小数、「0x」接頭辞のついた１６進数の整数、およびそれらの四則演算式が使用できます。

lv_peak：

波形データの１要素の最大値を設定します。
設定範囲は、0 より大きい値です。

【備考】変位の作り方
csv_signedのとき：変位をlv_peakで割って-1～1の変位を作ります。

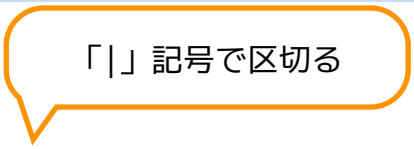
・type=csv_unsigned のとき

【記述例】

```
#MB:WVM_WAVE @=1 {
   type=csv_unsigned, lv_peak=16, |
   0,4,8,12,16,12,8,4
}
```

【解説】

フォーマットは次の通りです。




```
#MB:WVM_WAVE @=定義番号 {
   type=csv_unsigned, lv_peak=[], |
   [],[],[]...
}
```

※lv_peak=[]の後の「,」は無くても構いません。
※「|」の後に続くのは、カンマ区切りの数値で表現した波形データです。
※波形データに定義できる要素数の範囲は、２個～４０９６個です。

type=csv_unsigned：
「type=csv_signed」は、波形データ形式を「カンマ区切り、符号なし」に設定することを意味します。数値の形式は、１０進数の整数または浮動小数、「0x」接頭辞のついた１６進数の整数、およびそれらの四則演算式が使用できます。

lv_peak：
波形データの１要素の最大値を設定します。
設定範囲は、0 より大きい値です。

【備考】変位の作り方
csv_unsignedのとき：変位をlv_peakで割ったものを２倍して、
さらに１を減算して-1～1の変位を作ります。

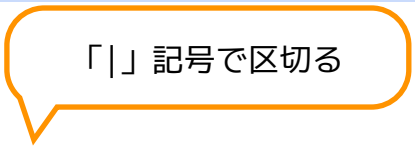
・type=seq_sint4 のとき

【記述例】

```
#MB:WVM_WAVE @=1 {
   type=seq_sint4, lv_peak=8, |
   0123456789abcdef
}
```

【解説】

フォーマットは次の通りです。




```
#MB:WVM_WAVE @=定義番号 {
   type=seq_sint4, lv_peak=[], |
   [][][]...
}
```

※lv_peak=[]の後の「,」は無くても構いません。
※「|」の後に続くのは、区切り文字なしの数値で表現した波形データです。
※波形データに定義できる要素数の範囲は、２個～４０９６個です。

type=seq_sint4：

「type=seq_sint4」は、波形データ形式を「区切り文字なし、符号あり」に設定することを意味します。
波形１要素の数値の形式は、0～9,a～f(A～F)の１文字による１６進数の整数です。波形データにはスペースや改行を含んでも良く、これらは全て読み飛ばされます。

【seq_sint4:文字と値の対応】

文字	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
変位	0	1	2	3	4	5	6	7	-8	-7	-6	-5	-4	-3	-2	-1

lv_peak：

波形データの１要素の最大値を設定します。
設定範囲は、１ ～ ８ です。

【備考】変位の作り方
seq_sint4のとき：変位をlv_peakで割って、-1～1の変位を作ります。

・type=seq_uint4 のとき

【記述例】

```
#MB:WVM_WAVE @=1 {
   type=seq_uint4, lv_peak=15, |
   0123456789abcdef
}
```

【解説】

フォーマットは次の通りです。

「|」記号で区切る

```
#MB:WVM_WAVE @=定義番号 {
   type=seq_uint4, lv_peak=[], |
   [][][]...
}
```

※lv_peak=[]の後の「,」は無くても構いません。
※「|」の後に続くのは、区切り文字なしの数値で表現した波形データです。
※波形データに定義できる要素数の範囲は、２個～４０９６個です。

type=seq_uint4：

「type=seq_uint4」は、波形データ形式を「区切り文字なし、符号なし」に設定することを意味します。
波形１要素の数値の形式は、0～9,a～f(A～F)の１文字による１６進数の整数です。波形データにはスペースや改行を含んでも良く、これらは全て読み飛ばされます。

【seq_uint4:文字と値の対応】

文字	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
変位	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

lv_peak：

波形データの１要素の最大値を設定します。
設定範囲は、1 ～ 15 です。

【備考】変位の作り方
seq_uint4のとき：変位をlv_peakで割ったものを２倍して、
さらに１を減算して-1～1の変位を作ります。

・type=seq_sint8 のとき

【記述例】

```
#MB:WVM_WAVE @=1 {
   type=seq_sint8, lv_peak=0x80 |
   00 1f 3f 5f 7f 80 a0 c0 e0
}
```

【解説】

フォーマットは次の通りです。

「|」記号で区切る

```
#MB:WVM_WAVE @=定義番号 {
   type=seq_sint8, lv_peak=[], |
   [][][]...
}
```

※lv_peak=[]の後の「,」は無くても構いません。
※「|」の後に続くのは、区切り文字なしの数値で表現した波形データです。
※波形データに定義できる要素数の範囲は、２個～４０９６個です。

type=seq_sint8：

「type=seq_sint8」は、波形データ形式を「区切り文字なし、符号あり」に設定することを意味します。
波形１要素の数値の形式は、0～9,a～f(A～F)の２文字１組による１６進数の整数です。波形データにはスペースや改行を含んでも良く、これらは全て読み飛ばされます。

【seq_sint8:文字と値の対応】

文字	00	1f	3f	5f	7f	80	a0	c0	e0	ff
変位	0	31	63	95	127	-128	-96	-64	-32	-1

lv_peak：

波形データの１要素の最大値を設定します。
設定範囲は、1 ～ 128 です。

【備考】変位の作り方
seq_sint8のとき：変位をlv_peakで割って、-1～1の変位を作ります。

・type=seq_uint8 のとき

【記述例】

```
#MB:WVM_WAVE @=1 {
   type=seq_uint8, lv_peak=0xff |
   00 1f 3f 5f 7f 9f bf df ff
}
```

【解説】

フォーマットは次の通りです。

「|」記号で区切る

```
#MB:WVM_WAVE @=定義番号 {
   type=seq_uint8, lv_peak=[], |
   [][][]...
}
```

※lv_peak=[]の後の「,」は無くても構いません。
※「|」の後に続くのは、区切り文字なしの数値で表現した波形データです。
※波形データに定義できる要素数の範囲は、２個～４０９６個です。

type=seq_uint8：

「type=seq_uint8」は、波形データ形式を「区切り文字なし、符号なし」に設定することを意味します。
波形１要素の数値の形式は、0～9,a～f(A～F)の２文字１組による１６進数の整数です。波形データにはスペースや改行を含んでも良く、これらは全て読み飛ばされます。

【seq_uint8:文字と値の対応】

文字	00	1f	3f	5f	7f	9f	bf	df	ff
変位	0	31	63	95	127	159	191	223	255

lv_peak：

波形データの１要素の最大値を設定します。
設定範囲は、1 ～ 255 です。

【備考】変位の作り方
seq_uint8のとき：変位をlv_peakで割ったものを２倍して、
さらに１を減算して-1～1の変位を作ります。

6.48. #MB:FMS_6OP：@@"fms"用：音色データ定義（６オペレータ）

【記述例】

```
#MB:FMS_6OP @=1 {
// IPH IEL    AR D1R D2R  RR D1L  TL  KS   MUL DT1 DT2 DT3 DT4    FB AME MSK
     0, -1,   31,  0,  0,  0,  0, 21,  0,    1,  0,  0,  0,  0,    5,  0,  0,
     0, -1,   18, 15,  9,  7,  3,  0,  0,    1,  0,  0,  0,  0,    0,  0,  0,
     0, -1,   31,  0,  0,  0,  0, 21,  0,    1,  0,  0, -5,  0,    5,  0,  0,
     0, -1,   18, 15,  9,  7,  3,  8,  0,    1,  0,  0, -5,  0,    0,  0,  0,
     0, -1,   31,  0,  0,  0,  0, 20,  0,    1,  0,  0,  5,  0,    0,  0,  0,
     0, -1,   28, 15,  9,  7,  3,  0,  0,    1,  0,  0,  5,  0,    0,  0,  0,
|    con=45,
}
@@"fms" @1 q6,8 o5 cdefg;
```

【解説】

ＦＭ音源（@@"fms"）の音色データを、６オペレータモードで定義します。

音色データ定義方法の概要は、
・行頭からの記述で、キーワード「#MB:FMS_6OP」を書く。
・続けて、スペースを置き、「@=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られた設定データを書く。
です。

定義番号：

定義番号は、@コマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 1023 です。
音色番号 0 番には初期データが入っています（上書可）。
未定義の音色番号を参照した場合には、0 番が読み出されます。

設定データ：

設定データの各パラメータは、カンマで区切って記述します。
最終パラメータの後のカンマは、あってもなくても構いません。
スペースや改行は読み飛ばされます。

６オペレータモードの音色データ定義では、
・１オペあたり１７個のデータを、６オペレータ分定義
・「|」記号で区切る
・音色全体に関わる設定を「項目名=値,」の形式で列挙する
の要領で記述します。

[#MB:FMS_6OP]定義内容一覧

OP1	IPH	IEL	AR	D1R	D2R	RR	D1L	TL	KS	MUL	DT1	DT2	DT3	DT4	FB	AME	MSK
OP2	IPH	IEL	AR	D1R	D2R	RR	D1L	TL	KS	MUL	DT1	DT2	DT3	DT4	FB	AME	MSK
OP3	IPH	IEL	AR	D1R	D2R	RR	D1L	TL	KS	MUL	DT1	DT2	DT3	DT4	FB	AME	MSK
OP4	IPH	IEL	AR	D1R	D2R	RR	D1L	TL	KS	MUL	DT1	DT2	DT3	DT4	FB	AME	MSK
OP5	IPH	IEL	AR	D1R	D2R	RR	D1L	TL	KS	MUL	DT1	DT2	DT3	DT4	FB	AME	MSK
OP6	IPH	IEL	AR	D1R	D2R	RR	D1L	TL	KS	MUL	DT1	DT2	DT3	DT4	FB	AME	MSK
全体	| con=[], reso=[],																

【備考】
ＦＭ音源の音色作成に関わる用語説明

con：（この項目は記述必須です）
オペレータ同士の接続設定を、コネクション番号で指定します。
設定範囲は、0 ～ 63 です。

reso：
@@"fms"ＦＭ音源モジュールの、フェーズジェネレータの解像度設定を、レゾリューション番号で指定します。
設定範囲は、0 ～ 8 です。
この設定による効果は、@mコマンドと同様です。
この項目の記述を省略した場合、音色設定時に解像度設定を行いません。

6.49. #MB:FMS_4OP：@@"fms"用：音色データ定義（４オペレータ）

【記述例】

```
#MB:FMS_4OP @=1 {
// IPH IEL    AR D1R D2R  RR D1L  TL  KS   MUL DT1 DT2 DT3 DT4    FB AME MSK
     0, -1,   31,  0,  0,  0,  0, 21,  0,    1,  0,  0,  0,  0,    5,  0,  0,
     0, -1,   18, 15,  9,  7,  3,  0,  0,    1,  0,  0,  0,  0,    0,  0,  0,
     0, -1,   31,  0,  0,  0,  0, 20,  0,    1,  0,  0,  5,  0,    0,  0,  0,
     0, -1,   28, 15,  9,  7,  3,  0,  0,    1,  0,  0,  5,  0,    0,  0,  0,
|    con=4,
}
@@"fms" @1 q6,8 o5 cdefg;
```

【解説】

ＦＭ音源（@@"fms"）の音色データを、４オペレータモードで定義します。

音色データ定義方法の概要は、
・行頭からの記述で、キーワード「#MB:FMS_4OP」を書く。
・続けて、スペースを置き、「@=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られた設定データを書く。
です。

定義番号：

定義番号は、@コマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 1023 です。
音色番号 0 番には初期データが入っています（上書可）。
未定義の音色番号を参照した場合には、0 番が読み出されます。

設定データ：

設定データの各パラメータは、カンマで区切って記述します。
最終パラメータの後のカンマは、あってもなくても構いません。
スペースや改行は読み飛ばされます。

４オペレータモードの音色データ定義では、
・１オペあたり１７個のデータを、４オペレータ分定義
・「|」記号で区切る
・音色全体に関わる設定を「項目名=値,」の形式で列挙する
の要領で記述します。

[#MB:FMS_4OP]定義内容一覧

OP1	IPH	IEL	AR	D1R	D2R	RR	D1L	TL	KS	MUL	DT1	DT2	DT3	DT4	FB	AME	MSK
OP2	IPH	IEL	AR	D1R	D2R	RR	D1L	TL	KS	MUL	DT1	DT2	DT3	DT4	FB	AME	MSK
OP3	IPH	IEL	AR	D1R	D2R	RR	D1L	TL	KS	MUL	DT1	DT2	DT3	DT4	FB	AME	MSK
OP4	IPH	IEL	AR	D1R	D2R	RR	D1L	TL	KS	MUL	DT1	DT2	DT3	DT4	FB	AME	MSK
全体	| con=[], reso=[],																

【備考】
ＦＭ音源の音色作成に関わる用語説明

con：（この項目は記述必須です）
オペレータ同士の接続設定を、コネクション番号で指定します。
設定範囲は、0 ～ 13 です。（通常0～7、拡張8～13）

reso：
@@"fms"ＦＭ音源モジュールの、フェーズジェネレータの解像度設定を、レゾリューション番号で指定します。
設定範囲は、0 ～ 8 です。
この設定による効果は、@mコマンドと同様です。
この項目の記述を省略した場合、音色設定時に解像度設定を行いません。

6.50. #MB:FMS_2OP：@@"fms"用：音色データ定義（２オペレータ）

【記述例】

```
#MB:FMS_2OP @=1 {
// IPH IEL    AR D1R D2R  RR D1L  TL  KS   MUL DT1 DT2 DT3 DT4    FB AME MSK
     0, -1,   31,  0,  0,  0,  0, 21,  0,    1,  0,  0,  0,  0,    5,  0,  0,
     0, -1,   18, 15,  9,  7,  3,  0,  0,    1,  0,  0,  0,  0,    0,  0,  0,
|    con=0,
}
@@"fms" @1 q6,8 o5 cdefg;
```

【解説】

ＦＭ音源（@@"fms"）の音色データを、２オペレータモードで定義します。

音色データ定義方法の概要は、
・行頭からの記述で、キーワード「#MB:FMS_2OP」を書く。
・続けて、スペースを置き、「@=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られた設定データを書く。
です。

定義番号：

定義番号は、@コマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 1023 です。
音色番号 0 番には初期データが入っています（上書可）。
未定義の音色番号を参照した場合には、0 番が読み出されます。

設定データ：

設定データの各パラメータは、カンマで区切って記述します。
最終パラメータの後のカンマは、あってもなくても構いません。
スペースや改行は読み飛ばされます。

２オペレータモードの音色データ定義では、
・１オペあたり１７個のデータを、２オペレータ分定義
・「|」記号で区切る
・音色全体に関わる設定を「項目名=値,」の形式で列挙する
の要領で記述します。

[#MB:FMS_2OP]定義内容一覧

OP1	IPH	IEL	AR	D1R	D2R	RR	D1L	TL	KS	MUL	DT1	DT2	DT3	DT4	FB	AME	MSK
OP2	IPH	IEL	AR	D1R	D2R	RR	D1L	TL	KS	MUL	DT1	DT2	DT3	DT4	FB	AME	MSK
全体	| con=[], reso=[],																

【備考】
ＦＭ音源の音色作成に関わる用語説明

con：（この項目は記述必須です）
オペレータ同士の接続設定を、コネクション番号で指定します。
設定範囲は、0 ～ 1 です。

reso：
@@"fms"ＦＭ音源モジュールの、フェーズジェネレータの解像度設定を、レゾリューション番号で指定します。
設定範囲は、0 ～ 8 です。
この設定による効果は、@mコマンドと同様です。
この項目の記述を省略した場合、音色設定時に解像度設定を行いません。

6.51. #MB:FMS_OPM：@@"fms"用：音色データ定義（ＯＰＭ互換）

【記述例】

```
#MB:FMS_OPM @=1 {
//  AR D1R D2R  RR D1L  TL  KS MUL DT1 DT2 AME
    31,  0,  0,  0,  0, 21,  0,  1,  3,  0,  0,
    18, 15,  9,  7,  3,  0,  0,  1,  3,  0,  0,
    31,  0,  0,  0,  0, 20,  0,  1,  7,  0,  0,
    28, 15,  9,  7,  3,  0,  0,  1,  7,  0,  0,
|   con=4, fb=5,
}
@@"fms" @1 q6,8 o5 cdefg;
```

【解説】

ＦＭ音源（@@"fms"）の音色データを、ＯＰＭ互換モードで定義します。

音色データ定義方法の概要は、
・行頭からの記述で、キーワード「#MB:FMS_OPM」を書く。
・続けて、スペースを置き、「@=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られた設定データを書く。
です。

定義番号：

定義番号は、@コマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 1023 です。
音色番号 0 番には初期データが入っています（上書可）。
未定義の音色番号を参照した場合には、0 番が読み出されます。

設定データ：

設定データの各パラメータは、カンマで区切って記述します。
最終パラメータの後のカンマは、あってもなくても構いません。
スペースや改行は読み飛ばされます。

ＯＰＭ互換モードの音色データ定義では、
・１オペあたり１１個のデータを、４オペレータ分定義
・「|」記号で区切る
・音色全体に関わる設定を「項目名=値,」の形式で列挙する
の要領で記述します。

[#MB:FMS_OPM]定義内容一覧

OP1	AR	D1R	D2R	RR	D1L	TL	KS	MUL	DT1	DT2	AME
OP2	AR	D1R	D2R	RR	D1L	TL	KS	MUL	DT1	DT2	AME
OP3	AR	D1R	D2R	RR	D1L	TL	KS	MUL	DT1	DT2	AME
OP4	AR	D1R	D2R	RR	D1L	TL	KS	MUL	DT1	DT2	AME
全体	| con=[], fb=[], reso=[], wf=[], lfrq=[], pmd=[], amd=[], pmsams=[], sync=[],										

【備考】
ＦＭ音源の音色作成に関わる用語説明

con：（この項目は記述必須です）
オペレータ同士の接続設定を、コネクション番号で指定します。
設定範囲は、0 ～ 13 です。（通常0～7、拡張8～13）

fb：（この項目は記述必須です）
「OP1」のフィードバックレベルを指定します。
設定範囲は、0 ～ 15 です。（通常0～7、拡張8～15）

reso：
@@"fms"ＦＭ音源モジュールの、フェーズジェネレータの解像度設定を、レゾリューション番号で指定します。
設定範囲は、0 ～ 8 です。
この設定による効果は、@mコマンドと同様です。
この項目の記述を省略した場合、当機能の設定を行いません。

wf：
ＯＰＭ互換モードのＨＬＦＯ設定における、wave formの設定を行います。
設定範囲は、0 ～ 3 です。
この項目の記述を省略した場合、当機能の設定を行いません。

lfrq：
ＯＰＭ互換モードのＨＬＦＯ設定における、lfo frequencyの設定を行います。
設定範囲は、0 ～ 255 です。
この項目の記述を省略した場合、当機能の設定を行いません。

pmd：
ＯＰＭ互換モードのＨＬＦＯ設定における、pitch modulation depthの設定を行います。
設定範囲は、0 ～ 127 です。
この項目の記述を省略した場合、当機能の設定を行いません。

amd：
ＯＰＭ互換モードのＨＬＦＯ設定における、amplitude modulation depthの設定を行います。
設定範囲は、0 ～ 127 です。
この項目の記述を省略した場合、当機能の設定を行いません。

pmsams：
ＯＰＭ互換モードのＨＬＦＯ設定における、pitch modulation sensitivityと、amplitude modulation sensitivityの設定を行います。
設定範囲は、8bit整数で、
　　上位4bitがpms（0～7）
　　下位4bitがams（0～3）
です。
例えば、pms=7、ams=3のとき、設定値は 0x73 になります。
この項目の記述を省略した場合、当機能の設定を行いません。

sync：
ＯＰＭ互換モードのＨＬＦＯ設定における、synchronizationの設定を行います。
設定範囲は、0 ～ 1 です。
この項目の記述を省略した場合、当機能の設定を行いません。

6.52. #MB:FMS_OPNA：@@"fms"用：音色データ定義（ＯＰＮＡ互換）

【記述例】

```
#MB:FMS_OPNA @=1 {
//  AR  DR  SR  RR  SL  TL  KS MUL  DT AME
    31,  0,  0,  0,  0, 21,  0,  1,  3,  0,
    18, 15,  9,  7,  3,  0,  0,  1,  3,  0,
    31,  0,  0,  0,  0, 20,  0,  1,  7,  0,
    28, 15,  9,  7,  3,  0,  0,  1,  7,  0,
|   con=4, fb=5,
}
@@"fms" @1 q6,8 o5 cdefg;
```

【解説】

ＦＭ音源（@@"fms"）の音色データを、ＯＰＮＡ互換モードで定義します。

音色データ定義方法の概要は、
・行頭からの記述で、キーワード「#MB:FMS_OPNA」を書く。
・続けて、スペースを置き、「@=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られた設定データを書く。
です。

定義番号：

定義番号は、@コマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 1023 です。
音色番号 0 番には初期データが入っています（上書可）。
未定義の音色番号を参照した場合には、0 番が読み出されます。

設定データ：

設定データの各パラメータは、カンマで区切って記述します。
最終パラメータの後のカンマは、あってもなくても構いません。
スペースや改行は読み飛ばされます。

ＯＰＮＡ互換モードの音色データ定義では、
・１オペあたり１０個のデータを、４オペレータ分定義
・「|」記号で区切る
・音色全体に関わる設定を「項目名=値,」の形式で列挙する
の要領で記述します。

[#MB:FMS_OPM]定義内容一覧-1

OP1	AR	DR	SR	RR	SL	TL	KS	MUL	DT	AME
OP2	AR	DR	SR	RR	SL	TL	KS	MUL	DT	AME
OP3	AR	DR	SR	RR	SL	TL	KS	MUL	DT	AME
OP4	AR	DR	SR	RR	SL	TL	KS	MUL	DT	AME
全体	| con=[], fb=[], reso=[], lfrq=[], amspms=[], sync=[],									

【備考】
ＦＭ音源の音色作成に関わる用語説明

con：（この項目は記述必須です）
オペレータ同士の接続設定を、コネクション番号で指定します。
設定範囲は、0 ～ 13 です。（通常0～7、拡張8～13）

fb：（この項目は記述必須です）
「OP1」のフィードバックレベルを指定します。
設定範囲は、0 ～ 15 です。（通常0～7、拡張8～15）

reso：
@@"fms"ＦＭ音源モジュールの、フェーズジェネレータの解像度設定を、レゾリューション番号で指定します。
設定範囲は、0 ～ 8 です。
この設定による効果は、@mコマンドと同様です。
この項目の記述を省略した場合、当機能の設定を行いません。

lfrq：
ＯＰＮＡ互換モードのＨＬＦＯ設定における、lfo frequencyの設定を行います。
設定範囲は、0 ～ 7 です。
この項目の記述を省略した場合、当機能の設定を行いません。

amspms：
ＯＰＮＡ互換モードのＨＬＦＯ設定における、amplitude modulation sensitivityと、pitch modulation sensitivityの設定を行います。
設定範囲は、8bit整数で、
　　上位4bitがams（0～3）
　　下位4bitがpms（0～7）
です。
例えば、ams=3、pms=7のとき、設定値は 0x37 になります。
この項目の記述を省略した場合、当機能の設定を行いません。

sync：
ＯＰＮＡ互換モードのＨＬＦＯ設定における、synchronizationの設定を行います。
設定範囲は、0 ～ 1 です。
この項目の記述を省略した場合、当機能の設定を行いません。

6.53. #MB:FMS_OPN：@@"fms"用：音色データ定義（ＯＰＮ互換）

【記述例】

```
#MB:FMS_OPN @=1 {
//  AR  DR  SR  RR  SL  TL  KS MUL  DT
    31,  0,  0,  0,  0, 21,  0,  1,  3,
    18, 15,  9,  7,  3,  0,  0,  1,  3,
    31,  0,  0,  0,  0, 20,  0,  1,  7,
    28, 15,  9,  7,  3,  0,  0,  1,  7,
|   con=4, fb=5,
}
@@"fms" @1 q6,8 o5 cdefg;
```

【解説】

ＦＭ音源（@@"fms"）の音色データを、ＯＰＮ互換モードで定義します。

音色データ定義方法の概要は、
・行頭からの記述で、キーワード「#MB:FMS_OPN」を書く。
・続けて、スペースを置き、「@=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られた設定データを書く。
です。

定義番号：

定義番号は、@コマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 1023 です。
音色番号 0 番には初期データが入っています（上書可）。
未定義の音色番号を参照した場合には、0 番が読み出されます。

設定データ：

設定データの各パラメータは、カンマで区切って記述します。
最終パラメータの後のカンマは、あってもなくても構いません。
スペースや改行は読み飛ばされます。

ＯＰＮ互換モードの音色データ定義では、
・１オペあたり９個のデータを、４オペレータ分定義
・「|」記号で区切る
・音色全体に関わる設定を「項目名=値,」の形式で列挙する
の要領で記述します。

[#MB:FMS_OPM]定義内容一覧-1-1

OP1	AR	DR	SR	RR	SL	TL	KS	MUL	DT
OP2	AR	DR	SR	RR	SL	TL	KS	MUL	DT
OP3	AR	DR	SR	RR	SL	TL	KS	MUL	DT
OP4	AR	DR	SR	RR	SL	TL	KS	MUL	DT
全体	| con=[], fb=[], reso=[],								

【備考】
ＦＭ音源の音色作成に関わる用語説明

con：（この項目は記述必須です）
オペレータ同士の接続設定を、コネクション番号で指定します。
設定範囲は、0 ～ 13 です。（通常0～7、拡張8～13）

fb：（この項目は記述必須です）
「OP1」のフィードバックレベルを指定します。
設定範囲は、0 ～ 15 です。（通常0～7、拡張8～15）

reso：
@@"fms"ＦＭ音源モジュールの、フェーズジェネレータの解像度設定を、レゾリューション番号で指定します。
設定範囲は、0 ～ 8 です。
この設定による効果は、@mコマンドと同様です。
この項目の記述を省略した場合、当機能の設定を行いません。

6.54. ＦＭ音源の音色作成に関する用語説明

Ｖ３ＭＭＬで扱うＦＭ音源（@@"fms"）はＯＰＭの上位互換です。
６つのオペレータを持っており、３種のオペレータモード（２／４／６）において音色を作成できます。
加えて、ＯＰＭのハードウェアに備わっていたＬＦＯ機能もサポートしており、この機能は便宜上ハードＬＦＯ（ＨＬＦＯ）と呼びます。

オペレータ（operator）

オペレータとは、１個の正弦波発生器です。
ＦＭ音源の音色作成では、各オペレータに対してパラメータを設定します。
オペレータの各パラメータは、次の２つに大別されます。

エンベロープジェネレータ部（振幅の時系列操作）
フェーズジェネレータ部（周波数と位相の操作）

さらに、「オペレータ同士を接続」することで周波数変調を行いますが、
変調される側のオペレータを「キャリア」
変調する側のオペレータを「モジュレータ」
と呼びます。実際に音を出すオペレータは「キャリア」です。

・ＨＬＦＯ（hardware low frequency oscillator）

ＨＬＦＯは、ＦＭ音源（@@"fms"）モジュール側でサポートするＬＦＯ機能です。
ＯＰＭ互換モードと、ＯＰＮＡ互換モードがあります。
ＬＦＯとは、ビブラート（周波数をゆらす）や、トレモロ（振幅をゆらす）を行う機能です。

ＯＰＭ互換モードのＨＬＦＯは、
　　WF、LFRQ、PMD、AMD、PMS、AMS、SYNC
の７種類で設定します。

WF：wave form
　　ＯＰＭ互換ＨＬＦＯで使用する波形を選択します。
　　指定範囲は 0～3。
　　0：のこぎり波
　　1：矩形波
　　2：三角波
　　3：ランダム波（ノイズ）

LFRQ：lfo frequency
　　ＯＰＭ互換ＨＬＦＯの周波数。範囲は 0～255。

PMD：pitch modulation depth
　　音程深度を指定。範囲は 0～127。

AMD：amplitude modulation depth
　　音量深度を指定。範囲は 0～127。

PMS：pitch modulation sensitivity
　　音程感度を指定。範囲は 0～7。

AMS：amplitude modulation sensitivity
　　音量感度を指定。範囲は 0～3。

SYNC：synchronization
　　ＨＬＦＯ位相のノートオン同期機能。範囲は 0～1。

ＯＰＮＡ互換モードのＨＬＦＯは、
　　LFRQ、PMS、AMS、SYNC
の４種類で設定します。波形パターンは正弦波のみです。

LFRQ：lfo frequency
　　ＯＰＮＡ互換ＨＬＦＯの周波数。範囲は 0～7。

PMS：pitch modulation sensitivity
　　音程感度を指定。範囲は 0～7。

AMS：amplitude modulation sensitivity
　　音量感度を指定。範囲は 0～3。

SYNC：synchronization
　　ＨＬＦＯ位相のノートオン同期機能。範囲は 0～1。

・エンベロープジェネレータ部

エンベロープでは、オペレータごとに、振幅の時系列操作の設定を行います。
パラメータは、
　　AR、D1R、D2R、RR、D1L、TL、KS、AME、MSK、IEL
の１１種類です。

AR：attack rate（0～31）
　　アタックレートは、ノートオンから最大レベルになるまでの速さです。
　　設定値が大きいほど速い立ち上がりになります。
　　最大レベルとは、TLで設定する出力レベルに相当します。

D1R：decay 1 rate（0～31）
　　ディケイ１レートは、アタックレート終了直後の減衰速度です。
　　設定値が大きいほど速い減衰になります。
　　ＯＰＮ系では「DR」と呼ばれます。

D2R：decay 2 rate（0～31）
　　ディケイ２レートは、ディケイ１レート終了直後の減衰速度です。
　　設定値が大きいほど速い減衰になります。
　　ＯＰＮ系では「SR(sustain rate)」と呼ばれます。

RR：release rate（0～15）
　　リリースレートは、ノートオフから最小レベルになるまでの速度です。
　　設定値が大きいほど速い減衰になります。

D1L：decay 1 level（0～15）
　　ディケイ１レベルは、ディケイ１レートからディケイ２レートへ
　　移行する境界となる出力レベルです。
　　0 のとき 0dB です。（結果的にディケイ１レートがスキップされます）
　　1 あたり -3dB で、境界となるレベルが変化します。
　　15 のときは特例で、-93dB になります。（-45dB からさらに -48dB）
　　15 のときは結果的にディケイ２レートが無効になります。
　　ＯＰＮ系では「SL(sustain level)」と呼ばれます。

TL：total level（0～127）
　　トータルレベルは、振幅の最大出力レベルです。
　　アタックレートにおける最大レベルに相当します。
　　設定値が大きいほど出力が小さくなります。
　　0 のとき 0dB です。（出力最大）
　　1 あたり -0.75dB で出力レベルが変化します。
　　127 のとき -95.25dBです。（出力最小）
　　TLは小数以下の指定も受け付けますが、小数以下１１ビットの

固定小数点数の精度に丸められて応答します。
これは内部の振幅値テーブルサイズ（解像度）の都合によるものです。

KS：key scale（0～3）
キースケールは、発音音程に追従して、AR, D1R, D2R, RR のレートを
自動的に内部で変化させる機能です。
0 でキースケールしません。
指定値が大きくなるほど、高い音程でエンベロープ変化が速くなります。

AME：amplitude modulation enable switch（0～1）
AMEは、ＨＬＦＯの振幅変化の対象となるオペレータを設定します。
0 のとき、ＨＬＦＯ振幅変化の対象にしません。
1 のとき、ＨＬＦＯ振幅変化の対象とします。
この機能により、モジュレータのみ、もしくはキャリアのみに
振幅変調を与えるような選択が可能になります。

MSK：operator mask（0～1）
オペレータマスクは、対象オペレータの出力をマスクするかどうか
を設定します。
0 のとき、通常通りのオペレータ出力。
1 のとき、オペレータ出力をマスクします。

IEL：initial envelope level（-1 または 0～128）
IELは、ノートオンによるエンベロープ開始時（アタック開始時）
の出力レベルを設定します。
-1 のとき、ノートオン直前の出力レベルを引き継ぎます。
IELが使用できない音色定義では IEL は -1 に内部で設定されます。
0～128 の範囲では、アタック開始時の出力レベルを強制的に設定します。
設定値と初期出力レベルの対応は、次の通りです。
0 のとき、0dB からアタック開始。
128 のとき、-96dB からアタック開始。
（１増えるごとに -0.75dB された出力レベルからのアタック開始）

・フェーズジェネレータ部

オペレータごとに、周波数と位相の操作の設定を行います。
パラメータは、
　　MUL、DT1、DT2、DT3、DT4、FB、IPH
の７種類です。

MUL：multiple（0～32）（特殊指定：-1）
　　マルチプルは、ノートオン周波数への倍率を指定します。
　　少数以下の指定、分数による指定も受け付けます。
　　ただし、0 は 0.5 倍とみなされます。（OPM/OPN(A)互換のため）
　　特殊指定で、-1 を指定すると 0倍 になります。
　　0倍の指定は、DT4 が使用できる音色設定において、
　　DT4 の周波数加算指定を、ノートオン周波数に左右されない
　　固定的な発音周波数として利用する使い方を想定しています。
　　（DT4 が使用できない音色定義では、-1 の特殊指定は使えません）

DT1：detune 1（0～7）
　　デチューン１は、周波数を微妙に ずらし ます。
　　1～3 のとき、数値が大きいほど周波数が高くなります。
　　5～7のとき、数値が大きいほど周波数が低くなります。
　　0,4のとき、周波数の変化を与えません。
　　ＯＰＮ系では単に「DT(detune)」と呼ばれます。
　　※デチューン１による周波数ずれは微小です。

DT2：detune 2（0～3）
　　デチューン２は、周波数を大きく ずらし ます。
　　指定値と周波数倍率の対応は次の通りです。
　　0 のとき、1.00倍
　　1 のとき、1.41倍
　　2 のとき、1.57倍
　　3 のとき、1.73倍

DT3：detune 3（-9600～9600）
　　デチューン３は、周波数を cent単位で ずらし ます。
　　cent単位では、100あたり半音、1200あたり１オクターブずれます。
　　0 で周波数変化なしです。
　　小数以下の指定、分数による指定も受け付けます。
　　DT3 が使用できない音色定義では、DT3 は 0 として内部で設定されます。

DT4：detune 4（-1048576～1048576）
　　デチューン４は、周波数を Hz単位で ずらします。
　　0 で周波数変化なしです。

小数以下の指定、分数による指定も受け付けます。
MUL の特殊指定で、ノートオン周波数の 0倍 になっている時は、
DT4 で指定する周波数がそのままノートオン周波数になります。
DT4 が使用できない音色定義では、DT4 は 0 として内部で設定されます。

FB：feedback level（0～15）
フィードバックは、自らのオペレータ出力を変調入力とする機能です。
OPM/OPN(A)互換モードでは、オペレータ１のみが FB 機能の対象ですが、
それ以外のモードでは、全オペレータが対象です。
0 を指定した場合は、FB は無効です。
0 より大きい数値を指定した場合、指定値を n とすると、
$FB = 2^{(n-5)}\ \pi$
となり、指定値が大きくなるほど強い FB が掛かります。
（n=7 のとき FB=4π）
少数以下の指定、分数による指定も受け付けます。

IPH：initial phase（0.0以上～1.0未満 または -1.0以上～0.0未満）
イニシャル・フェーズは、ノートオン時の位相を設定します。
小数以下の指定、分数による指定も受け付けます。
0.0以上～1.0未満のとき
　ノートオンの都度、位相を（指定値×２π）にします。
　OPM/OPN(A)互換の音色定義では、IPH は 0 として内部で設定されます。
-1.0以上～0.0未満のとき
　ノートオンの都度、位相を設定しません。
　ただし、音色設定時など、IPHを設定した時にだけ、
　位相を（指定値×(-1)×２π）にします。
　このとき、指定値が（-1）の場合、位相は0と同様です。

・オペレータ同士の接続

CON：connection

コネクションは、オペレータ同士の接続形態
（キャリアとモジュレータの決定、および変調パターン選択）
を指定します。

６オペレータモードの接続形態
４オペレータモードの接続形態
２オペレータモードの接続形態

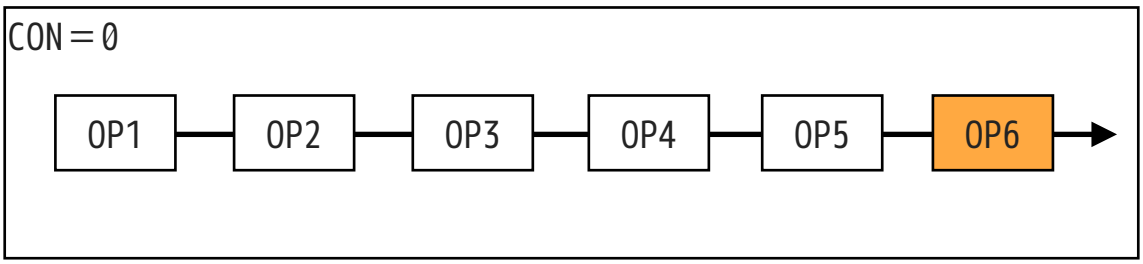
6.55. ＦＭ音源６オペレータモードの接続形態

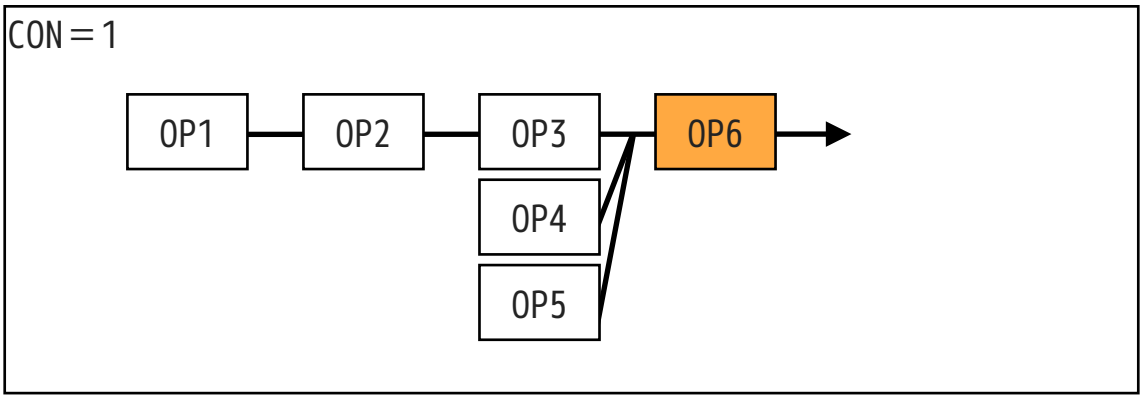
６オペレータモード音色設定
　　#MB:FMS_6OP
における、CON（connection：オペレータの接続パターン）の内容を示します。

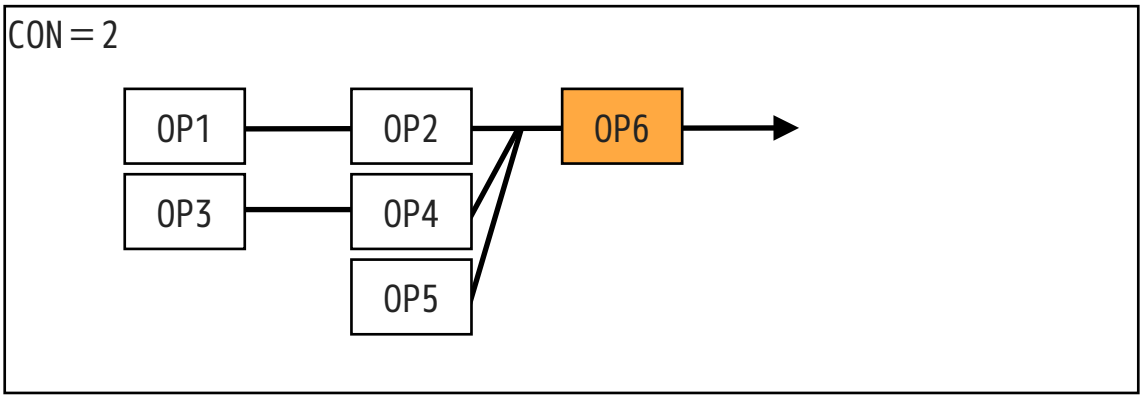
色付きのオペレータがキャリアで、それ以外がモジュレータです。
キャリアが複数ある CON では、コーラスや和音の効果を出すことも可能です。
全てのオペレータにセルフフィードバック機能があります。

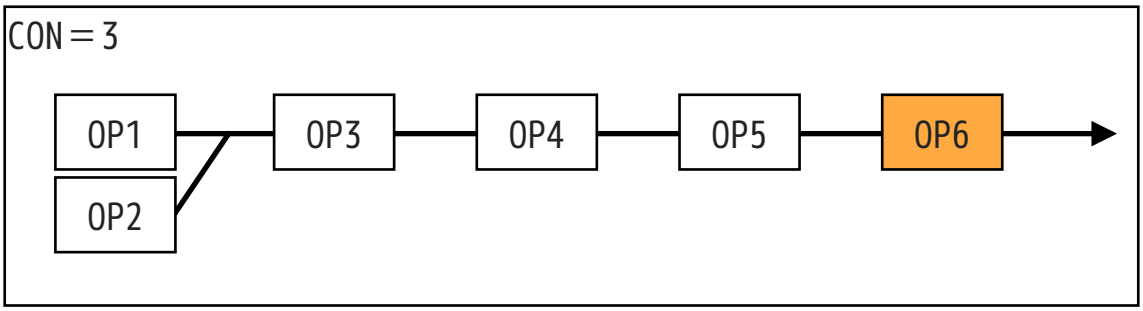
６オペレータモードの CON は、0～63 で指定します。

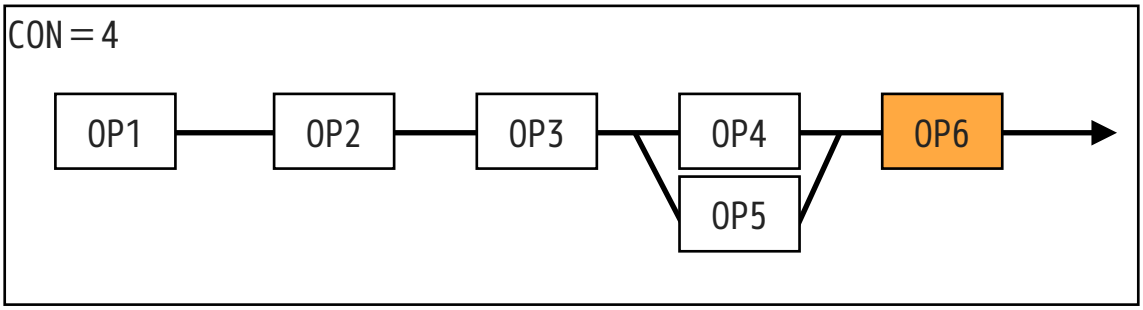
グループ１：CON=0～9：[1,2,3,4,5,6]

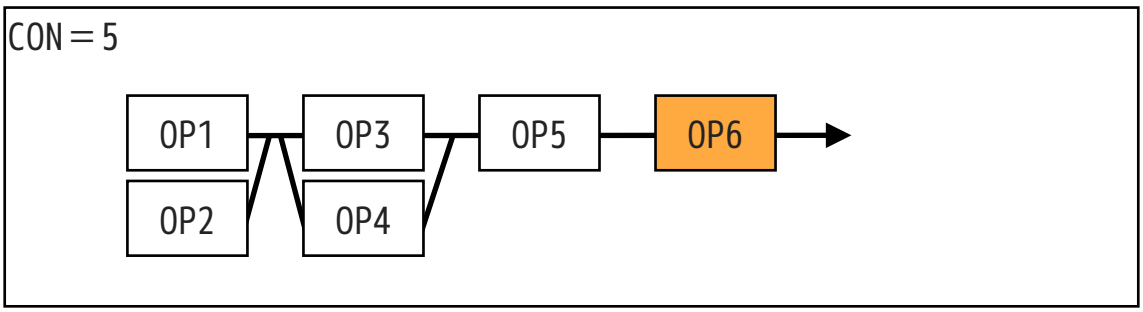


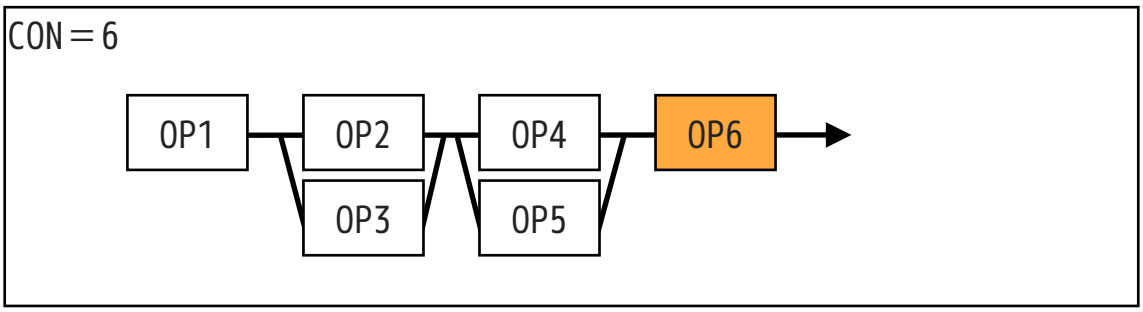


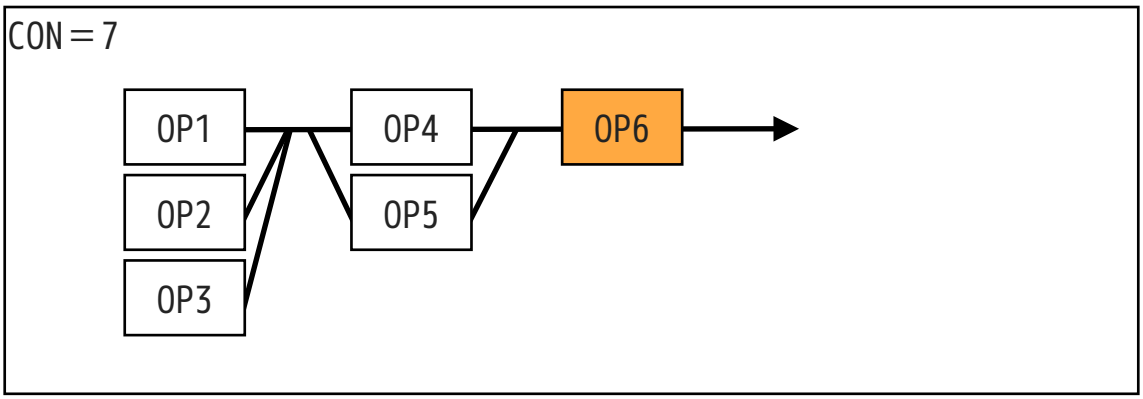


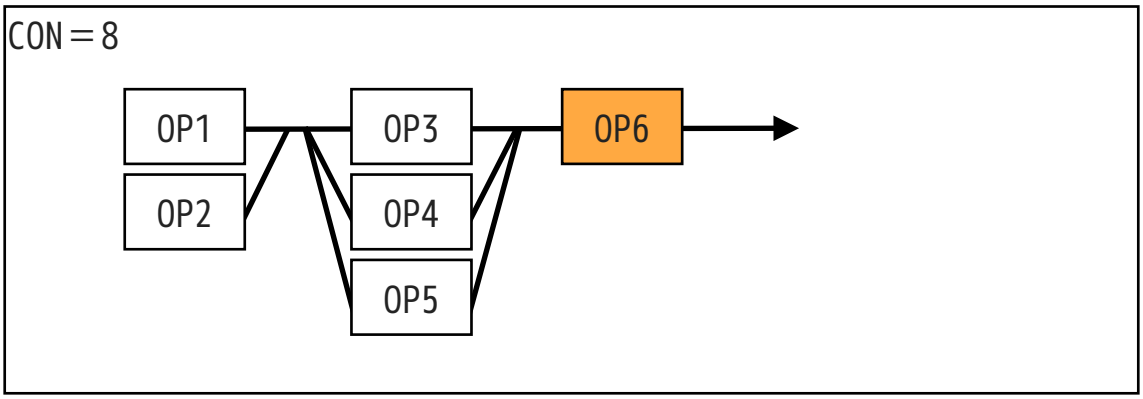

CON＝3
OP1
OP2
OP3
OP4
OP5
OP6

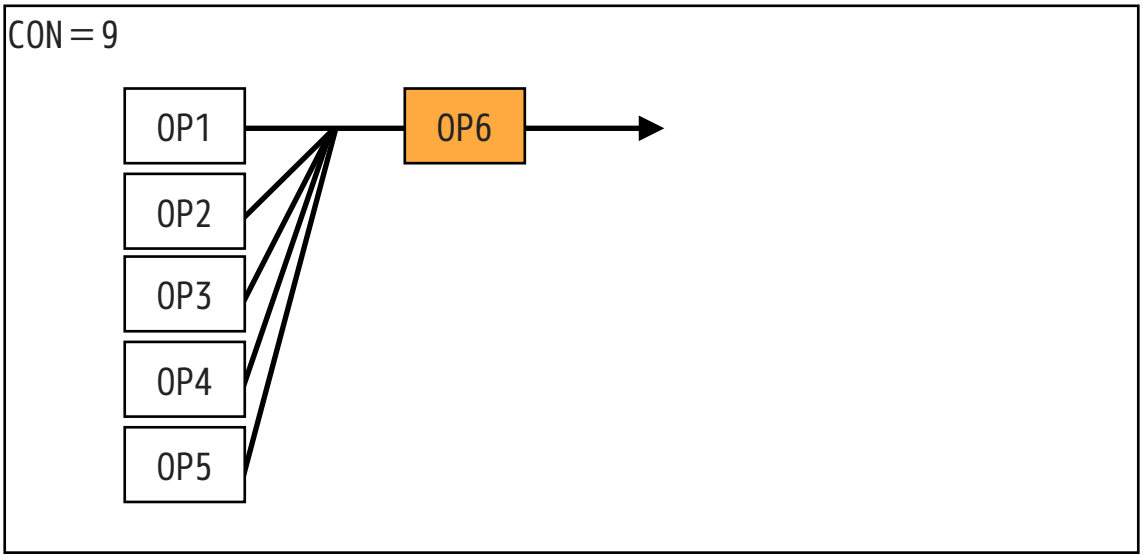

CON＝4
OP1
OP2
OP3
OP4
OP5
OP6


CON＝5
OP1
OP2
OP3
OP4
OP5
OP6


CON＝6
OP1
OP2
OP3
OP4
OP5
OP6

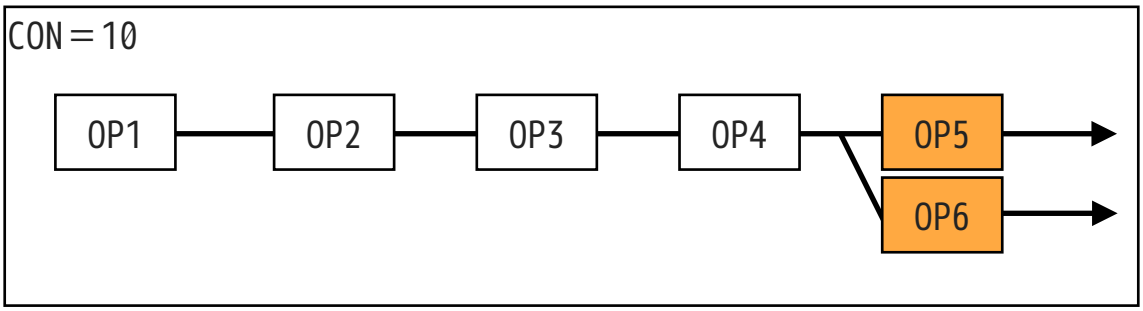

CON＝7
OP1
OP2
OP3
OP4
OP5
OP6


CON＝8
OP1
OP2
OP3
OP4
OP5
OP6

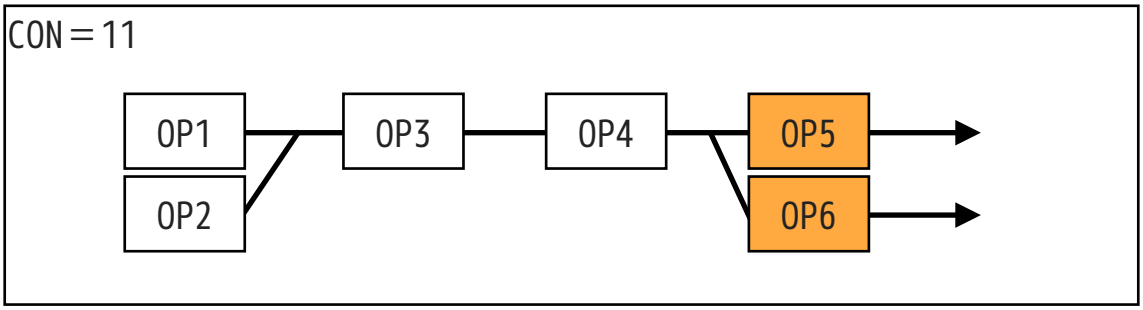

CON＝9
OP1
OP2
OP3
OP4
OP5
OP6

グループ２：CON=10～21：[1,2,3,4,5][6]

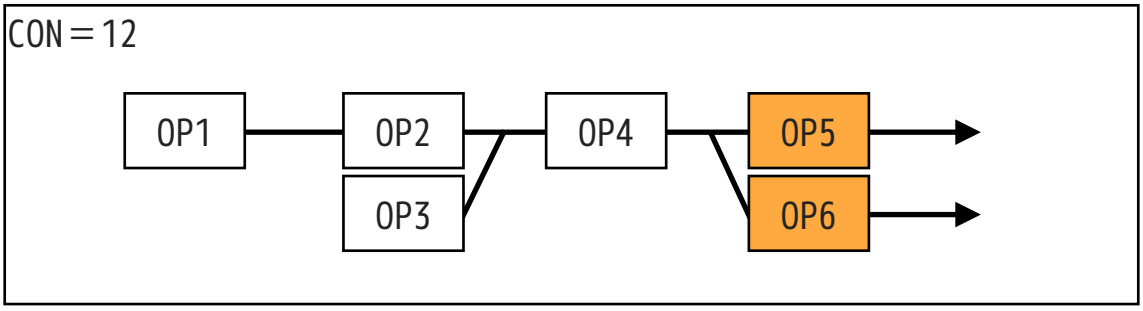
CON＝10



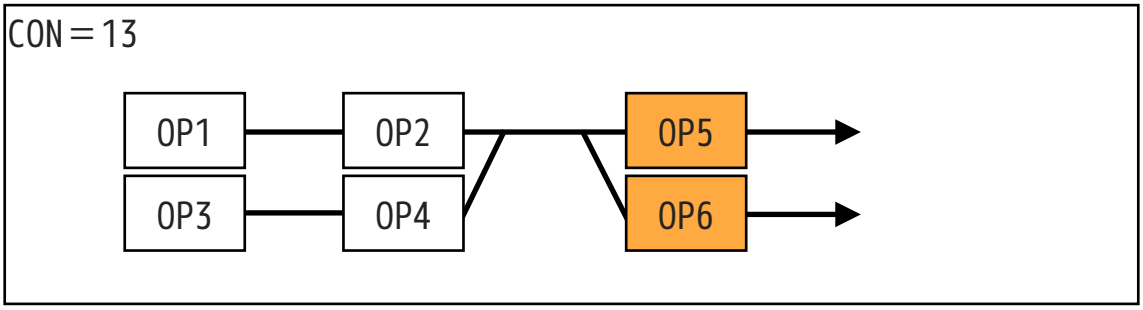
CON＝11



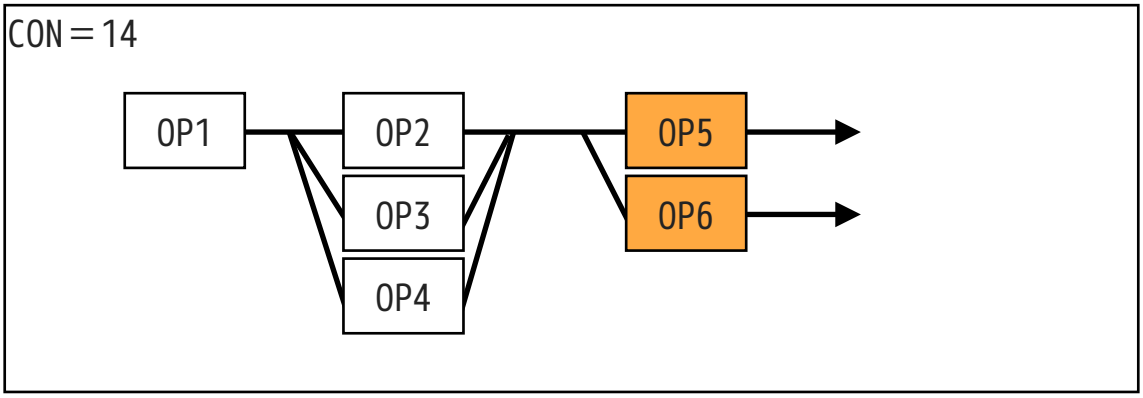
CON＝12



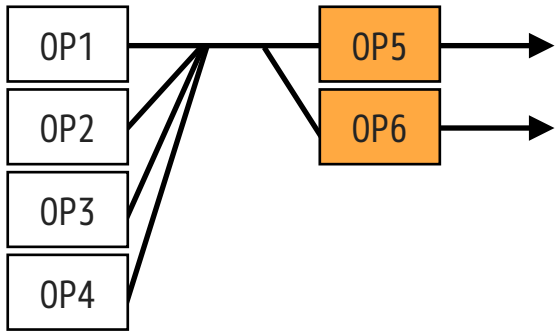
CON＝13



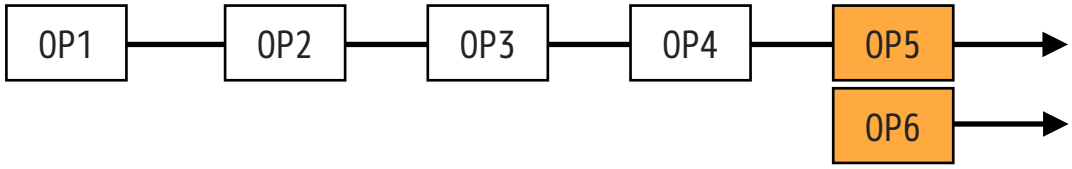
CON＝14



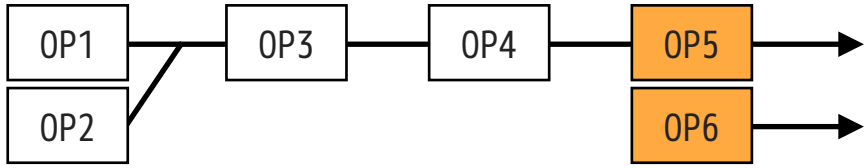
CON＝15



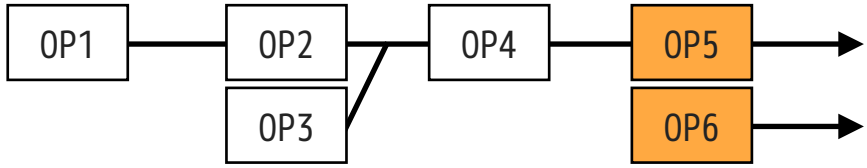
CON＝16



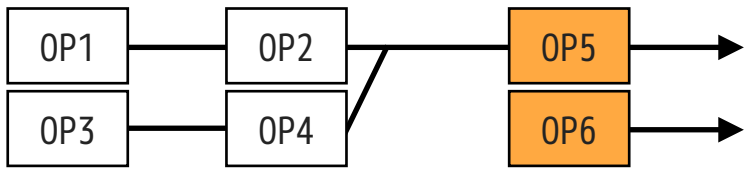
CON＝17



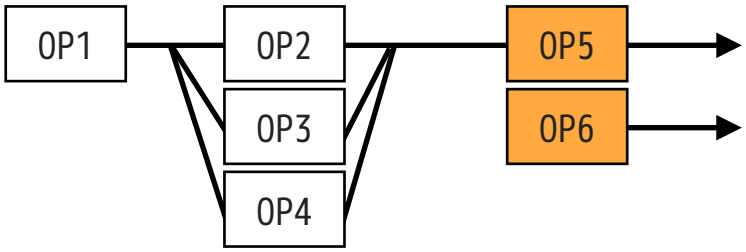
CON＝18



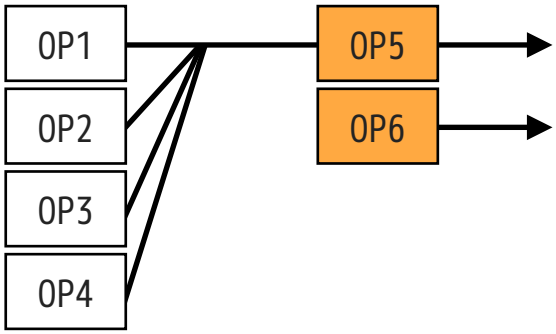
CON＝19



CON＝20

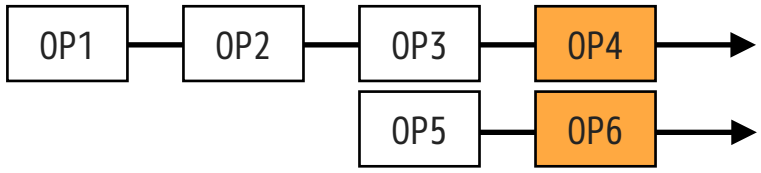


CON＝21

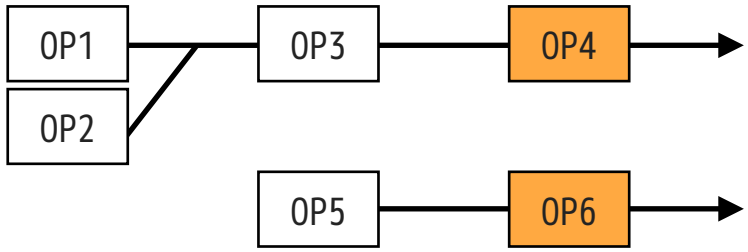


グループ３：CON=22～25：[1,2,3,4][5,6]

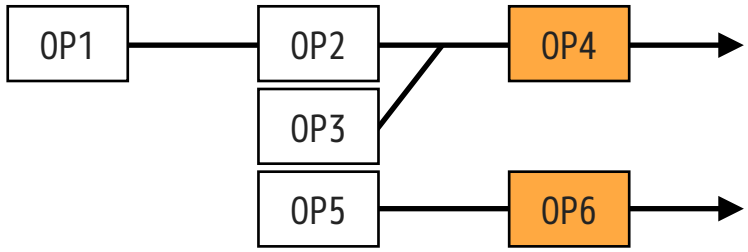
CON＝22



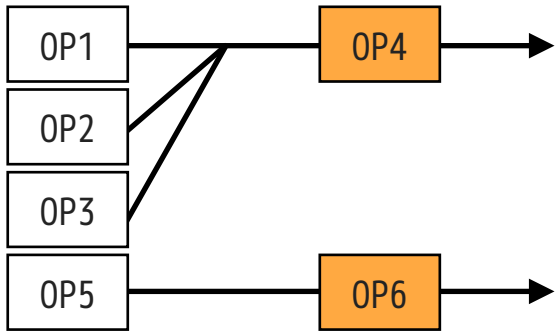
CON＝23



CON＝24

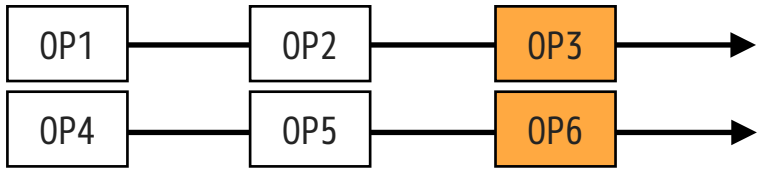


CON＝25

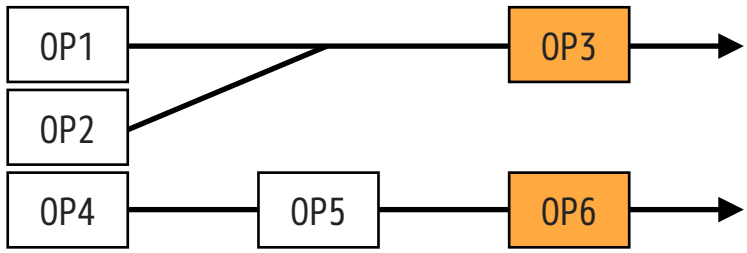


グループ４ ：CON=26～28：[1,2,3][4,5,6]

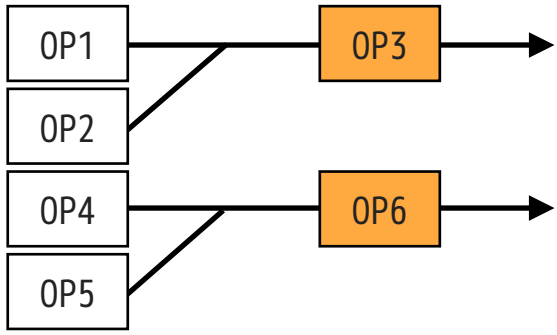
CON＝26



CON＝27

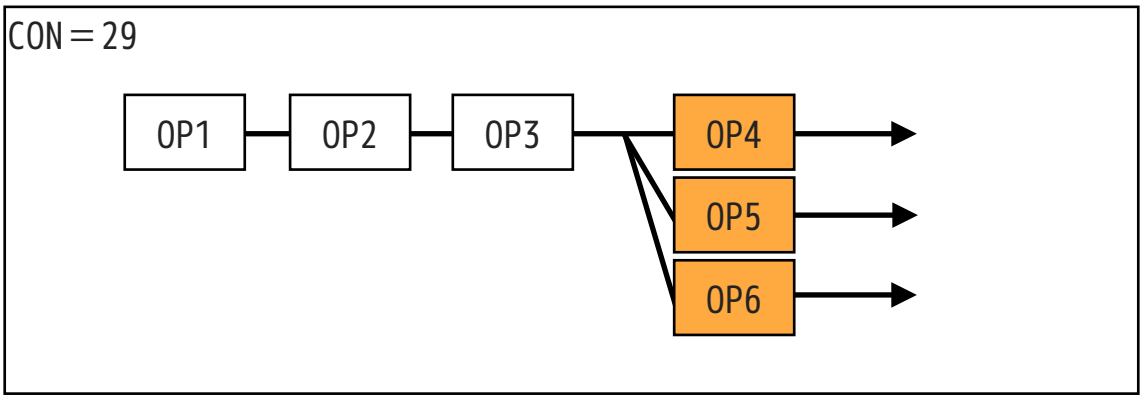


CON＝28

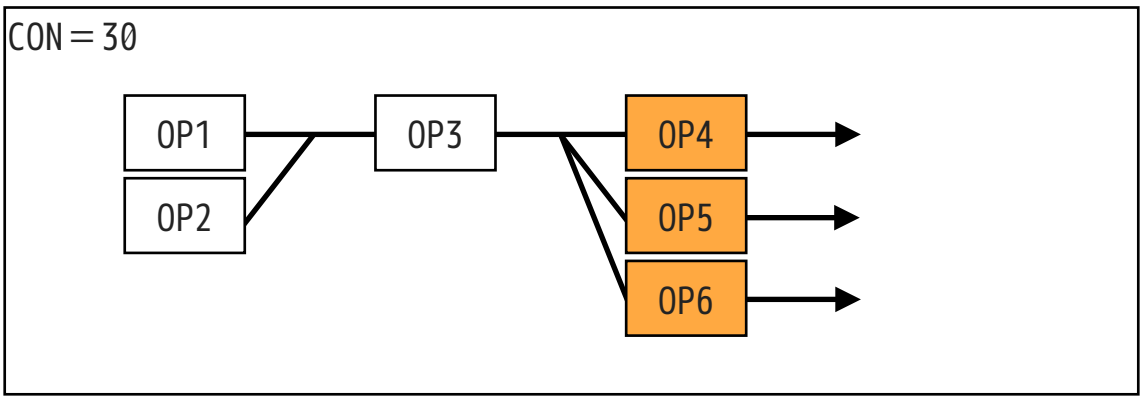


グループ５：CON=29～40：[1,2,3,4][5][6]

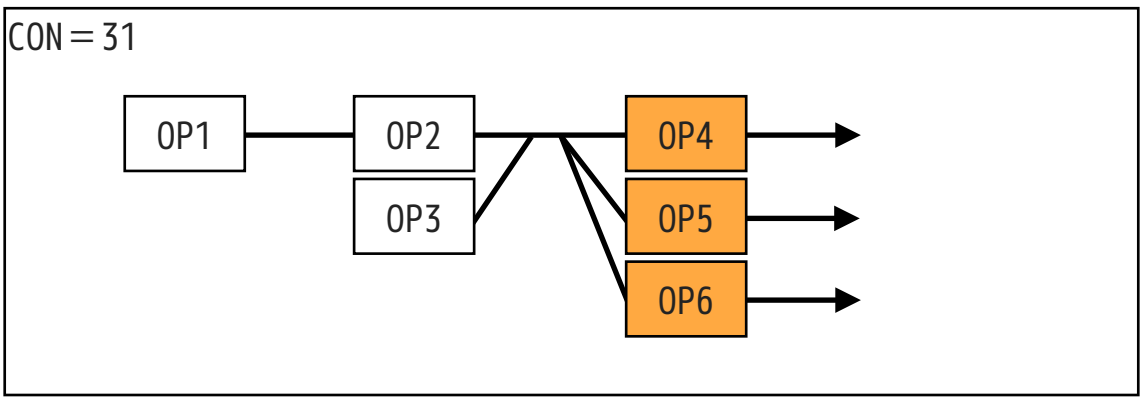
CON＝29



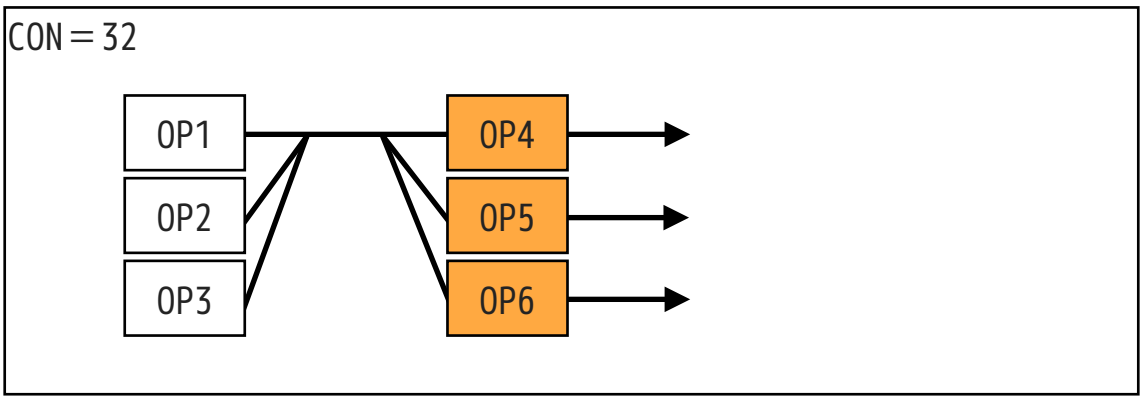
CON＝30



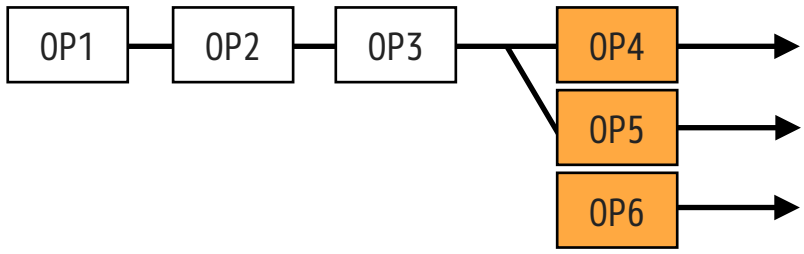
CON＝31



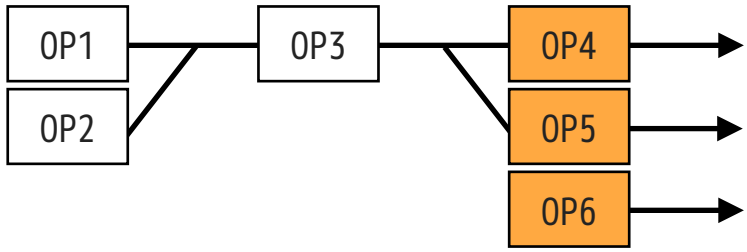
CON＝32



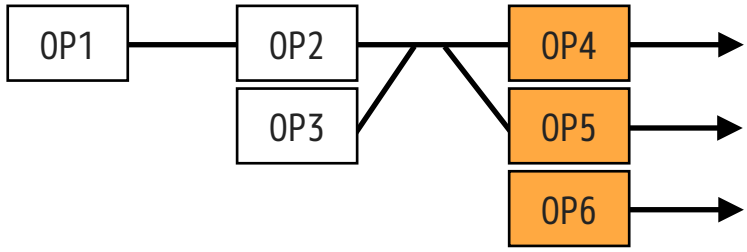
CON＝33



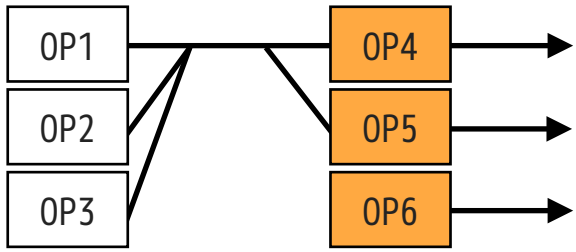
CON＝34



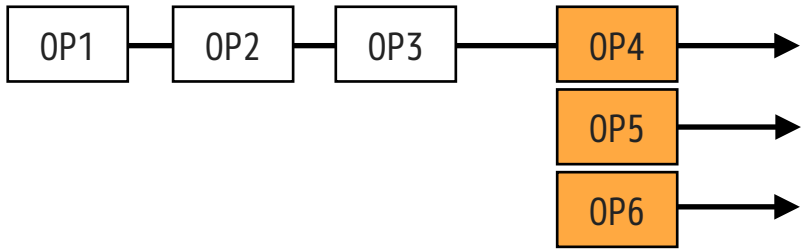
CON＝35



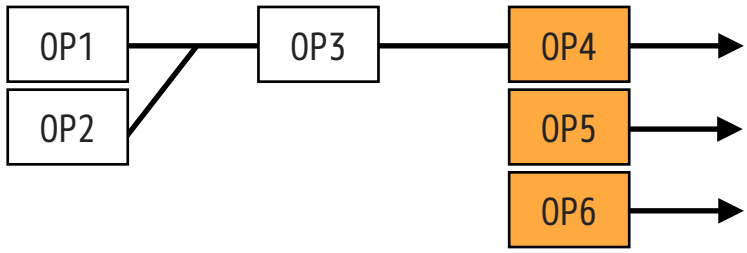
CON＝36



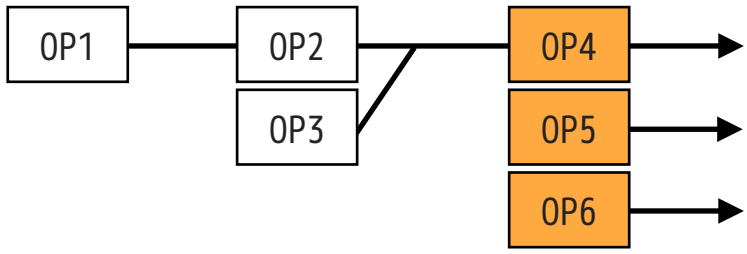
CON＝37



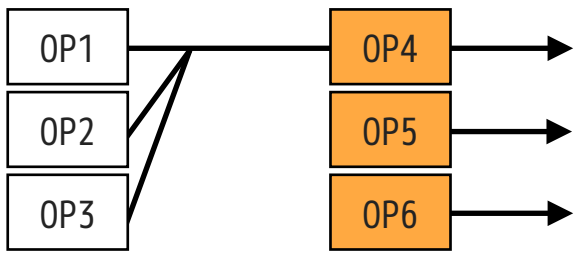
CON＝38



CON＝39

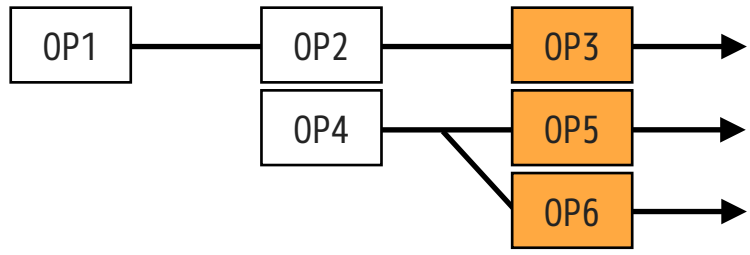


CON＝40

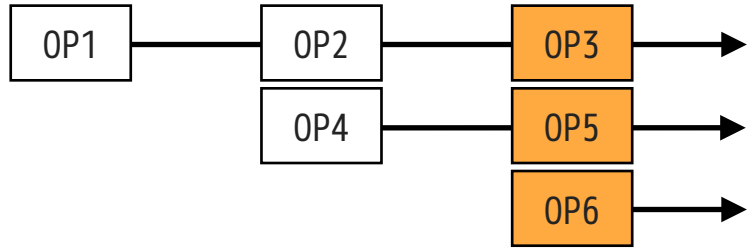


グループ６：CON=41～44：[1,2,3][4,5][6]

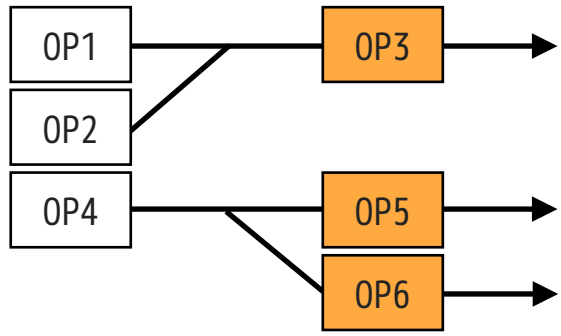
CON＝41



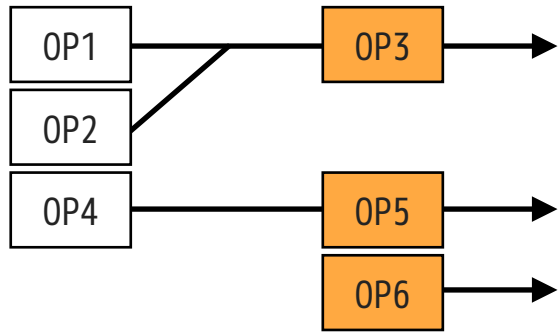
CON＝42



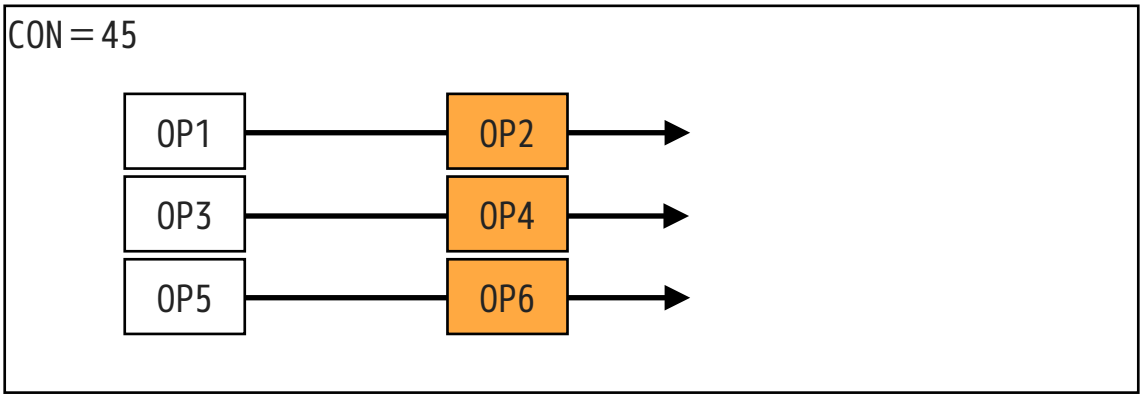
CON＝43



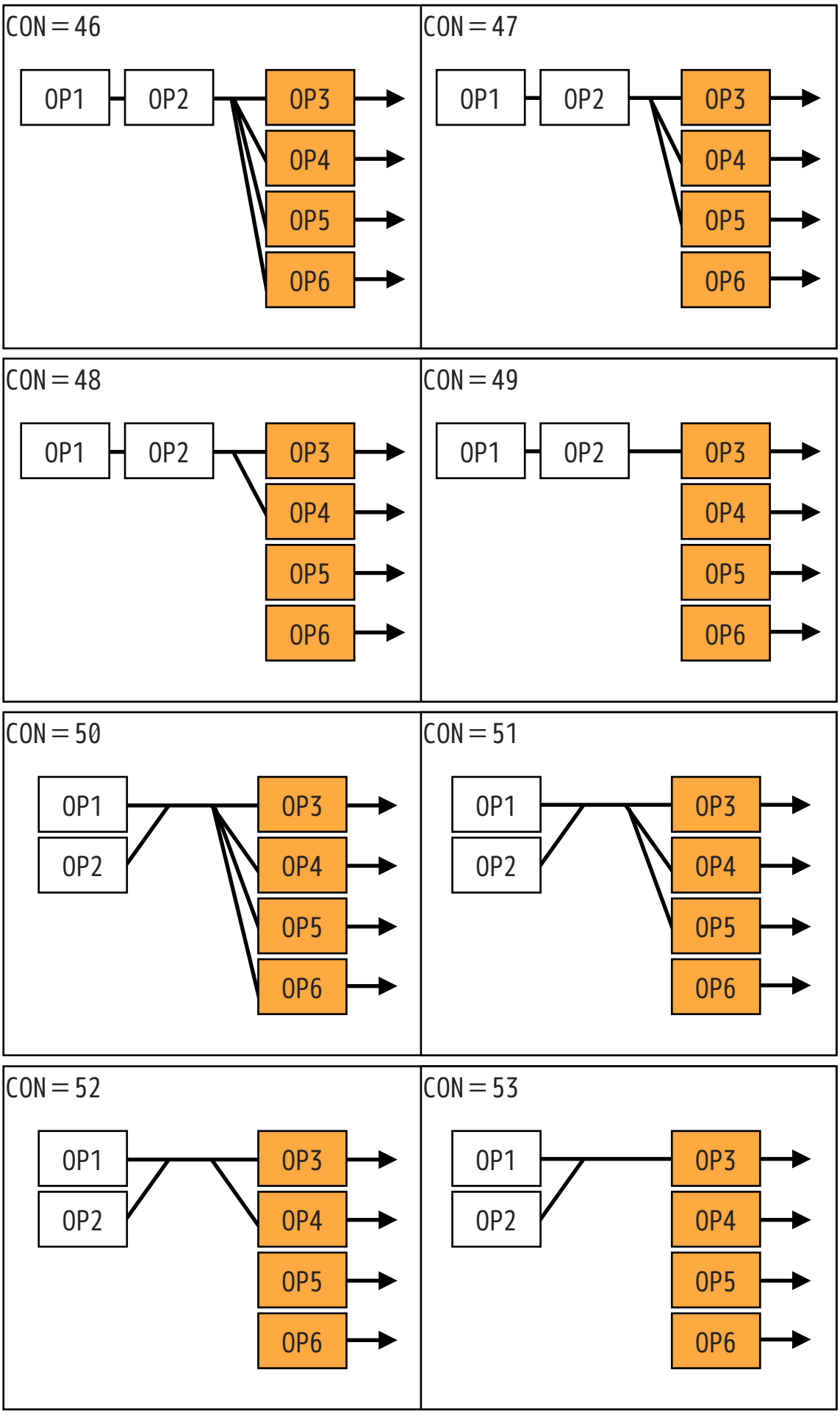
CON＝44



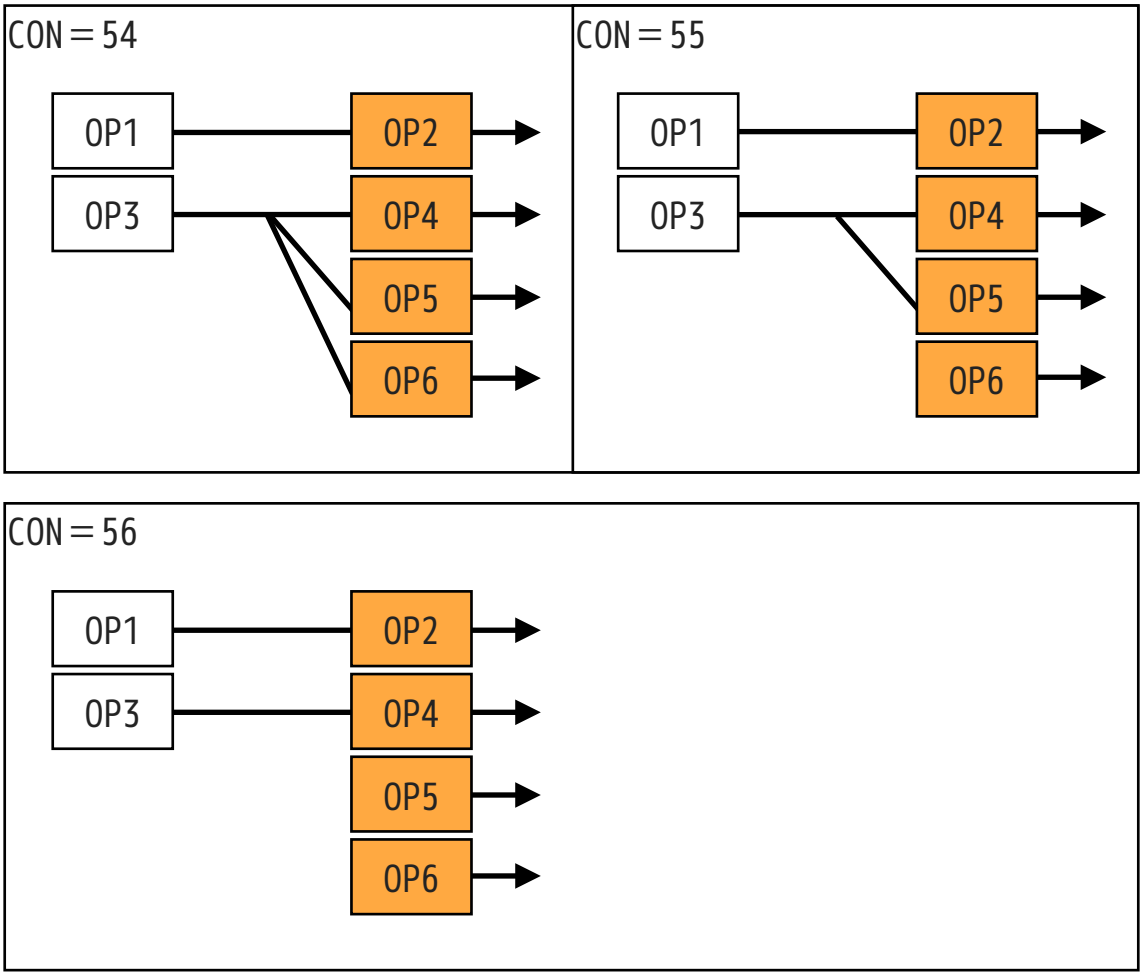
グループ７：CON=45：[1,2][3,4][5,6]



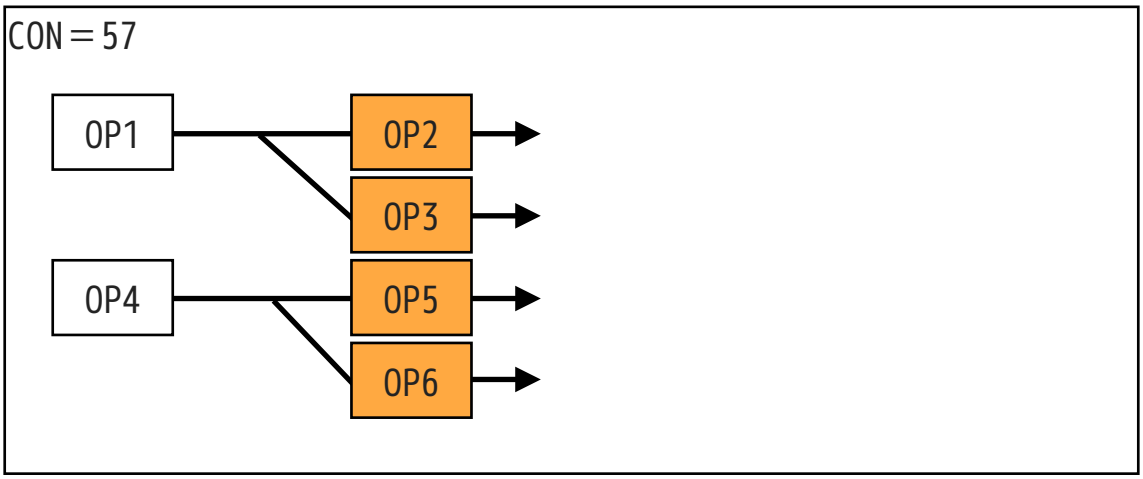
グループ８：CON=46～53：[1,2,3][4][5][6]



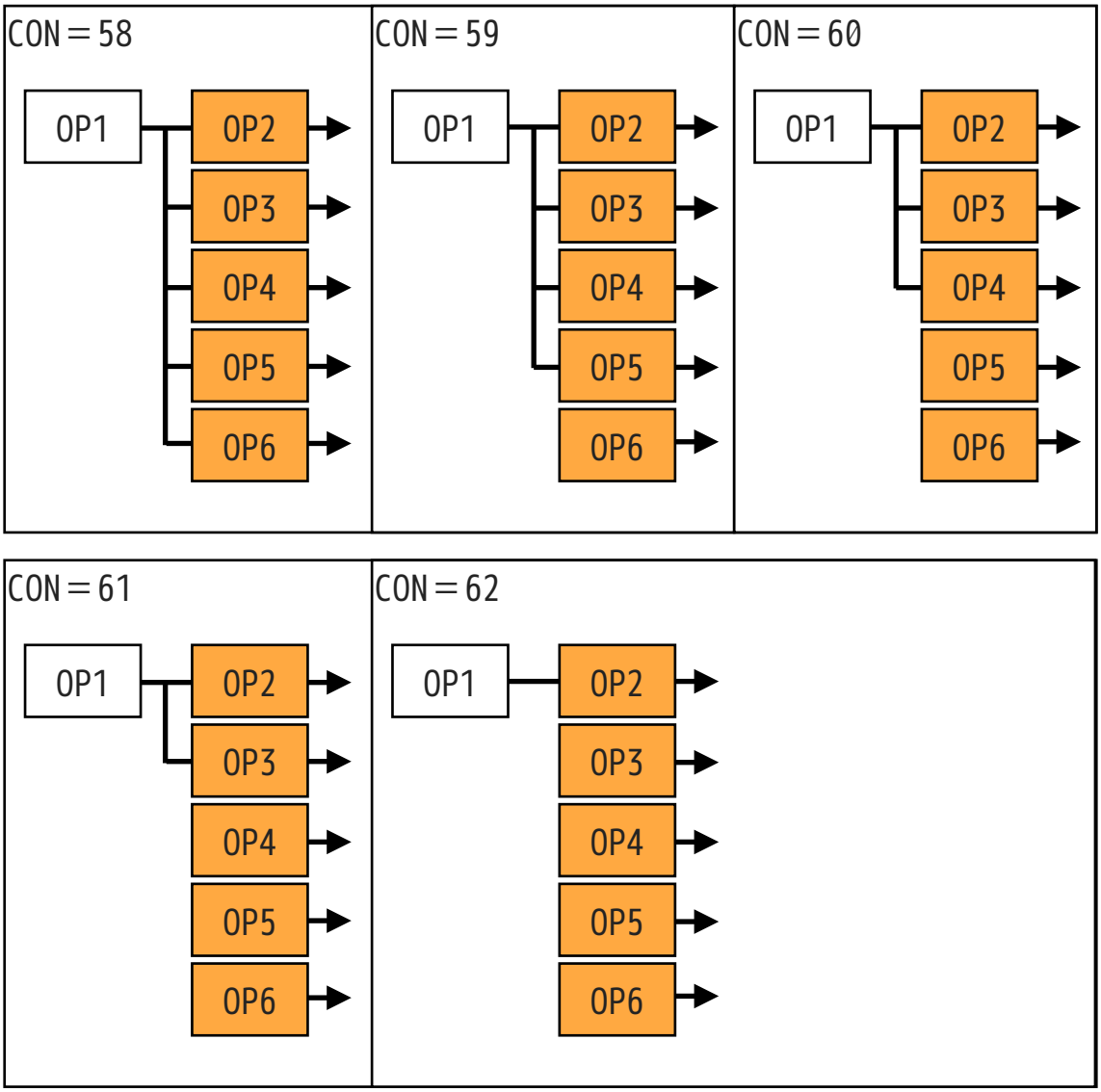
グループ９：CON=54～56：[1,2][3,4][5][6]



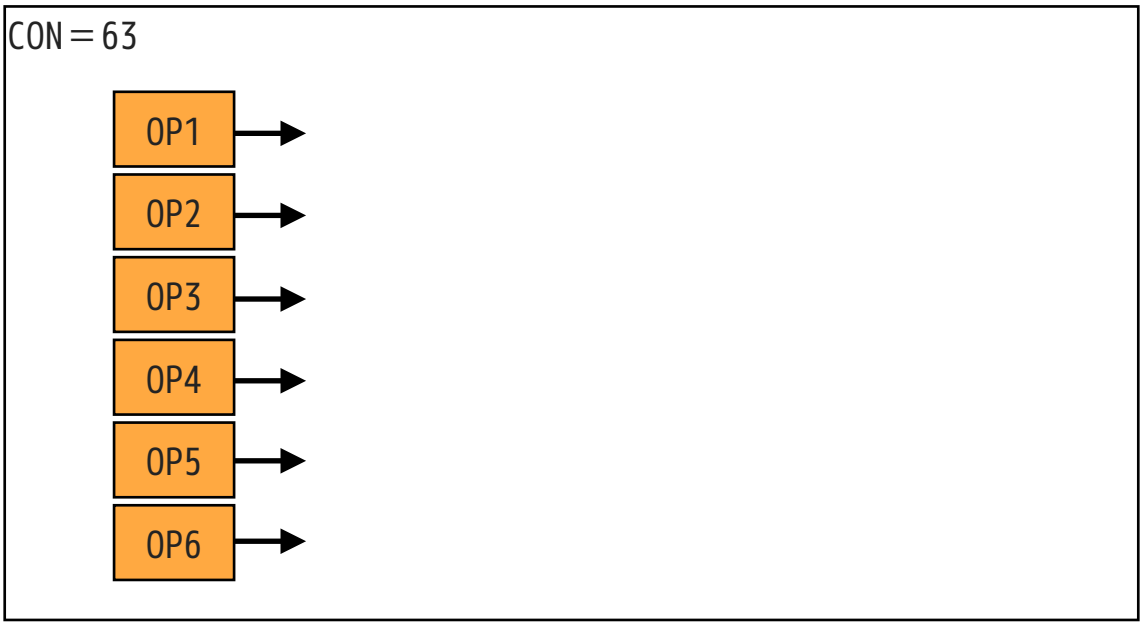
グループ１０：CON=57：[1,2][3][4,5][6]



グループ１１：CON=58～62：[1,2][3][4][5][6]



グループ１２：CON=63：[1][2][3][4][5][6]



6.56. ＦＭ音源４オペレータモードの接続形態

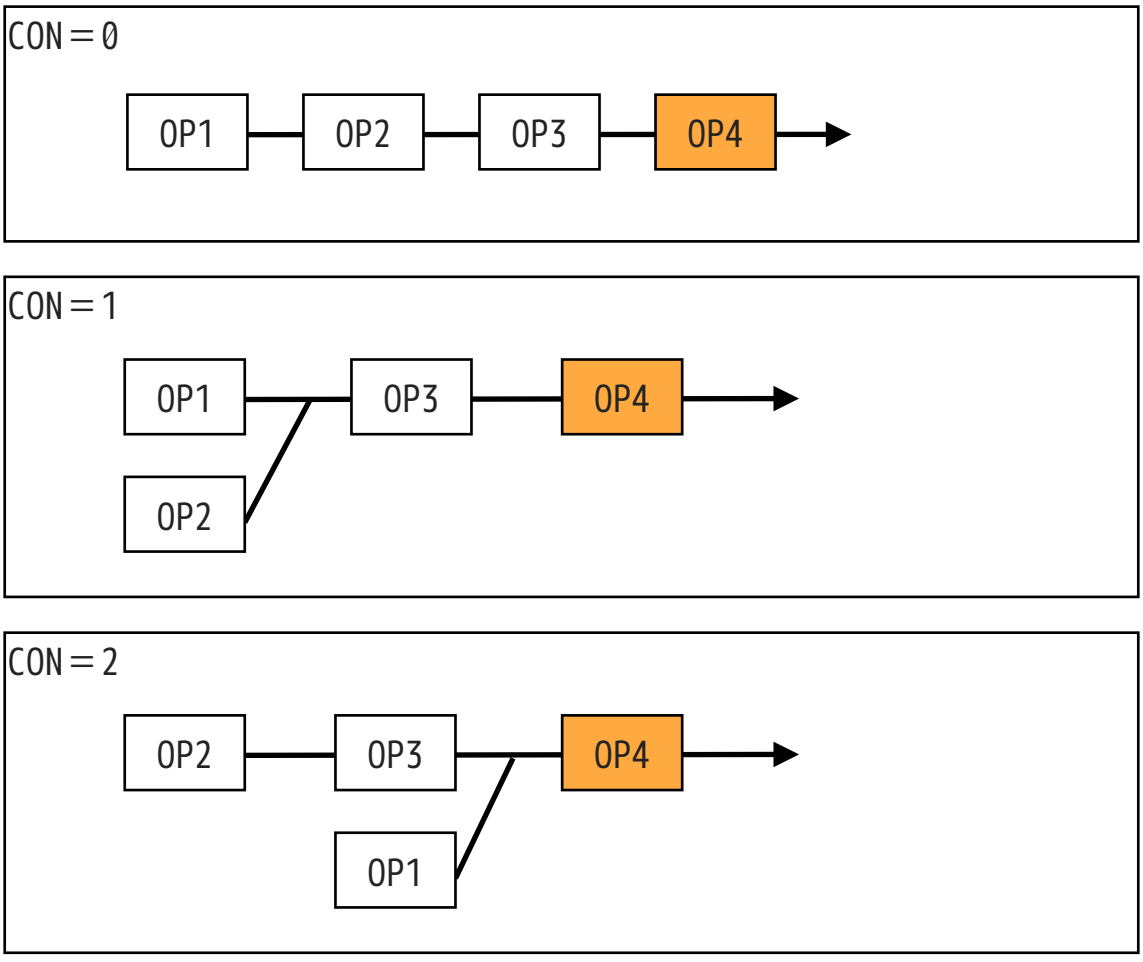
４オペレータモードの音色設定

```
#MB:FMS_4OP
#MB:FMS_OPM      （ＯＰＭ互換）
#MB:FMS_OPNA     （ＯＰＮＡ互換）
#MB:FMS_OPN      （ＯＰＮ互換）
```

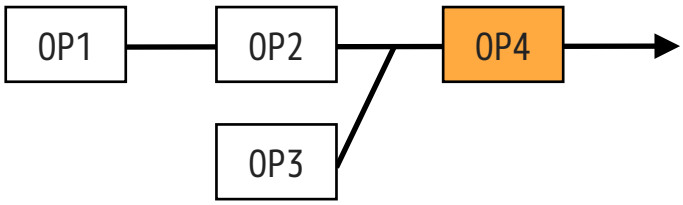
における、CON（connection：オペレータの接続パターン）の内容を示します。

色付きのオペレータがキャリアで、それ以外がモジュレータです。キャリアが複数あるCONでは、コーラスや和音の効果を出すことも可能です。
全てのオペレータにセルフフィードバック機能があります。

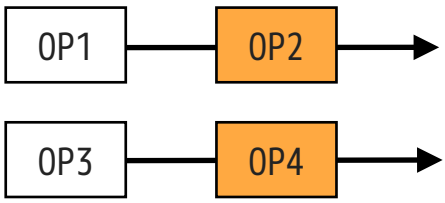
４オペレータモードの CON は、0～13 で指定します。
（通常0～7、拡張8～13）



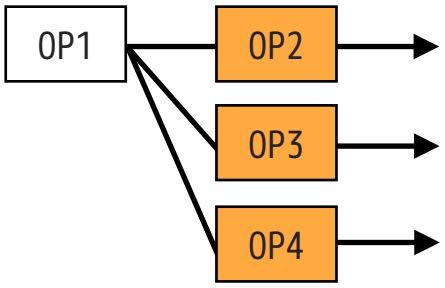
CON＝3



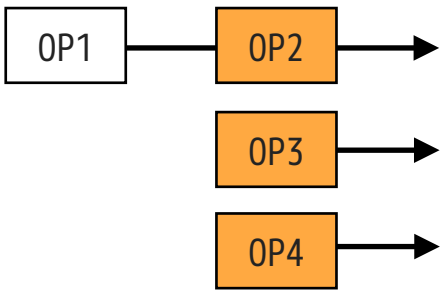
CON＝4

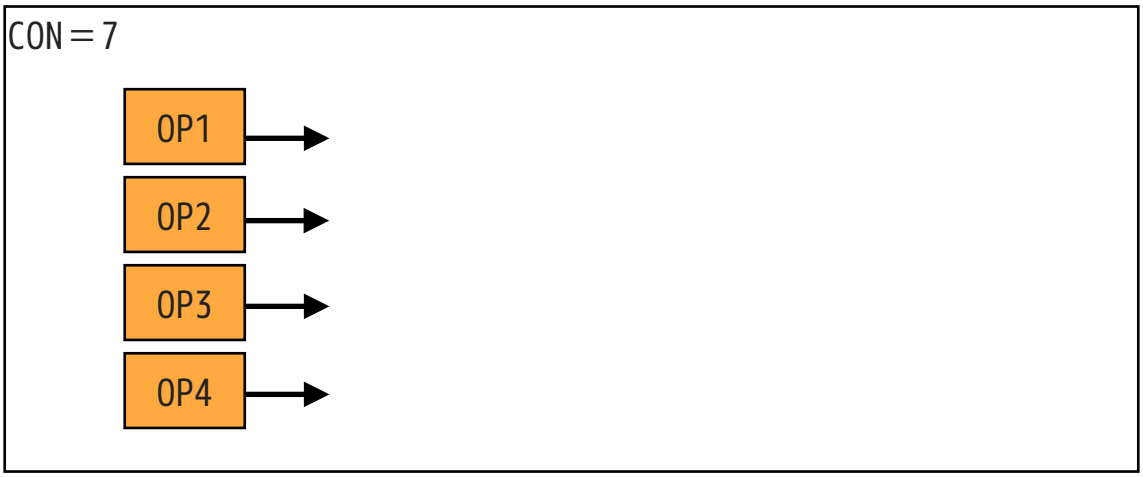


CON＝5

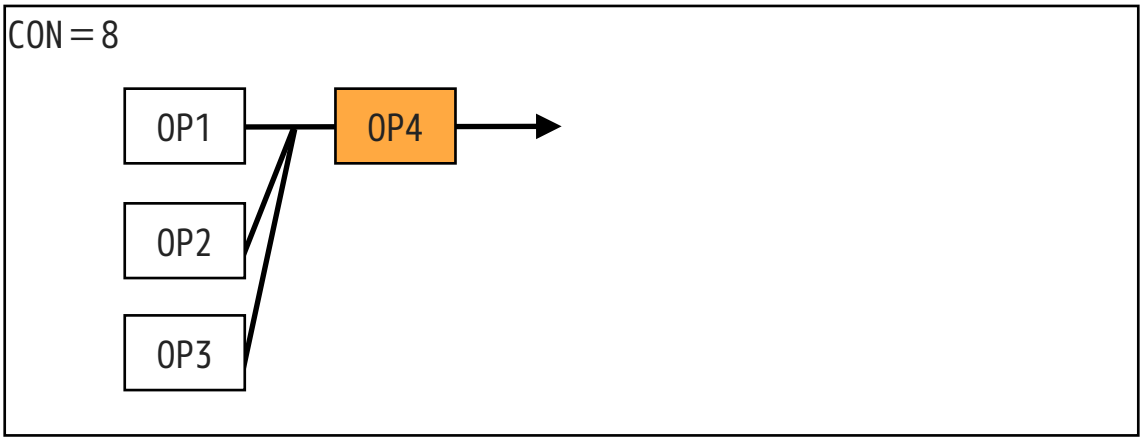


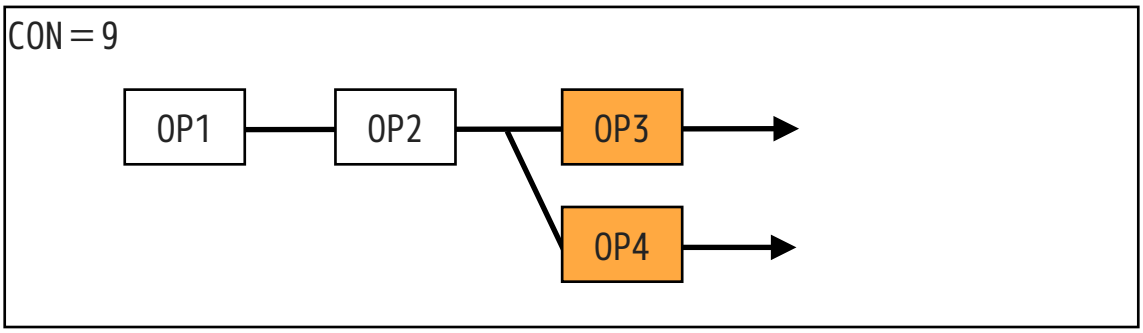
CON＝6

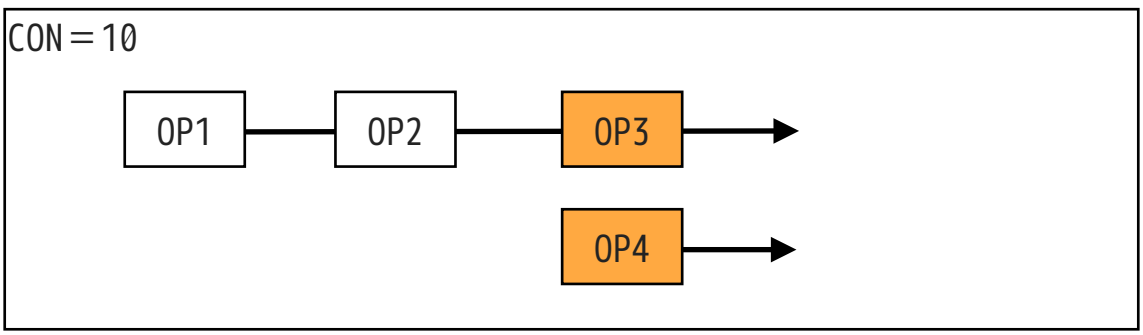




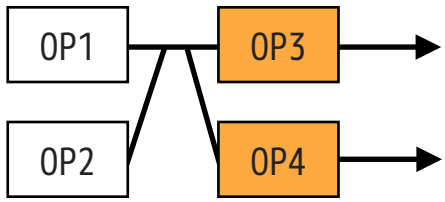
以下、独自拡張の接続形態（8～13）：



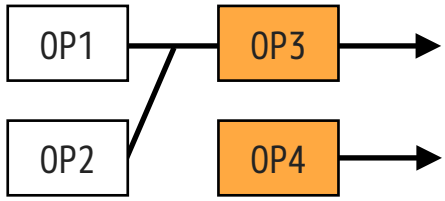




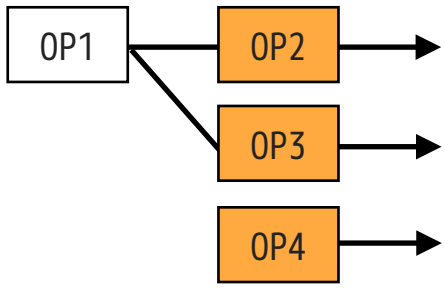
CON＝11



CON＝12



CON＝13



6.57. ＦＭ音源２オペレータモードの接続形態

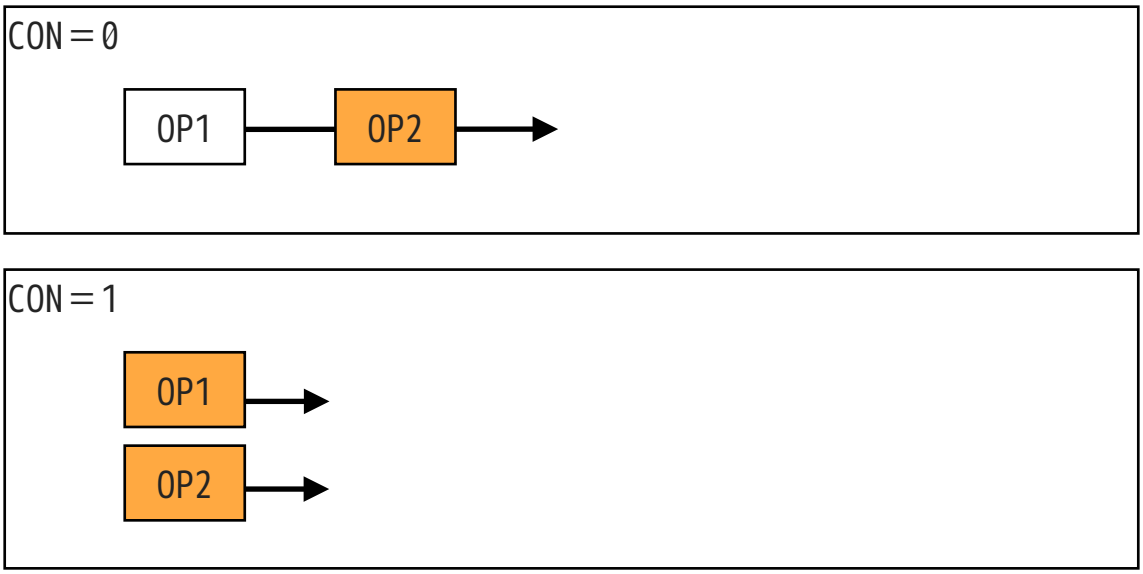
４オペレータモードの音色設定

　　#MB:FMS_2OP

における、CON（connection：オペレータの接続パターン）の内容を示します。

色付きのオペレータがキャリアで、それ以外がモジュレータです。キャリアが複数あるCONでは、コーラスや和音の効果を出すことも可能です。
全てのオペレータにセルフフィードバック機能があります。

２オペレータモードの CON は、0～1 で指定します。



6.58. #MB:LOAD_SAMPLING：サンプリングファイルの読み込み

【記述例】

```
#MB:LOAD_SAMPLING {
//load file
   v3mml__testpcm.v3b
|
//pick pcm
   smp=0: init_lv=64,
   smp=1: init_lv=40,
   smp=2,
   smp=3,
   smp=4,
   smp=5,
   smp=6,
   smp=7,
   smp=8,
|
//assign
   target=pcm: @=0: start=0: end=-1: loop=-1: key=48: fs=3579545/144: smp=0,
   target=pcm: @=1: start=0: end=-1: loop=-1: key=48: fs=3579545/144: smp=1,
   target=pcm: @=2: start=0: end=-1: loop=-1: key=48: fs=15625: smp=2,
   target=pcm: @=3: start=0: end=-1: loop=-1: key=48: fs=15625: smp=3,
   target=pcm: @=4: start=0: end=-1: loop=-1: key=48: fs=15625/2: smp=4,
   target=pcm: @=5: start=0: end=-1: loop=-1: key=48: fs=16075.6510416667: smp=5,
   target=pcm: @=6: start=0: end=-1: loop=-1: key=48: fs=6000:  smp=6,
   target=pcm: @=7: start=0: end=-1: loop=-1: key=48: fs=48000: smp=7,
   target=pcm: @=8: start=0: end=-1: loop=-1: key=48: fs=48000: smp=8,
}
```

【解説】

サンプリングデータのバイナリファイルを読み込んで、使用できるように、準備の設定を行います。（ロードサンプリング）

ロードサンプリングの定義方法の概要は、
・行頭からの記述で、キーワード「#MB:LOAD_SAMPLING」を書く。
・続けて、スペースを置き、中括弧「{}」で括られた設定データを書く。

です。

設定データの概要ですが、中身は各種記号で分割されて定義されます。
以後、便宜上、
「|」記号で分割された内容を、「チャンク」と呼びます。
さらに、チャンクの内容は各種記号で分割されていて、
「,」記号で分割された内容を、「エレメント」と呼びます。
「:」記号で分割された内容を、「パラメータ」と呼びます。
「=」記号で分割された内容を、「ペア」と呼びます。

設定データは、まず３つのチャンクに分割されます。

第１チャンクには、読み込むバイナリファイル名を記述します。

第２チャンクには、バイナリファイルに含まれるサンプリングデータの何番目のものを展開して読み込むか（抽出）を列挙します。

第３チャンクには、どの音源モジュールに対し、どういう設定で、第２チャンクで抽出したサンプリングデータを割り当てるかを列挙します。

・第１チャンク詳細

読み込むバイナリファイル名を、１つだけ記述します。

【注意】
・「|」記号を含むファイル名は使えません。
・先頭および末尾にスペースの付くファイル名は使えません。
・ファイル名の拡張子に制限は設けていませんが、標準は「.v3b」です。
・読み込み対象になる場所は、プレイリストのカレントフォルダ配下です。
　プレイリストのカレントフォルダは、
　PlayListウインドウの最下部に書かれています。
・読み込むファイルのフォーマットは、あらかじめ定められています。
　（フォーマットとは）

・第２チャンク詳細

バイナリファイルに含まれるサンプリングデータの何番目のものを展開して読み込むかを列挙します。

列挙する内容は、１個以上のエレメントになります。
エレメントの内容は、１個以上のパラメータになっています。
パラメータの内容は、１組のペアになっています。
ペアは、項目名=設定内容 という記述になっています。
以下、ペアの「項目名」ごとに説明します。

項目名「smp」

バイナリファイルに含まれるサンプリングデータの何番目のものを展開するかを定義します。
設定範囲は 0 ～（ファイル内での定義数-1）です。
上述の記述例を元に説明すると、第２チャンクの最初のエレメントは

```
smp=0: init_lv=64,
```

という定義になっています。
これは、バイナリファイル内の０番目を読んで抽出番号０番とし、
バイナリの内容を 16bit signed pcm に展開して割り当てる、
という意味になります。

バイナリファイル内の「何番目を指すか」と「抽出番号」は連動します。
例えば、

```
smp=7,
```

という定義は、記述例では順番に並んでいますが、どの位置に「smp=7,」が書かれていても、７番目を読んで抽出番号を７番とする、という意味になります。

展開すると表現する理由は、扱うサンプリングデータが pcm 以外に adpcm や dpcm などもあり、読み込み時に 16bit signed pcm に統一的に変換して利用するためです。

※「抽出番号」は、第３チャンクで参照されます。

項目名「init_lv」

展開対象になるサンプリングバイナリが、dpcmだった場合に指定する、dpcm差分計算の初期値になります。
設定範囲は 0 ～ 127 です。
変位が 0 となる中心値は 64 です。

・第３チャンク詳細

どの音源モジュールに対し、どういう設定で、第２チャンクで抽出したサンプリングデータを割り当てるかを列挙します。

列挙する内容は、１個以上のエレメントになります。
エレメントの内容は、１個以上のパラメータになっています。
パラメータの内容は、１組のペアになっています。
ペアは、項目名=設定内容 という記述になっています。
以下、ペアの「項目名」ごとに説明します。

項目名「target」

割り当てる対象の音源モジュール名を、キーワードで指定します。
現状、設定の選択肢は、

```
pcm
wvm
```

以上の２種類です。

target以外の必須になる項目名は、targetの内容に別々に決まっています。

「target=pcm」の場合：
　「target」以外の必須項目名：
　「@」「start」「end」「loop」「key」「fs」「smp」
　省略可能な項目名：
　「amp_mul」
「target=wvm」の場合：
　「target」以外の必須項目名：
　「@」「smp」
　省略可能な項目名：
　「amp_mul」

「target=pcm」の場合の項目名「@」

音源モジュール「@@"pcm"」におけるサブフォーム番号指定「@」の何番に割り当てるかを設定します。
設定範囲は 0 ～ 1023 です。

初期設定で波形番号 0番 にダミーの初期データが入っています（上書可）。
ＭＭＬで未定義の波形番号を参照した場合は、0番 が読み出されます。

「target=pcm」の場合の項目名「start」

参照するサンプリングデータの開始インデックス番号を設定します。
設定範囲は、0 ～（サンプル総数-1）です。
特例として、「-1」を記述すると、サンプリングデータの末尾の番号として扱われます。

「target=pcm」の場合の項目名「end」

参照するサンプリングデータの終了インデックス番号を設定します。
設定範囲は、0 ～（サンプル総数-1）です。
特例として、「-1」を記述すると、サンプリングデータの末尾の番号として扱われます。

「target=pcm」の場合の項目名「loop」

参照するサンプリングデータのループ開始インデックス番号を設定します。
設定範囲は、0 ～（サンプル総数-1）です。
特例として、「-1」を記述すると、ループ機能無効（サンプリングデータの末尾の番号）として扱われます。

「target=pcm」の場合の項目名「key」

「fs=[]」のサンプリング周波数を、どの音程番号に割り当てるかを指定します。
（音程番号とは）
設定範囲は 0 ～ 119 です。
例えば、

```
fs=8000, key=48,
```

の場合、o4c のＭＭＬ記述でサンプリング周波数 8kHz 相当の再生になります。

「target=pcm」の場合の項目名「fs」

参照するサンプリングデータのサンプリング周波数を指定します。
設定範囲は 2000.0 ～ 384000.0 です。

「target=pcm」の場合の項目名「amp_mul」

参照するサンプリングデータの振幅への倍率を指定します。
この項目を省略した場合は、1.0（倍率変更なし）が指定されたものとみなされます。
負数を指定した場合は、振幅が上下反転することになります。

「target=pcm」の場合の項目名「smp」

参照するサンプリングデータを、第２チャンクで決めた、どの抽出番号のものにするかを指定します。
設定範囲は 0 ～（ファイル内での定義数-1）です。

「target=wvm」の場合の項目名「@」

音源モジュール「@@"wvm"」におけるサブフォーム番号指定「@」の何番に割り当てるかを設定します。
指定可能範囲は 0 ～ 1023 です。

初期設定で波形番号 0番 にダミーの初期データが入っています（上書可）。
ＭＭＬで未定義の波形番号を参照した場合は、0番 が読み出されます。

「target=wvm」の場合の項目名「smp」

参照するサンプリングデータを、第２チャンクで決めたどの抽出番号のものにするかを指定します。

「target=wvm」の場合の項目名「amp_mul」

参照するサンプリングデータの振幅への倍率を指定します。
この項目を省略した場合は、1.0（倍率変更なし）が指定されたものとみなされます。
負数を指定した場合は、振幅が上下反転することになります。

6.59. 読み込むバイナリファイルのフォーマット

メタデータ定義のために一時バイナリ領域に読み込むファイルのフォーマットは、大きく分けて３つのチャンクから成り立っています。

第１チャンク：ヘッダー
第２チャンク：アロケーションテーブル
第３チャンク：実データ群

また、バイナリデータファイル全体のサイズは２ギガバイトまでです。上限を超えるサイズの場合、演奏開始で読み込む際にエラーとなります。

第１チャンク詳細（原則32バイト）

(1)文字列16バイト（終端なし固定サイズ）：
　　当ファイルの認識用ヘッダー。次のように定義。
　　V3MML...20220622
　　ヘッダー確認は現状先頭10文字で行われます。
　　末尾８文字（20220622）はデータフォーマット番号で、
　　更新される可能性があります。
(2)整数４バイト(big endian)：
　　ファイル先頭から第２チャンク先頭までのオフセット。
　　原則 $00000020 です。
(3)整数４バイト(big endian)：
　　ファイル先頭から第３チャンク先頭までのオフセット。
　　第２チャンクでの定義数に応じて変化します。
(4)予約８バイト：
　　拡張用の予約領域。

第２チャンク詳細（サイズ可変）

データアロケーション１個あたり１６バイト定義。
アロケーション個数は(2)と(3)から算出。

データアロケーション１個あたりの詳細
(5)整数４バイト(big endian)：
　　当アロケーションの開始位置。(3)からのオフセット。
(6)整数４バイト(big endian)：
　　当アロケーションのバイト数。
　　バイト数が０の場合は未定義とみなします。
　　（サンプル数ではなく、バイト数であることに注意）

(7)予約４バイト：
　　拡張用の予約領域。
(8)文字列４バイト（終端なし固定サイズ）：
　　当アロケーションの種別を表す文字列。
　　これはバインドされるファイルの拡張子で、
　　ドットで始まる４文字です。（例「.16l」）

(8)における現在のサポート拡張子（８種類）：
.16l（いち・ろく・える）
signed 16bit PCM、リトルエンディアン、モノラルデータです。

.16b（いち・ろく・びー）
signed 16bit PCM、ビッグエンディアン、モノラルデータです。

.u08（ゆー・ぜろ・はち）
unsigned 8bit PCM、モノラルデータです。

.s08（えす・ぜろ・はち）
signed 8bit PCM、モノラルデータです。

.ayb（えー・わい・びー）
YM2608互換 4bit ADPCM 、ニブルの順番は上位から下位、モノラルデータです。

.amb（えー・えむ・びー）
OKI互換 4bit ADPCM 、ニブルの順番は上位から下位、モノラルデータです。

.aml（えー・えむ・える）
OKI互換 4bit ADPCM 、ニブルの順番は下位から上位、モノラルデータです。

.dpc（でぃー・ぴー・しー）
FC互換 DPCM 、上位ビットから下位ビット方向への１ビットずつの８ビット単位、モノラルデータです。

第３チャンク詳細（サイズ可変）

バイナリファイルの内容そのものが貼り付けられます。
各ファイル内容の開始位置は(5)、サイズ（バイト数）は(6)の通りです。

7.エンベロープ関連

7.1. @eoc[str],[1]：エンベロープオプション CLOCK

7.2. @eor[str],[1]：エンベロープオプション RESOLUTION

【記述例】

```
#MB:ENV_A @ea=1 { peak=15, init=8, |
   n:2:15, n:12:10, n:30:10, n:10:12, l:10:8, n:10:12,
   r:i1:6, n:40:6, n:1:0
}
t100 q11,16 v15 @@"pls" @ea1
@eoc"sec",1/120
@eor"smp"
l2 o5 cegefdgg cegefdg&g;
```

【解説】

エンベロープ（@ea,@ef）への、オプション設定を行います。
@eoc[str],[1] および @eor[str],[1] は、

```
#MB:CONFIG {
   env_clock: unit=sec: rate=1/60,
   env_resol: unit=sec: rate=1/300,
}
```

による設定をトラックごとに変更したい場合に使用します。

@eoc[str],[1]

エンベロープクロックの設定が対象になります。
エンベロープクロックは、エンベロープの１節での経過時間の単位です。

指定する引数は次の通りです。
引数[str]　：時間単位
引数[1]　　：時間単位への倍率
引数はカンマで区切って指定します。

【@eocの設定内容】

引数[str]	引数[str]の設定内容と、引数[1]の設定範囲
"sec"	時間単位が「１秒」になります。 引数[1]の設定範囲は、1/2400[秒] ～ 1.0[秒] です。
"tick"	時間単位は「１tickカウント」になります。 引数[1]の設定範囲は、０より大きい値です。 ただし、最小は1/2400[秒]に制限されます。

@eor[str],[1]

エンベロープ解像度の設定が対象になります。
エンベロープ解像度は、振幅計算更新の最少時間の単位です。

指定する引数は次の通りです。
引数[str]　：解像度の時間単位
引数[1]　　：解像度の時間単位への倍率
引数はカンマで区切って指定します。

【@eorの設定内容】

引数[str]	引数[str]の設定内容と、引数[1]の設定範囲
"smp"	解像度の時間単位が「１サンプル（最大解像度）」になります。 引数[1]は不要です。（引数[1]は指定できません）
"sec"	解像度の時間単位が「１秒」になります。 引数[1]の設定範囲は、1/2400[秒] ～ 1.0[秒] です。
"tick"	解像度の時間単位は「１tickカウント」になります。 引数[1]の設定範囲は、０より大きい値です。 ただし、最小は1/2400[秒]に制限されます。

【備考】
時間単位が「１tickカウント」の場合、テンポにより時間単位が変動します。

エンベロープ指定後にテンポを変更した場合、テンポ変更後のノートオンからエンベロープの時間計算が自動追従します。
つまり、ノートオン中にテンポ変更が掛かった場合は、そのノートオン中は時間計算の自動追従はできませんが、次のノートオンから自動追従します。

7.3. @ea[1]：音量エンベロープ

7.4. #MB:ENV_A：音量エンベロープ定義

【記述例】

```
#MB:ENV_A @ea=1 {
   peak=15, init=8, |
   n:2:15, n:12:10, n:30:10, n:10:12, l:10:8, n:10:12,
   r:i1:6, n:40:6, n:1:0
}
t100 q11,16 v15 @@"pls" @ea1
l2 o5 cegefdgg cegefdg&g;
```

このように書くと、

- #MB:ENV_A によって、定義番号１番に音量エンベロープ内容を割り当てています。
- @ea1によって、定義番号１番の #MB:ENV_A の内容を適用。

といった定義と適用を記述できます。

【備考】記述例のエンベロープ定義の内容

音量８で開始、
時間２で最大（１５）、時間１２でレベル１０、
時間３０でレベル１０維持、時間１０でレベル１２、
｛繰り返し開始｝時間１０でレベル８、時間１０でレベル１２、
繰り返し開始にジャンプ。
以上の流れのどこでノートオフになるかは音長次第ですが、
ノートオフになったら、
レート１でレベル６、時間４０でレベル６維持、時間１でレベル０になる。

【解説】

音量エンベロープコマンド（@ea[1]）：
音量エンベロープ定義（#MB:ENV_A）の定義番号を整数で指定します。
定義番号は 0 ～ 1023 の整数です。
未定義の番号を指定するとエラーになります。

トラック先頭における初期設定には定義番号は無く、システム側で用意された暫定的な音量エンベロープが掛かっています。
【注】@@"fms"（ＦＭ音源）には、@EAコマンド は無効です。
　　@@"fms"では音色データ定義内のエンベロープを使用します。

音量エンベロープ定義（#MB:ENV_A）：
音量エンベロープコマンド（@ea[1]）で使用する、音量エンベロープ設定を定義します。

音量エンベロープ定義方法の概要は、
・行頭からの記述で、キーワード「#MB:ENV_A」を書く。
・続けて、スペースを置き、「@ea=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られた設定データを書く。
です。

定義番号：

定義番号は、@eaコマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 1023 です。

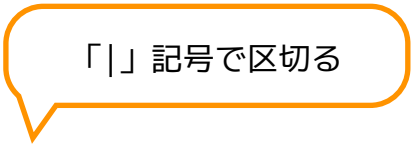
設定データ：

【記述例】

```
#MB:ENV_A @ea=1 {
   peak=15, offset=0, init=8, |
   n:2:15, n:12:10, n:30:10, n:10:12, l:10:8, n:10:12,
   r:i1:6, n:40:6, n:1:0
}
```

【解説】

フォーマットは次の通りです。




```
#MB:ENV_A @ea=定義番号 {
     peak=[], offset=[], init=[], |
     [ENV節],[ENV節],...（少なくとも２個以上のENV節）
}
```

peak：
peak=[]では、エンベロープの出力レベルの最大値を、0 より大きい値で指定します。
出力レベルの計算結果は、0 ～ 1 の浮動小数点数で内部管理されています。
（データ定義内で指定される出力レベル値を、peak値で除算し、0 ～ 1 に変換して使っています）
ただし、出力レベルの計算結果は、
　　（レベル値 + offset値）÷ peak値
の結果が、0 ～ 1 の範囲に収まらない場合はエラーになります。

offset：（省略可能。省略時の値は0）
offset=[]では、エンベロープの出力レベルに対するオフセット値を指定します。オフセット値の加算は、peak値による除算を行う前に実行されます。

音量エンベロープでは、リリース中の無限演奏状態を避けるため、リリース最終節のレベルをゼロにしなければならない制限があります。
そのため、オフセット値を 0 以外にした場合、オフセット値×(-1)のレベルでリリース最終節を終わらせる必要がある点に注意してください。

init：
init=[]では、エンベロープ開始時点の出力レベルを、数値または「&」記号で指定します。
「&」を指定したとき：
ノートオン時の初期レベルが、直前までの演奏による出力レベルを引き継いだ値になります。
数値を指定したとき：
ノートオン時の初期レベルが、指定した数値になります。
数値の指定範囲は、0 ～ peak値 です。

ENV節：
ENV節は、「:」で区切られた３個１組のパラメータを、「,」区切りで複数列挙したものです。
ENV節は、２組以上定義します。（最大2048組まで）
２組とは、最低でもエンベロープ開始時点の節と、リリース開始時点の節が必要なためです。
ENV節内の要素の名前を、次のように決め、それぞれについて説明します。
　　[ptr]:[rate]:[lv],

ENV節の要素[ptr]：
[ptr]には、今回のエンベロープ節の種類を、N、L、R の３択で指定します（大文字でも小文字でも指定可能）。
　　N：通常のエンベロープの節
　　L：ループエントリのエンベロープの節
　　R：リリースエントリのエンベロープの節
L の節は、R より前の節で定義しなければなりません。
L の節は、１つのエンベロープ定義で１回だけ使用できますが、０回でも構いません。
R の節は、１つのエンベロープ定義で１回だけ、必ず使用しなければなりません。
L の節は、R の直前の節を、サスティンの末尾と解釈することで、サスティンの末尾の節から L の節へ無条件ジャンプするポイントになりますが、L の節が、サスティン末尾であった場合、ループリクエストは無視されます。（自分自身にループジャンプはしません）

ENV節の要素[rate]：
[rate]には、今回のエンベロープ節における変化レートを、
0 以上の数値で指定します。

変化レートの数値は、先頭１文字のアルファベット（大文字小文字問わず）でモード指定を記述することで、２種類の解釈がなされます。
モード指定は省略可能で、省略した場合、T を指定したものとみなされます。

T（ティー）を指定したとき：
[rate]の解釈を、時間（time）とします。
変化時間を固定的に指定します。
時間の単位は、#MB:CONFIGまたは @eocコマンドの、エンベロープクロック設定に依存します。
このモードでは、ディケイ時など、始点レベルと到達点レベルの差が毎回固定的な場合は変化の傾きが一定になりますが、リリース時など、始点レベルと到達点レベルの差が一定でない場合は、時間が固定される都合、変化の傾きが変わります。

I（アイ）を指定したとき：
[rate]の解釈を、変化率（勾配:incline）とします。
時間当たりの変化量を、
　　最大変移量 ÷ rate値
で求めることで、固定化された変化量を設定できます。
そのため、リリース時など、始点レベルと到達点レベルの差が一定でない場合は、変化の傾きが固定される都合、時間が変わります。

[rate]の値が大きいほどゆっくり推移します。0 のときは時間 0 で推移終了します。

【例１】

```
I3
```

変化率３の記述。

【例２】

```
10
```

時間１０の記述。この例ではTが省略されている状態。

ENV節の要素[lv]：
[lv]には、今回のエンベロープ節における到達点レベルを、数値で指定します。
数値の指定範囲は、0 ～ peak値 です。
前回のレベルと今回のレベルが同じ場合で、[rate] が I0 または T0 の場合、時間 0 で何もせず次のエンベロープ節に移ります。

7.5. @ef[1]：フィルタエンベロープ

7.6. #MB:ENV_F：フィルタエンベロープ定義

【記述例】

```
#MB:ENV_A @ea=1 {
  peak=15, init=0, |
  n:1:15, n:24:13, r:96:0
}
#MB:ENV_F @ef=1 {
  peak=100, init=0, |
  n:10:100, n:20:100, n:120:30, r:60:0
}
#MB:FILTER_D @fd=0 { type=lpf_h, depth_@ef=100, freq=40, resonance=0 }

t120 q11,16 v15 @@"pls" @ea1 @ef1 @fd0
l2 o5 cegefdgg cegefdg&g;
```

このように書くと、

- #MB:ENV_F によって、定義番号１番にフィルタエンベロープ内容を割り当てています。
- @ef1によって、定義番号１番の #MB:ENV_F の内容を適用。

といった定義と適用を記述できます。

【備考】記述例のフィルタエンベロープ定義の内容
レベル０で開始、
時間１０で最大（１００）、時間２０でレベル１００を維持、
時間１２０でレベル３０に減衰、以後レベル３０を維持。
以上の流れのどこでノートオフになるかは音長次第ですが、
ノートオフになったら、
時間６０でレベル０になる。

【解説】
フィルタエンベロープコマンド（@ef[1]）：
フィルタエンベロープ定義（#MB:ENV_F）の定義番号を整数で指定します。
定義番号は 0 ～ 1023 の整数です。
未定義の番号を指定するとエラーになります。
トラック先頭における初期設定には定義番号は無く、システム側で用意された暫定的なフィルタエンベロープが掛かっています。
@efコマンドは、@fdコマンド（動的フィルタ）が有効の場合に機能します。

フィルタエンベロープ定義（#MB:ENV_F）：
フィルタエンベロープコマンド（@ef[1]）で使用する、フィルタエンベロープ設定を定義します。

フィルタエンベロープ定義方法の概要は、
・行頭からの記述で、キーワード「#MB:ENV_F」を書く。
・続けて、スペースを置き、「@ef=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られた設定データを書く。
です。

定義番号：
定義番号は、@efコマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 1023 です。

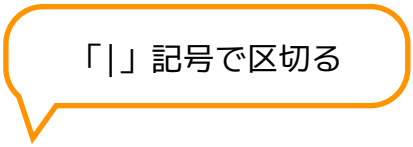
設定データ：

【記述例】

```
#MB:ENV_F @ef=1 {
  peak=100, init=0, |
  n:10:100, n:20:100, n:120:30, r:60:0
}
```

【解説】

フォーマットは次の通りです。




```
#MB:ENV_F @ef=定義番号 {
      peak=[], offset=[], init=[], |
      [ENV節],[ENV節],...（少なくとも２個以上のENV節）
}
```

peak：

peak=[]では、エンベロープの出力レベルの最大値を、0 より大きい値で指定します。
出力レベルの計算結果は、0 ～ 1 の浮動小数点数で内部管理されています。
（データ定義内で指定される出力レベル値を、peak値で除算し、0 ～ 1 に変換して使っています）
ただし、出力レベルの計算結果は、
　　（レベル値 + offset値）÷ peak値
の結果が、0 ～ 1 の範囲に収まらない場合はエラーになります。

offset：（省略可能。省略時の値は0）

offset=[]では、エンベロープの出力レベルに対するオフセット値を指定します。オフセット値の加算は、peak値による除算を行う前に実行されます。

フィルタエンベロープでは、音量エンベロープと違い、 リリース最終節のレベルをゼロにする必要はありません。

init：

init=[]では、エンベロープ開始時点の出力レベルを、数値または「&」記号で指定します。

「&」を指定したとき：

ノートオン時の初期レベルが、直前までの演奏による出力レベルを引き継いだ値になります。

数値を指定したとき：

ノートオン時の初期レベルが、指定した数値になります。
数値の指定範囲は、0 ～ peak値 です。

ENV節：

ENV節は、「:」で区切られた３個１組のパラメータを、「,」区切りで複数列挙したものです。
ENV節は、２組以上定義します。（最大2048組まで）
２組とは、最低でもエンベロープ開始時点の節と、リリース開始時点の節が必要なためです。
ENV節内の要素の名前を、次のように決め、それぞれについて説明します。

```
    [ptr]:[rate]:[lv],
```

ENV節の要素[ptr]：

[ptr]には、今回のエンベロープ節の種類を、N、L、R の３択で指定します（大文字でも小文字でも指定可能）。

N：通常のエンベロープの節
L：ループエントリのエンベロープの節
R：リリースエントリのエンベロープの節

L の節は、R より前の節で定義しなければなりません。
L の節は、１つのエンベロープ定義で１回だけ使用できますが、０回でも構いません。
R の節は、１つのエンベロープ定義で１回だけ、必ず使用しなければなりません。
L の節は、R の直前の節を、サスティンの末尾と解釈することで、サスティンの末尾の節から L の節へ無条件ジャンプするポイントになりますが、L の節が、サスティン末尾であった場合、ループリクエストは無視されます。（自分自身にループジャンプはしません）

ENV節の要素[rate]：

[rate]には、今回のエンベロープ節における変化レートを、
0 以上の数値で指定します。

変化レートの数値は、先頭１文字のアルファベット（大文字小文字問わず）でモード指定を記述することで、２種類の解釈がなされます。
モード指定は省略可能で、省略した場合、T を指定したものとみなされます。

T（ティー）を指定したとき：

[rate]の解釈を、時間（time）とします。
変化時間を固定的に指定します。
時間の単位は、#MB:CONFIGまたは @eocコマンドの、エンベロープクロック設定に依存します。
このモードでは、ディケイ時など、始点レベルと到達点レベルの差が毎回固定的な場合は変化の傾きが一定になりますが、リリース時など、始点レベルと到達点

レベルの差が一定でない場合は、時間が固定される都合、変化の傾きが変わります。

I（アイ）を指定したとき：
[rate]の解釈を、変化率（勾配:incline）とします。
時間当たりの変化量を、
　　最大変移量 ÷ rate値
で求めることで、固定化された変化量を設定できます。
そのため、リリース時など、始点レベルと到達点レベルの差が一定でない場合は、変化の傾きが固定される都合、時間が変わります。
[rate]の値が大きいほどゆっくり推移します。0 のときは時間 0 で推移終了します。

【例１】

```
I3
```

変化率３の記述。

【例２】

```
10
```

時間１０の記述。この例ではTが省略されている状態。

ENV節の要素[lv]：
[lv]には、今回のエンベロープ節における到達点レベルを、数値で指定します。
数値の指定範囲は、0 ～ peak値 です。
前回のレベルと今回のレベルが同じ場合で、[rate] が I0 または T0 の場合、時間 0 で何もせず次のエンベロープ節に移ります。

7.7. @zr[str],[1]：音量エンベロープのダンパー設定

【記述例】

```
@zr"env",0.1
```

この場合、音量エンベロープ（@ea）のダンパーレートを 0.1 にします。

```
@zr"fms",12
```

この場合、ＦＭ音源（@@"fms"）の音量エンベロープのダンパーレートを 12 にします。

【解説】

音量エンベロープの、ダンパー（消音機能）設定を行います。
当コマンドは、zコマンドおよび@azコマンドで実行されるダンパー機能の設定を変更したい場合に利用します。
トラック先頭における初期設定は、

```
#MB:CONFIG { damper_rate: env=0.05: fms=15, }
```

による設定があればその内容に従います。
（なければ上記CONFIGのdamper_rate初期値に従います）

指定する引数は次の通りです。
引数[str]...宛先（destination）
引数[1].....レート（rate）
引数はカンマで区切って指定します。

引数[str]（destination）

ダンパー設定の宛先を、３文字の識別子で指定します。

【@zr[str],[1]：宛先の識別子一覧】

env	@eaコマンドの音量エンベロープ
fms	ＦＭ音源（@@"fms"）のエンベロープ

引数[1]（rate）

ダンパーレートを指定します。

destination が「env」のとき

rateは 0 以上の数値で指定します。小数以下の指定も受け付けます。
指定値はエンベロープ定義内における変化率（ I のモード）によるレートとして受け付けられます。減衰時間はエンベロープクロック（時間単位）の設定に依存します。

destination が「fms」のとき

rateは 0 ～ 15 の整数で指定します。
ダンパー実行時に、全てのオペレータのリリースレートが、指定値に書き換えられます。

7.8. z：エンベロープダンパー（消音機能）実行

【記述例】

```
#MB:ENV_A @ea=1 { peak=15, init=8, |
   n:2:15, n:12:13, r:100:0
}
t120 q16,16 v15 @@"pls" @ea1 l8 o5
crergrer fzr dzr gzr gzr
crergrer fzr dzr g&gzr;
```

この場合、音符の直後が「r」になっている部分はリリース音が残りますが、
音符の直後が「zr」になっている部分はリリース音が残らず消音されます。

【解説】

エンベロープのダンパー（消音機能）を実行します。
引数はありません。

実行されるダンパーは、メタデータ設定「#MB:CONFIG { damper_rate:...}」または、@zrコマンドによる設定内容に従います。

当コマンドでは、音量エンベロープ（@ea）または、ＦＭ音源（@@"fms"）エンベロープのうち、演奏中であるリリースを臨時的に書き換えて、即時無音までリリースを終結させる用途で使用します。書き換えたリリースは、次回ノートオンの際、自動的に元のレートに戻されます。

例えば、リリースを長く設定したエンベロープで、ブレイクしたい休符前などに zコマンドを配置することで、即時消音したいときに使用します。

スラー中に zコマンドを使用した場合は、強制的にスラーが中断（ノートオフ）されて、zコマンド直後の音符があらためてノートオンされます。

7.9. @az[1]：オートダンパー機能

【記述例】

```
#MB:CONFIG {
   env_clock: unit=sec: rate=1/120,
   env_resol: unit=smp,
   damper_rate: env=0.03: fms=15,
}
#MB:ENV_A @ea=1 { peak=15, init=&, | n:5:15, r:180:0 }
t120 q8,8 v15 @@"pls" @ea1 l8 o5
@az0  crergrer fr dr gr gr z r4
@az1  crergrer fr dr gr gr z r4
q4    c4e4g4e4 f4 d4 g4 g4 z r4
q8@q3 c4e4g4e4 f4 d4 g4 g4 z r4;
```

この場合、@az1以降の音符が、オートダンパーが有効になります。

【解説】

エンベロープのオートダンパー（自動消音機能）を設定します。
引数[1]には、次のノートオンの何tickカウント手前でダンパーを掛けるかを、数値で指定します。
設定範囲は 0 以上の整数です。
引数[1]が 0 の場合、オートダンパーは無効です。
トラック先頭における初期設定は、@az0 です。

実行されるダンパーは、メタデータ設定「#MB:CONFIG { damper_rate:...}」または、@zrコマンドによる設定内容に従います。

オートダンパーの使用目的は、アタックが緩いエンベロープの「緩さ」を強調することです。

【備考】オートダンパーの挙動
オートダンパー有効の際は、「次のノートオンの何tickカウント手前か」を判断するため、ＭＭＬコンパイラは、音符または休符を処理する時、次に来るのが何かを先読みしています。
先読み処理中、
今回が音符だった場合：

先読み処理では、まず、次に来るのが音符か休符かトラック終端が現れるまで、その他のコマンドを読み飛ばします。
次回が音符だった場合：
引数[1]のtickカウント手前でダンパー処理をし、引数[1]だけ休符。
次回がスラーでつなぐ音符だった場合：
今回の音符においてダンパー処理関係は何もしない。
次回が休符だった場合：
今回の音符においてダンパー処理関係は何もしない。
次回がトラック終端だった場合：
音符だった場合と同様。

今回が休符だった場合：
先読み処理では、まず、次に来るのが音符か休符かトラック終端が現れるまで、その他のコマンドを読み飛ばします。
次回が音符だった場合：
引数[1]のtickカウント手前でダンパー処理をし、引数[1]だけ休符。
次回が休符だった場合：
今回の音符においてダンパー処理関係は何もしない。
次回がトラック終端だった場合：
音符だった場合と同様。

ただし、今回が音符だった場合の「引数[1]のtickカウント手前」の判断は、
q / @q におけるゲート処理も有効だった場合、
ゲートによる休符部分内に「手前」が収まっていればそこでダンパー処理をしますが、ゲートによる休符部分よりも「手前」だった場合には、「手前」が優先されてゲートより前にキーオフ＆ダンパー処理が入り、残りが休符になります。

さらに、「手前」が今回の音符の長さ以上になる場合は、オートダンパーによるダンパー機能は無効になります。例えば、@az1では、tickカウント１の音符にはオートダンパーは掛かりません。

8. ＬＦＯ関連

8.1. @loc[str],[1]：ＬＦＯオプション CLOCK

8.2. @lor[str],[1]：ＬＦＯオプション RESOLUTION

【記述例】

```
#MB:ENV_A @ea=1 { peak=15, init=& | n:0:15, r:1:0 }
#MB:LFO_P @lp=1 { depth=400, width=24, delay=48, form=sin }

t120 q12,16 v15 @@"pls" o5 @ea1
@loc"sec",1/90
@lor"sec",1/600
@lp1 c1&2;
```

【解説】

ＬＦＯ全種類への、オプション設定を行います。
@loc[str],[1] および @lor[str],[1] は、

```
#MB:CONFIG {
   lfo_clock: unit=sec: rate=1/60,
   lfo_resol: unit=sec: rate=1/300,
}
```

による設定をトラックごとに変更したい場合に使用します。

@loc[str],[1]
ＬＦＯ全種類への、時間単位を設定します。
時間単位は、width の周期時間と、delay に影響します。

指定する引数は次の通りです。
引数[str] ：時間単位
引数[1] ：時間単位への倍率
引数はカンマで区切って指定します。

【@locの設定内容】

引数[str]	引数[str]の設定内容と、引数[1]の設定範囲
"sec"	時間単位が「１秒」になります。 引数[1]の設定範囲は、1/2400[秒] ～ 1.0[秒] です。
"tick"	時間単位は「１tickカウント」になります。 引数[1]の設定範囲は、０より大きい値です。 ただし、最小は1/2400[秒]に制限されます。

@lor[str],[1]
ＬＦＯ時間軸の解像度（ＬＦＯ変位計算の最少時間単位）を設定します。
この設定が有効になる対象は、
　　ピッチＬＦＯ
　　yコマンドＬＦＯ
　　フィルタＬＦＯ
の３種類です。

ＬＦＯ解像度は、ポルタメント機能による周波数変更の最小時間単位にもなっています。

指定する引数は次の通りです。
引数[str]　：解像度の時間単位
引数[1]　　：解像度の時間単位への倍率
引数はカンマで区切って指定します。

【@lorの設定内容】

引数[str]	引数[str]の設定内容と、引数[1]の設定範囲
"sec"	解像度の時間単位が「１秒」になります。 引数[1]の設定範囲は、1/2400[秒] ～ 1.0[秒] です。
"tick"	解像度の時間単位は「１tickカウント」になります。 引数[1]の設定範囲は、０より大きい値です。 ただし、最小は1/2400[秒]に制限されます。

【備考】
時間単位が「１tickカウント」の場合、テンポにより時間単位が変動します。
エンベロープ指定後にテンポを変更した場合、テンポ変更後のノートオンからエンベロープの時間計算が自動追従します。
つまり、ノートオン中にテンポ変更が掛かった場合は、そのノートオン中は時間計算の自動追従はできませんが、次のノートオンから自動追従します。

8.3. @lp[1]：ピッチＬＦＯ

8.4. #MB:LFO_P：ピッチＬＦＯ定義

【記述例】

```
#MB:CONFIG {
   lfo_clock: unit=tick: rate=1,
   lfo_resol: unit=sec: rate=1/300,
}
#MB:LFOTBL_ATK 1 {
   loop=0, cmpl=0, offset=0, denom=1, width_mode=step, |
   0, 1, 0, -1,
}
#MB:ENV_A @ea=1 { peak=15, init=& | n:1:15, r:1:0 }
#MB:LFO_P @lp=0  { depth=400, width=24, delay=48, form=sin }
#MB:LFO_P @lp=1  { depth=400, width=24, delay=48, form=sin_u }
#MB:LFO_P @lp=10 { depth=400, width=24, delay=48, form=saw }
#MB:LFO_P @lp=11 { depth=400, width=24, delay=48, form=saw_u }
#MB:LFO_P @lp=12 { depth=400, width=24, delay=48, form=saw_os }
#MB:LFO_P @lp=20 { depth=400, width=24, delay=48, form=tri }
#MB:LFO_P @lp=21 { depth=400, width=24, delay=48, form=tri_u }
#MB:LFO_P @lp=30 { depth=400, width=24, delay=48, form=pls }
#MB:LFO_P @lp=31 { depth=400, width=24, delay=48, form=pls_u }
#MB:LFO_P @lp=40 { depth=400, width=4,  delay=48, form=nzw }
#MB:LFO_P @lp=41 { depth=400, width=4,  delay=48, form=nzw_u }
#MB:LFO_P @lp=50 { depth=400, width=24, delay=48, form=table, tbl_atk=1 }
#MB:LFO_P @lp=60 { depth=400, width=4,  delay=48, form=nlbend, bendrate=64 }

t120 q12,16 v15 @@"pls" o5 @ea1
@lp0 c1&2 @lp1 c1&2
@lp10 c1&2 @lp11 c1&2 @lp12 c1&2
@lp20 c1&2 @lp21 c1&2
@lp30 c1&2 @lp31 c1&2
@lp40 c1&2 @lp41 c1&2
@lp50 c1&2
@lp60 c1&2;
```

このように書くと、#MB:LFO_P と @lpコマンドによって、ピッチＬＦＯの定義と適用を記述できます。

【解説】
ピッチＬＦＯコマンド（@lp[1]）：
ピッチＬＦＯ定義（#MB:LFO_P）の定義番号を整数で指定します。
定義番号は 0 ～ 1023 の整数です。
未定義の番号を指定するとエラーになります。
トラック先頭における初期設定では、ピッチＬＦＯは停止状態です。

ピッチＬＦＯ定義（#MB:LFO_P）：
ピッチＬＦＯコマンド（@lp[1]）で使用する、ピッチＬＦＯ設定を定義します。

ピッチＬＦＯ定義方法の概要は、
・行頭からの記述で、キーワード「#MB:LFO_P」を書く。
・続けて、スペースを置き、「@lp=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られた設定データを書く。
です。

定義番号：
定義番号は、@lpコマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 1023 です。

設定データ：

【記述例】

```
#MB:LFO_P @lp=0  { depth=400, width=24, delay=48, form=sin }
```

【解説】
フォーマットは次の通りです。

```
#MB:LFO_P @lp=定義番号 {
     depth=[], width=[], delay=[], form=[],
     pwm=[],          (form=plsまたはform=pls_uの場合に指定可能)
     tbl_atk=[],      (form=tableの場合に指定可能)
     tbl_rel=[],      (form=tableの場合に指定可能)
     bendrate=[],     (form=nlbendの場合に指定可能)
}
```

depth：
depth=[]では、揺らす音程の振幅を数値で指定します。
単位は、psコマンドで設定されている音程の解像度です。
ゼロを指定した場合、ピッチＬＦＯが無効になります。
負数を指定した場合、変化パターンの上下が反転した動作になります。
振幅は「depth × formの波形変位」で得られます。
formの波形変位は、通常 -1 ～ 1 ですが、
ユーザー定義テーブルの場合は、自由に変位が定義できるため、振幅演算が
「depth × テーブル値」となる点に注意してください。

width：
width=[]では、formの指定内容によって、widthの時間単位が変わります。

formが、sin*,saw*,tri*,pls*, のとき：
width は form波形１周期の長さを数値で指定します。
数値は１以上の値を指定してください。
周期の時間単位の初期設定は「１tickカウント」ですが、ＬＦＯクロック設定（@locコマンドなど）により、テンポに依存しない時間単位（0.01秒など）にも出来ます。

formが、nzw*,table,nlbend, のとき：
width は ＬＦＯステップ数を数値で指定します。
数値は１以上の値で指定してください。
ＬＦＯステップ数は、ＬＦＯ解像度設定（@lorコマンドなど）で決まる、ＬＦＯ専用の最少時間単位です。
さらに、form=table の場合は、ユーザー定義テーブル内の設定で、widthの動作モードを、ＬＦＯステップ数とＬＦＯシフト数の２つから選択します。
詳細は「#MB:LFOTBL_ATK」を参照してください。

delay：
delay=[]では、ノートオンからＬＦＯ開始までの遅延時間を数値で指定します。
時間単位は width 同様です。

delay が 0 以上のとき：
ノートオンの都度、ＬＦＯシーケンスが再スタートします。
（ノートオン同期モード）

delay が負数のとき：
ノートオンでＬＦＯシーケンスが再スタートしない、ノートオン非同期モードとなります。いかなる負数の値も、このモード指定と判定されます。

ノートオン非同期モードにおいて、任意のタイミングでＬＦＯ位相のリセットを行いたい場合は、ＬＦＯの再スタートコマンドを使用します。
ＬＦＯコマンドの指定時には、同期・非同期モードに関わらずＬＦＯシーケンスは位相ゼロからスタートします。

form：

form=[]では、変化パターンの波形を指定します。（波形パターン詳細）

【formと波形の対応表】（#MB:LFO_P）

form	波形
sin	サイン波
sin_u	調整サイン波
saw	ノコギリ波
saw_u	調整ノコギリ波
saw_os	ノコギリ波ワンショット
tri	三角波
tri_u	調整三角波
pls	パルス波
pls_u	調整パルス波
nzw	ホワイトノイズ
nzw_u	調整ホワイトノイズ
table	ユーザー定義テーブル
nlbend	ノンリニアベンド

form=pls* のとき：

追加で次の項目を設定します。
　　pwm=[],　　（任意：デューティ比）
pwmの設定範囲は 0.05 ～ 0.995 です。
pwmを省略した場合の初期設定は「pwm=0.5」（デューティー比=50%）です。

form=table のとき：

追加で次の項目を設定します。
　　tbl_atk=[],　　（必須：#MB:LFOTBL_ATKで定義したテーブル番号）
　　tbl_rel=[],　　（任意：#MB:LFOTBL_RELで定義したテーブル番号）
tbl_relを省略した場合、常に「tbl_rel=-1」と解釈され、tbl_relは未使用扱いになります。

tbl_atkの設定範囲は 0 ～ 4095 です。
tbl_relの設定範囲は 0 ～ 4095 です。
ただし、いずれも、未定義のテーブル番号を指定した場合、ＬＦＯ変化が無効になります（変化のないダミーテーブルが参照されます）。

form=nlbend のとき：
追加で次の項目を設定します。
　　bandrate=[],　（必須：ノンリニアベンドの変化率）
bendrateの設定範囲は 1.0 ～ 512.0 です。
1未満もしくは512超えの指定は、それぞれ1、512に制限されます。

8.5. @la[1]：音量ＬＦＯ

8.6. #MB:LFO_A：音量ＬＦＯ定義

【記述例】

```
#MB:CONFIG {
   lfo_clock: unit=tick: rate=1,
   lfo_resol: unit=sec: rate=1/300,
}
#MB:ENV_A @ea=1 { peak=15, init=& | n:1:15, r:1:0 }
#MB:LFOTBL_ATK 1 {
   loop=0, cmpl=0, offset=0, denom=1, width_mode=step, |
   0, -1.0, 0, -0.25,
}
#MB:LFO_A @la=0  { depth=4, width=24, delay=48, form=sin_a }
#MB:LFO_A @la=10 { depth=4, width=24, delay=48, form=saw_a }
#MB:LFO_A @la=20 { depth=4, width=24, delay=48, form=tri_a }
#MB:LFO_A @la=30 { depth=4, width=24, delay=48, form=pls_a }
#MB:LFO_A @la=40 { depth=4, width=4,  delay=48, form=nzw_a }
#MB:LFO_A @la=50 { depth=4, width=24, delay=48, form=table, tbl_atk=1 }

t120 q12,16 v15 @@"pls" o5 @ea1
@la0  c1&2 @la10 c1&2 @la20 c1&2
@la30 c1&2 @la40 c1&2 @la50 c1&2;
```

このように書くと、#MB:LFO_A と @laコマンドによって、音量ＬＦＯの定義と適用を記述できます。

【解説】

音量ＬＦＯコマンド（@la[1]）：
音量ＬＦＯ定義（#MB:LFO_A）の定義番号を整数で指定します。
定義番号は 0 ～ 1023 の整数です。
未定義の番号を指定するとエラーになります。
トラック先頭における初期設定では、音量ＬＦＯは停止状態です。

音量ＬＦＯ定義（#MB:LFO_A）：
音量ＬＦＯコマンド（@la[1]）で使用する、音量ＬＦＯ設定を定義します。

音量ＬＦＯ定義方法の概要は、

・行頭からの記述で、キーワード「#MB:LFO_A」を書く。
・続けて、スペースを置き、「@la=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られた設定データを書く。
です。

定義番号：

定義番号は、@laコマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 1023 です。

設定データ：

【記述例】

```
#MB:LFO_A @la=0  { depth=4, width=24, delay=48, form=sin_a }
```

【解説】

フォーマットは次の通りです。

```
#MB:LFO_A @la=定義番号 {
     depth=[], width=[], delay=[], form=[],
     pwm=[],          （form=plsまたはform=pls_uの場合に指定可能）
     tbl_atk=[],      （form=tableの場合に指定可能）
     tbl_rel=[],      （form=tableの場合に指定可能）
}
```

depth：

depth=[]では、揺らす音量の振幅を設定します。
単位は vsコマンドで設定されている音量スケールです。
ゼロを指定した場合、音量ＬＦＯが無効になります。
音量ＬＦＯの場合、音量変化の変位は、0 で始まり、変位を減らす方向で推移させていて、利得は発生させない（リミッターが掛かる）設計です。
音量変化の変位が 0 とは、指定の音量通りという状態です。
例えば、変化パターンのサイン波では 0 から（-1）までの間で変化するので、depth に３を指定していれば、音量変化は vsコマンドによるスケールで
0 ～（-3）で揺れることになります。
depthに負数を指定した場合、変化パターンの上下が反転しますが、利得が発生しないよう制限されるので、意味の無い状態になります。

form がユーザー定義テーブルの場合、depth はテーブル値への倍率として動作します。この場合、テーブル値の内容によって自由に変位が定義できてしまうため、演算結果の範囲が 0 ～ 負数 になるようテーブル値への配慮が必要です（正の数は利得とみなされて 0 に制限されるため）。

width：

width=[]では、formの指定内容によって、widthの時間単位が変わります。

formが、sin_a,saw_a,tri_a,pls_a, のとき：

width は form波形１周期の長さを数値で指定します。
数値は１以上の値を指定してください。
周期の時間単位の初期設定は「１tickカウント」ですが、ＬＦＯクロック設定（@locコマンドなど）により、テンポに依存しない時間単位（0.01秒など）にも出来ます。

formが、nzw_a,table, のとき：

width は ＬＦＯステップ数を数値で指定します。
数値は１以上の値で指定してください。
ＬＦＯステップ数は、ＬＦＯ解像度設定（@lorコマンドなど）で決まる、ＬＦＯ専用の最少時間単位です。
さらに、form=table の場合は、ユーザー定義テーブル内の設定で、widthの動作モードを、ＬＦＯステップ数とＬＦＯシフト数の２つから選択します。
詳細は「#MB:LFOTBL_ATK」を参照してください。

delay：

delay=[]では、ノートオンからＬＦＯ開始までの遅延時間を数値で指定します。
時間単位は width 同様です。

delay が 0 以上のとき：

ノートオンの都度、ＬＦＯシーケンスが再スタートします。
（ノートオン同期モード）

delay が負数のとき：

ノートオンでＬＦＯシーケンスが再スタートしない、ノートオン非同期モードとなります。いかなる負数の値も、このモード指定と判定されます。
ノートオン非同期モードにおいて、任意のタイミングでＬＦＯ位相のリセットを行いたい場合は、ＬＦＯの再スタートコマンドを使用します。
ＬＦＯコマンドの指定時には、同期・非同期モードに関わらずＬＦＯシーケンスは位相ゼロからスタートします。

form：
form=[]では、変化パターンの波形を指定します。（波形パターン詳細）

【formと波形の対応表】（#MB:LFO_A）

form	波形
sin_a	調整サイン波
saw_a	調整ノコギリ波
tri_a	調整三角波
pls_a	調整パルス波
nzw_a	調整ホワイトノイズ
table	ユーザー定義テーブル

form=pls_a のとき：
追加で次の項目を設定します。
　　pwm=[],　　（任意：デューティ比）
pwmの設定範囲は 0.05 ～ 0.995 です。
pwmを省略した場合の初期設定は「pwm=0.5」（デューティー比=50%）です。

form=table のとき：
追加で次の項目を設定します。
　　tbl_atk=[],　　（必須：#MB:LFOTBL_ATKで定義したテーブル番号）
　　tbl_rel=[],　　（任意：#MB:LFOTBL_RELで定義したテーブル番号）
tbl_relを省略した場合、常に「tbl_rel=-1」と解釈され、tbl_relは未使用扱いになります。
tbl_atkの設定範囲は 0 ～ 4095 です。
tbl_relの設定範囲は 0 ～ 4095 です。
ただし、いずれも、未定義のテーブル番号を指定した場合、ＬＦＯ変化が無効になります（変化のないダミーテーブルが参照されます）。

8.7. @lb[1]：パンポットバランスＬＦＯ

8.8. #MB:LFO_B：パンポットＬＦＯ定義

【記述例】

```
#MB:CONFIG {
   lfo_clock: unit=tick: rate=1,
   lfo_resol: unit=sec: rate=1/300,
}
#MB:LFOTBL_ATK 1 {
   loop=0, cmpl=0, offset=0, denom=1, width_mode=step, |
   0, 1, 0, -1,
}
#MB:ENV_A @ea=1 { peak=15, init=& | n:1:15, r:1:0 }
#MB:LFO_B @lb=0  { depth=100, width=24, delay=48, form=sin }
#MB:LFO_B @lb=1  { depth=100, width=24, delay=48, form=sin_u }
#MB:LFO_B @lb=10 { depth=100, width=24, delay=48, form=saw }
#MB:LFO_B @lb=11 { depth=100, width=24, delay=48, form=saw_u }
#MB:LFO_B @lb=12 { depth=100, width=24, delay=48, form=saw_os }
#MB:LFO_B @lb=20 { depth=100, width=24, delay=48, form=tri }
#MB:LFO_B @lb=21 { depth=100, width=24, delay=48, form=tri_u }
#MB:LFO_B @lb=30 { depth=100, width=24, delay=48, form=pls }
#MB:LFO_B @lb=31 { depth=100, width=24, delay=48, form=pls_u }
#MB:LFO_B @lb=40 { depth=100, width=4,  delay=48, form=nzw }
#MB:LFO_B @lb=41 { depth=100, width=4,  delay=48, form=nzw_u }
#MB:LFO_B @lb=50 { depth=100, width=24, delay=48, form=table, tbl_atk=1 }
#MB:LFO_B @lb=60 { depth=100, width=4,  delay=48, form=nlbend, bendrate=64 }

t120 q12,16 v15 @@"pls" o5 @ea1 @p0 p0
@lb0 c1&2 @lb1 c1&2
@lb10 c1&2 @lb11 c1&2 @lb12 c1&2
@lb20 c1&2 @lb21 c1&2
@lb30 c1&2 @lb31 c1&2
@lb40 c1&2 @lb41 c1&2
@lb50 c1&2
@lb60 c1&2;
```

このように書くと、#MB:LFO_B と @lbコマンドによって、
パンポットバランスＬＦＯの定義と適用を記述できます。

【解説】
パンポットバランスＬＦＯコマンド（@lb[1]）：
パンポットバランスＬＦＯ定義（#MB:LFO_B）の定義番号を整数で指定します。
定義番号は 0 ～ 1023 の整数です。
未定義の番号を指定するとエラーになります。
トラック先頭における初期設定では、パンポットバランスＬＦＯは停止状態です。

パンポットバランスＬＦＯ定義（#MB:LFO_B）：
パンポットバランスＬＦＯコマンド（@lb[1]）で使用する、パンポットバランスＬＦＯ設定を定義します。

パンポットバランスＬＦＯ定義方法の概要は、
・行頭からの記述で、キーワード「#MB:LFO_B」を書く。
・続けて、スペースを置き、「@lb=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られた設定データを書く。
です。

定義番号：
定義番号は、@lbコマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 1023 です。

設定データ：

【記述例】

```
#MB:LFO_B @lb=0  { depth=100, width=24, delay=48, form=sin }
```

【解説】
フォーマットは次の通りです。

```
#MB:LFO_B @lb=定義番号 {
     depth=[], width=[], delay=[], form=[],
     pwm=[],          (form=plsまたはform=pls_uの場合に指定可能)
     tbl_atk=[],      (form=tableの場合に指定可能)
     tbl_rel=[],      (form=tableの場合に指定可能)
     bendrate=[],     (form=nlbendの場合に指定可能)
}
```

depth：
depth=[]では、揺らすパンポットの振幅を、-200 ～ 200 で指定します。
少数以下の指定も受け付けます。
ゼロを指定した場合、ＬＦＯが無効になります。
負数を指定した場合、変化パターンの上下が反転した動作になります。
パンポットが @p0 の状態で、depth が 100 で三角波のＬＦＯをかけた場合、パンポットは最大幅で揺れます。上限の絶対値が200である理由は、パンポットを片方いっぱいに寄せてから、変位がもう片方にしか振れない波形を使用する場合を想定しています。
パンポットの制限値を超えるＬＦＯをかけた場合は、超えた部分が制限値にリミットされながらＬＦＯがかかります。
振幅は「depth × formの波形変位」で得られます。
formの波形変位は、通常 -1 ～ 1 ですが、
ユーザー定義テーブルの場合は、自由に変位が定義できるため、振幅演算が「depth × テーブル値」となる点に注意が必要です。

width：
width=[]では、formの指定内容によって、widthの時間単位が変わります。

formが、sin*,saw*,tri*,pls*, のとき：
width は form波形１周期の長さを数値で指定します。
数値は１以上の値を指定してください。
周期の時間単位の初期設定は「１tickカウント」ですが、ＬＦＯクロック設定（@locコマンドなど）により、テンポに依存しない時間単位（0.01秒など）にも出来ます。

formが、nzw*,table,nlbend, のとき：
width は ＬＦＯステップ数を数値で指定します。
数値は１以上の値で指定してください。
ＬＦＯステップ数は、ＬＦＯ解像度設定（@lorコマンドなど）で決まる、ＬＦＯ専用の最少時間単位です。
さらに、form=table の場合は、ユーザー定義テーブル内の設定で、widthの動作モードを、ＬＦＯステップ数とＬＦＯシフト数の２つから選択します。
詳細は「#MB:LFOTBL_ATK」を参照してください。

delay：
delay=[]では、ノートオンからＬＦＯ開始までの遅延時間を数値で指定します。
時間単位は width 同様です。

delay が 0 以上のとき：

ノートオンの都度、ＬＦＯシーケンスが再スタートします。
（ノートオン同期モード）

delay が負数のとき：
ノートオンでＬＦＯシーケンスが再スタートしない、ノートオン非同期モードとなります。いかなる負数の値も、このモード指定と判定されます。
ノートオン非同期モードにおいて、任意のタイミングでＬＦＯ位相のリセットを行いたい場合は、ＬＦＯの再スタートコマンドを使用します。
ＬＦＯコマンドの指定時には、同期・非同期モードに関わらずＬＦＯシーケンスは位相ゼロからスタートします。

form：
form=[]では、変化パターンの波形を指定します。（波形パターン詳細）

【formと波形の対応表】（#MB:LFO_B）

form	波形
sin	サイン波
sin_u	調整サイン波
saw	ノコギリ波
saw_u	調整ノコギリ波
saw_os	ノコギリ波ワンショット
tri	三角波
tri_u	調整三角波
pls	パルス波
pls_u	調整パルス波
nzw	ホワイトノイズ
nzw_u	調整ホワイトノイズ
table	ユーザー定義テーブル
nlbend	ノンリニアベンド

form=pls* のとき：
追加で次の項目を設定します。
　　pwm=[],　　（任意：デューティ比）
pwmの設定範囲は 0.05 ～ 0.995 です。
pwmを省略した場合の初期設定は「pwm=0.5」（デューティー比=50%）です。

form=table のとき：

追加で次の項目を設定します。

tbl_atk=[],　（必須：#MB:LFOTBL_ATKで定義したテーブル番号）

tbl_rel=[],　（任意：#MB:LFOTBL_RELで定義したテーブル番号）

tbl_relを省略した場合、常に「tbl_rel=-1」と解釈され、tbl_relは未使用扱いになります。

tbl_atkの設定範囲は 0 ～ 4095 です。

tbl_relの設定範囲は 0 ～ 4095 です。

ただし、いずれも、未定義のテーブル番号を指定した場合、ＬＦＯ変化が無効になります（変化のないダミーテーブルが参照されます）。

form=nlbend のとき：

追加で次の項目を設定します。

bandrate=[],　（必須：ノンリニアベンドの変化率）

bendrateの設定範囲は 1.0 ～ 512.0 です。

1未満もしくは512超えの指定は、それぞれ1、512に制限されます。

8.9. @lf[1]：フィルタＬＦＯ

8.10. #MB:LFO_F：フィルタＬＦＯ定義

【記述例】

```
#MB:CONFIG {
   lfo_clock: unit=tick: rate=1,
   lfo_resol: unit=sec: rate=1/300,
}
#MB:LFOTBL_ATK 1 {
   loop=0, cmpl=0, offset=0, denom=1, width_mode=step, |
   0, 1.0, 0, 0.5,
}
#MB:ENV_A @ea=1 { peak=15, init=& | n:1:15, r:1:0 }
#MB:LFO_F @lf=1  { depth=-12, width=24, delay=24, form=sin_u }
#MB:LFO_F @lf=11 { depth=-12, width=24, delay=24, form=saw_u }
#MB:LFO_F @lf=21 { depth=-12, width=24, delay=24, form=tri_u }
#MB:LFO_F @lf=31 { depth=-12, width=24, delay=24, form=pls_u }
#MB:LFO_F @lf=41 { depth=-12, width=4,  delay=24, form=nzw_u }
#MB:LFO_F @lf=50 { depth=-12, width=24, delay=24, form=table, tbl_atk=1 }
#MB:LFO_F @lf=60 { depth=-12, width=4,  delay=24, form=nlbend, bendrate=64 }

#MB:FILTER_D @fd=1 { type=lpf_h, depth_@ef=0, freq=24000, resonance=0 }

t90 q12,16 v15 @@"pls" o5 @ea1 @fd1
@lf1  c1 @lf11 c1 @lf21 c1 @lf31 c1
@lf41 c1 @lf50 c1 @lf60 c1;
```

このように書くと、#MB:LFO_F と @lfコマンドによって、フィルタＬＦＯの定義と適用を記述できます。

フィルタＬＦＯは、#MB:LFO_F と @lfコマンドに加え、@fdコマンドで、フィルタが掛かった状態で使用します。
フィルタＬＦＯで操作するのは、@fdコマンドで適用されているフィルタの、カットオフ周波数が対象になります。

【解説】
フィルタＬＦＯコマンド（@lf[1]）：
フィルタＬＦＯ定義（#MB:LFO_F）の定義番号を整数で指定します。
定義番号は 0 ～ 1023 の整数です。
未定義の番号を指定するとエラーになります。
トラック先頭における初期設定では、フィルタＬＦＯは停止状態です。

フィルタＬＦＯ定義（#MB:LFO_F）：
フィルタＬＦＯコマンド（@lf[1]）で使用する、フィルタＬＦＯ設定を定義します。

フィルタＬＦＯ定義方法の概要は、
・行頭からの記述で、キーワード「#MB:LFO_F」を書く。
・続けて、スペースを置き、「@lf=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られた設定データを書く。
です。

定義番号：
定義番号は、@lfコマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 1023 です。

設定データ：

【記述例】

```
#MB:LFO_F @lf=1  { depth=-12, width=24, delay=24, form=sin_u }
```

【解説】
フォーマットは次の通りです。

```
#MB:LFO_F @lf=定義番号 {
     depth=[], width=[], delay=[], form=[],
     pwm=[],          (form=plsまたはform=pls_uの場合に指定可能)
     tbl_atk=[],      (form=tableの場合に指定可能)
     tbl_rel=[],      (form=tableの場合に指定可能)
     bendrate=[],     (form=nlbendの場合に指定可能)
}
```

depth：
depth=[]では、揺らすカットオフの振幅を 0 ～ 100 で指定します。
ゼロを指定した場合、ＬＦＯが無効になります。
負数を指定した場合、変化パターンの上下が反転した動作になります。
振幅は「depth × formの波形変位」で得られます。
formの波形変位は、通常 -1 ～ 1 ですが、
ユーザー定義テーブルの場合は、自由に変位が定義できるため、振幅演算が
「depth × テーブル値」となる点に注意してください。

width：
width=[]では、formの指定内容によって、widthの時間単位が変わります。

formが、sin*,saw*,tri*,pls*, のとき：
width は form波形１周期の長さを数値で指定します。
数値は１以上の値を指定してください。
周期の時間単位の初期設定は「１tickカウント」ですが、ＬＦＯクロック設定（@locコマンドなど）により、テンポに依存しない時間単位（0.01秒など）にも出来ます。

formが、nzw*,table,nlbend, のとき：
width は ＬＦＯステップ数を数値で指定します。
数値は１以上の値で指定してください。
ＬＦＯステップ数は、ＬＦＯ解像度設定（@lorコマンドなど）で決まる、ＬＦＯ専用の最少時間単位です。
さらに、form=table の場合は、ユーザー定義テーブル内の設定で、widthの動作モードを、ＬＦＯステップ数とＬＦＯシフト数の２つから選択します。
詳細は「#MB:LFOTBL_ATK」を参照してください。

delay：
delay=[]では、ノートオンからＬＦＯ開始までの遅延時間を数値で指定します。
時間単位は width 同様です。

delay が 0 以上のとき：
ノートオンの都度、ＬＦＯシーケンスが再スタートします。
（ノートオン同期モード）

delay が負数のとき：
ノートオンでＬＦＯシーケンスが再スタートしない、ノートオン非同期モードとなります。いかなる負数の値も、このモード指定と判定されます。

ノートオン非同期モードにおいて、任意のタイミングでＬＦＯ位相のリセットを行いたい場合は、ＬＦＯの再スタートコマンドを使用します。
ＬＦＯコマンドの指定時には、同期・非同期モードに関わらずＬＦＯシーケンスは位相ゼロからスタートします。

form：

form=[]では、変化パターンの波形を指定します。（波形パターン詳細）

【formと波形の対応表】（#MB:LFO_F）

form	波形
sin	サイン波
sin_u	調整サイン波
saw	ノコギリ波
saw_u	調整ノコギリ波
saw_os	ノコギリ波ワンショット
tri	三角波
tri_u	調整三角波
pls	パルス波
pls_u	調整パルス波
nzw	ホワイトノイズ
nzw_u	調整ホワイトノイズ
table	ユーザー定義テーブル
nlbend	ノンリニアベンド

form=pls* のとき：

追加で次の項目を設定します。
　　pwm=[],　　（任意：デューティ比）
pwmの設定範囲は 0.05 ～ 0.995 です。
pwmを省略した場合の初期設定は「pwm=0.5」（デューティー比=50%）です。

form=table のとき：

追加で次の項目を設定します。
　　tbl_atk=[],　　（必須：#MB:LFOTBL_ATKで定義したテーブル番号）
　　tbl_rel=[],　　（任意：#MB:LFOTBL_RELで定義したテーブル番号）
tbl_relを省略した場合、常に「tbl_rel=-1」と解釈され、tbl_relは未使用扱いになります。

tbl_atkの設定範囲は 0 ～ 4095 です。
tbl_relの設定範囲は 0 ～ 4095 です。
ただし、いずれも、未定義のテーブル番号を指定した場合、ＬＦＯ変化が無効になります（変化のないダミーテーブルが参照されます）。

form=nlbend のとき：
追加で次の項目を設定します。
　　bandrate=[],　（必須：ノンリニアベンドの変化率）
bendrateの設定範囲は 1.0 ～ 512.0 です。
1未満もしくは512超えの指定は、それぞれ1、512に制限されます。

8.11. @ly[1]：yコマンドＬＦＯ

8.12. #MB:LFO_Y：yコマンドＬＦＯ定義

【記述例】

```
#MB:CONFIG {
   env_clock: unit=sec: rate=1/60,
   env_resol: unit=smp,
   lfo_clock: unit=sec: rate=1/60,
   lfo_resol: unit=sec: rate=1/60,
}
#MB:VOLUME_SCALE vs=1 {
   type=linear, v=15, vr=0, vl=15, vlr=0, vh=15,
}
#MB:ENV_A @ea=1 { peak=15, init=0 |
   n:0:15, n:1:15, n:0:14, n:1:14, n:0:13, n:1:13,
   n:0:12, n:1:12, n:0:12, n:1:12, n:0:11, n:1:11,
   n:0:11, n:1:11, n:0:10,
   r:0:2, n:32:2, n:0:0
}
#MB:LFOTBL_ATK 1 { loop=2, cmpl=0, offset=0, denom=1, width_mode=step, |
   0.25, 0.25, 0.5,
}
#MB:LFOTBL_REL 9 { loop=0, cmpl=0, offset=0, denom=1, width_mode=step, |
   0.125,
}
#MB:LFO_Y @ly=1 {
   depth=1, width=1, delay=0, yform=pls, yfunc=pwm, tbl_atk=1, tbl_rel=9
}
t120 l8 vs1 v15 @@"pls" @ea1 @ly1 o4
 q8,8 f16a16 q3 >d<fadfa>d<
   q8 e16a16 q3 >c<ea>c<ea>c<;
```

このように書くと、#MB:LFO_Y と @lyコマンドによって、yコマンドＬＦＯの定義と適用を記述できます。
この記述例では、パルス音源のデューティ比に対して、次の時間的変化を与えています。

　　ノートオンから2/60秒だけDT比25%、以降DT比50%。
　　ノートオフからはDT比12.5%

以上

【解説】
yコマンドＬＦＯコマンド（@ly[1]）：
yコマンドＬＦＯ定義（#MB:LFO_Y）の定義番号を整数で指定します。
定義番号は 0 ～ 1023 の整数です。
未定義の番号を指定するとエラーになります。
トラック先頭における初期設定では、yコマンドＬＦＯは停止状態です。

yコマンドＬＦＯ定義（#MB:LFO_Y）：
yコマンドＬＦＯコマンド（@ly[1]）で使用する、yコマンドＬＦＯ設定を定義します。
yコマンドＬＦＯ定義では、揺らす波形はユーザー定義テーブル方式に固定されていて、それに従ったyコマンドのコマンド発行シーケンスが行われます。

yコマンドＬＦＯ定義方法の概要は、
・行頭からの記述で、キーワード「#MB:LFO_Y」を書く。
・続けて、スペースを置き、「@ly=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られた設定データを書く。
です。

定義番号：
定義番号は、@lyコマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 1023 です。

設定データ：

【記述例】

```
#MB:LFO_Y @ly=1 {
   depth=1, width=1, delay=0, yform=pls, yfunc=pwm, tbl_atk=1, tbl_rel=9
}
```

【解説】
フォーマットは次の通りです。

```
#MB:LFO_Y @ly=定義番号 {
     depth=[], width=[], delay=[], yform=[], yfunc=[],
     tbl_atk=[], tbl_rel=[],
}
```

depth：
depth=[]では、揺らす値としてユーザー定義テーブルから得られる値への倍率を数値で指定します。
ゼロを指定した場合、ＬＦＯが無効になります。

width：
width=[]では、時間単位としてＬＦＯステップ数を数値で指定します。
（揺らすパターンがユーザー定義テーブル方式に固定されているため）
数値は１以上の値で指定してください。
ＬＦＯステップ数は、ＬＦＯ解像度設定（@lorコマンドなど）で決まる、ＬＦＯ専用の最少時間単位です。
【注】
widthの動作モードは、ユーザー定義テーブル内の設定で、ＬＦＯステップ数かＬＦＯシフト数かを選択することができます（初期設定はＬＦＯステップ数）。
詳細は「#MB:LFOTBL_ATK」を参照してください。

delay：
delay=[]では、ノートオンからＬＦＯ開始までの遅延時間を数値で指定します。
時間単位は width 同様です。

delay が 0 以上のとき：
ノートオンの都度、ＬＦＯシーケンスが再スタートします。
（ノートオン同期モード）

delay が負数のとき：
ノートオンでＬＦＯシーケンスが再スタートしない、ノートオン非同期モードとなります。いかなる負数の値も、このモード指定と判定されます。
ノートオン非同期モードにおいて、任意のタイミングでＬＦＯ位相のリセットを行いたい場合は、ＬＦＯの再スタートコマンドを使用します。
ＬＦＯコマンドの指定時には、同期・非同期モードに関わらずＬＦＯシーケンスは位相ゼロからスタートします。

【注意】
delay によりyコマンドＬＦＯが遅延されている間は、ＬＦＯによるyコマンドが発行されないため、ノートオン直前まで発行されていたＬＦＯによるyコマンドが残ったままの形になり、想定外のシーケンスになる可能性があります。
このような問題を避けるには、delay を 0 とし、ＬＦＯテーブル内のデータ定義で、遅延に相当するデータを含むデータ列にします。

yform：
yform=[]では、yコマンドの宛先となる、音源モジュール名を、文字列で指定します。（音源モジュール名一覧）

yfunc：
yfunc=[]では、yコマンドの宛先となる、音源モジュールに対応した機能名を、文字列で指定します。
機能名の文字列は、各音源モジュールの説明内の、
　　【yコマンドにおける、音源モジュールへの設定】
の項の内容を参照してください。

tbl_atk：
yコマンドＬＦＯ定義では、揺らすパターンはユーザー定義テーブル方式に固定されているので、テーブル番号を指定する必要があります。
tbl_atk=[]では、#MB:LFOTBL_ATKで定義したテーブル番号を指定します。
#MB:LFOTBL_ATKは、ノートオンの際にシーケンスされるテーブル定義です。
tbl_atkの設定範囲は 0 ～ 4095 です。

tbl_rel：
yコマンドＬＦＯ定義では、揺らすパターンはユーザー定義テーブル方式に固定されているので、テーブル番号を指定する必要があります。
tbl_rel=[]では、#MB:LFOTBL_RELで定義したテーブル番号を指定します。
#MB:LFOTBL_RELは、ノートオフの際にシーケンスされるテーブル定義です。
tbl_relの設定範囲は 0 ～ 4095 または -1 です。
tbl_rel=-1 の場合、tbl_relは未使用扱いとなり、ノートオフの際には、参照されるテーブルに変更がかからず、ノートオンで参照されているテーブルが継続して参照されて yコマンドＬＦＯが掛かり続けます。

8.13. @l*z：ＬＦＯの停止

【記述例】

```
@lpz
```

【解説】

指定のＬＦＯシーケンサを停止状態にします。

@lpz	ピッチＬＦＯを停止状態にします。
@laz	音量ＬＦＯを停止状態にします。
@lbz	パンポットバランスＬＦＯを停止状態にします。
@lfz	フィルタＬＦＯを停止状態にします。
@lyz	yコマンドＬＦＯを停止状態にします。

8.14. @l*r：ＬＦＯの再スタート

【記述例】

```
@lpr
```

【解説】

指定のＬＦＯシーケンスを再スタートさせます。
スラー途中でＬＦＯを再スタートさせる場合や、ＬＦＯのノートオン非同期モードで、任意のタイミングで再スタートさせる場合などの使用を想定しています。

@lpr	ピッチＬＦＯを再スタートさせます。
@lar	音量ＬＦＯを再スタートさせます。
@lbr	パンポットバランスＬＦＯを再スタートさせます。
@lfr	フィルタＬＦＯを再スタートさせます。
@lyr	yコマンドＬＦＯを再スタートさせます。

8.15. @l*t[1],[2]：ＬＦＯテーブル番号変更

【記述例】

```
@lpt1,20
```

ピッチＬＦＯで使用中の、ノートオン同期用テーブル番号を 20 に変更します。

```
@lat1,10 @lat2,101
```

音量ＬＦＯで使用中の、ノートオン同期用テーブル番号を 10 に変更し、
ノートオフ同期用テーブル番号を 101 に変更します。

【解説】

指定のＬＦＯにおいて使用中の、ユーザー定義テーブル番号を変更します。

@LPT[1],[2]	ピッチＬＦＯで使用中の、ユーザー定義テーブル番号を変更します。
@LAT[1],[2]	音量ＬＦＯで使用中の、ユーザー定義テーブル番号を変更します。
@LBT[1],[2]	パンポットバランスＬＦＯで使用中の、ユーザー定義テーブル番号を変更します。
@LFT[1],[2]	フィルタＬＦＯで使用中の、ユーザー定義テーブル番号を変更します。
@LYT[1],[2]	yコマンドＬＦＯで使用中の、ユーザー定義テーブル番号を変更します。

指定する引数は次の通りです。
引数[1]...ユーザー定義テーブルの種別
引数[2]...ユーザー定義テーブルの番号
引数はカンマで区切って指定します。

引数[1]

変更したいユーザー定義テーブルの種別を、1 または 2 で指定します。
1 の場合：ノートオン時に使用するユーザー定義テーブル
2 の場合：ノートオフ時に使用するユーザー定義テーブル

引数[2]

使用するユーザー定義テーブル番号を指定します。
ここで指定する番号は、
　　引数[1] が 1 の場合、「#MB:LFOTBL_ATK」で定義済みの番号
　　引数[1] が 2 の場合、「#MB:LFOTBL_REL」で定義済みの番号
を指定してください。
未定義番号を指定した場合、0 を指定したものとみなされます（0番は未定義でも内部保持のダミーデータが参照されます）。

引数[1] が 2 の場合は特別に、テーブル番号に -1 を指定することができます。-1 を指定すると、ノートオフ同期でのテーブル番号変更機能が無効になります。（引数[1] が 1 の時の負数指定は 0 とみなされます）

8.16. #MB:LFOTBL_ATK：ＬＦＯテーブル定義(ATK)

【記述例】

```
#MB:CONFIG {
   lfo_clock: unit=tick: rate=1,
   lfo_resol: unit=sec: rate=1/600,
}
#MB:LFOTBL_ATK 1 {
   loop=0, cmpl=0, offset=0, denom=1, width_mode=step, |
   0, 1, 0, -1,
}
#MB:LFOTBL_ATK 2 {
   loop=0, cmpl=1, offset=0, denom=1, width_mode=step, |
   0, 1, 0, -1,
}
#MB:LFO_P @lp=1 { depth=400, width=36, delay=18, form=table, tbl_atk=1, }
#MB:LFO_P @lp=2 { depth=400, width=36, delay=18, form=table, tbl_atk=2, }
#MB:ENV_A @ea=1 { peak=15, init=& | n:0:15, n:360:0, r:1:0 }

t120 q16,16 o5 @@"pls" v15 l2 @ea1
@lp1 ceg @lp2 ceg;
```

@lp1ではギザギザのピッチ変化、@lp2では滑らかなピッチ変化を掛けています。

【解説】

ＬＦＯの変化パターン、ユーザー定義テーブルモードで使用する、テーブルデータを定義します。
「#MB:LFOTBL_ATK」は、ノートオン同期用に使用されます。
また、ユーザー定義テーブルのデータ定義領域は、全てのＬＦＯで共有して利用するため、例えばピッチＬＦＯ用に定義した内容であっても、音量ＬＦＯなど他のＬＦＯからでも利用できてしまう点に注意して下さい。

テーブル定義方法の概要は、
・行頭からの記述で、キーワード「#MB:LFOTBL_ATK」を書く。
・続けて、スペースを置き、「定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られた設定データを書く。
です。

定義番号：

定義番号は、ＬＦＯデータ定義中のテーブル番号と合致するように定義します。
定義番号は、「#MB:LFO_P」「#MB:LFO_A」「#MB:LFO_B」「#MB:LFO_F」
「#MB:LFO_Y」の定義内における、tbl_atk=[] 項目に設定する番号に相当します。
定義番号の設定範囲は 0 ～ 4095 です。
未定義のテーブル番号を参照した場合は、0 番が読み出されます。

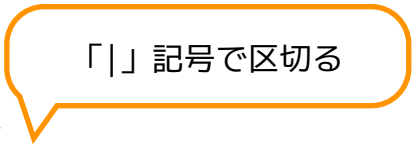
設定データ：

【記述例】

```
#MB:LFOTBL_ATK 1 {
   loop=0, cmpl=0, offset=0, denom=1, width_mode=step, |
   0, 1, 0, -1,
}
```

【解説】

フォーマットは次の通りです。




```
#MB:LFOTBL_ATK 定義番号 {
     loop=[], cmpl=[], offset=[], denom=[],
     width_mode=[], shift_reso=[],          |
     [],[],[]...,
}
```

設定データ（#MB:LFOTBL_ATKの中身）は、「|」記号１つで区切られ、前半をヘッダ部、後半をボディ部と呼びます。

「|」記号を省略した場合は、ボディ部のみの定義とみなされ、ヘッダ部で指定されるはずだった内容は全て初期値のままとして扱われます。

ヘッダ部は、loop,cmpl,offset,denom,width_mode,shift_resoの６項目からなり、いずれも省略可能です。省略した場合は、初期値のままとして扱われます。

ボディ部は、１個以上（最大2048個）のテーブル値をカンマ区切りで記述します。

loop：

loop=[]では、ループエントリインデックスを指定します。
テーブルシーケンスをループさせる場合は、ループエントリインデックスを整数（0 以上 2047以下）で指定します。
ループさせない場合は、-1 と指定します。
ループさせない場合、テーブル末尾に到達後は、テーブル末尾の値が維持されます。

cmpl：

cmpl=[]では、テーブル値を読み出すときの補間モードを、0 または 1 で指定します。

0 の場合：

０次補間モードです。
０次補間モードでは、読み出し位置が少数以下切捨ての場所にあるテーブル値がそのまま使われるため、ガクガクした形になります。
このモードは、急峻な変化の表現に向いています。このモードで滑らかに変化させるには多くのテーブル値を要します。

1 の場合：

１次補間モードです。
１次補完モードでは、読み出し位置（少数以下含む）に応じて、前後のテーブル値から線形補完した値が使われるため、少ないテーブル値でも滑らかに変化させられます。

offset：

offset=[]では、テーブル値への加算値（少数以下も指定可）を設定します。
テーブル値の読み出しが、テーブル値に offset を加算した値で行われます。

denom：

denom=[]では、テーブル値への分母を設定します。
0 より大きい数値（少数以下も指定可）を指定してください。
テーブル値の読み出しが、テーブル値を denom で除算した値で行われます。
計算の優先順位は、オフセット加算が優先です。

実際の値 = （テーブル値 + offset）÷ denom

となります。

width_mode：
width_mode=[]では、ＬＦＯの width で受け取る時間幅のタイプを、「step」または「shift」の文字列で指定します。

step の場合：
width をステップ数として扱います。
テーブルシーケンスはＬＦＯ解像度(*)を１ステップとして動作しますが、
「何ステップ経過後にテーブルインデックスを１進めるか」
を決定する動作モードとなります。
この場合、ＬＦＯでの width 受付範囲は 1 以上になります。少数以下の指定も受け付けます。

shift の場合：
width をシフト数として扱います。
テーブルシーケンスはＬＦＯ解像度(*)を１ステップとして動作しますが、
「１ステップ経過後に、テーブルインデックスをどれだけ進めるか」
を決定するモードとなります。
この場合、ＬＦＯでの width 受付範囲は、0 より大きく、shift_reso 以下になります。（width は shift_reso で除算され、0 より大きく 1 以下の範囲で利用）

(*)ＬＦＯ解像度：
「#MB:CONFIG { lfo_resol:... }」または「@lor[str],[1]」による設定

shift_reso：
shift_reso=[]では、「width_mode=shift」を指定した場合のみ有効な、シフト数に対する分母を設定します（シフト数の解像度）。小数以下の指定も受け付けます。
0 より大きい数値を指定してください。

ボディ部のテーブル値：
ボディ部のテーブル値では、1 個以上（最大2048個まで）のテーブル値をカンマ区切りで指定します。テーブル値は少数以下の指定も受け付けます。
最大個数を超える定義を行うとエラーとなり、テーブル自体が定義されません。

8.17. #MB:LFOTBL_REL：ＬＦＯテーブル定義(REL)

【記述例】

```
#MB:CONFIG {
   env_clock: unit=sec: rate=1/60,
   env_resol: unit=smp,
   lfo_clock: unit=sec: rate=1/60,
   lfo_resol: unit=sec: rate=1/60,
}
#MB:VOLUME_SCALE vs=1 {
   type=linear, v=15, vr=0, vl=15, vlr=0, vh=15,
}
#MB:ENV_A @ea=1 { peak=15, init=0 | n:0:15, r:0:7, n:32:7, n:0:0 }
#MB:LFOTBL_ATK 1 {
   loop=2, cmpl=0, offset=0, denom=1, width_mode=step, |
   0.5, 0.5, 0.25,
}
#MB:LFOTBL_REL 9 {
   loop=0, cmpl=0, offset=0, denom=1, width_mode=step, |
   0.125,
}
#MB:LFO_Y @ly=1 {
   depth=1, width=1, delay=0, yform=pls, yfunc=pwm, tbl_atk=1, tbl_rel=9
}

t120 q8,16
@@"pls" @ea1 @ly1
l2 o5 cegefdggcegefdg&g;
```

yコマンドＬＦＯにて、パルス波のデューティ比を時間変化させています。
ノートオン中は、最初の1/30[秒]がデューティ比 50% で、それ以後 25%。
ゲート切れやノートオフになったらデューティ比を 12.5% にしています。

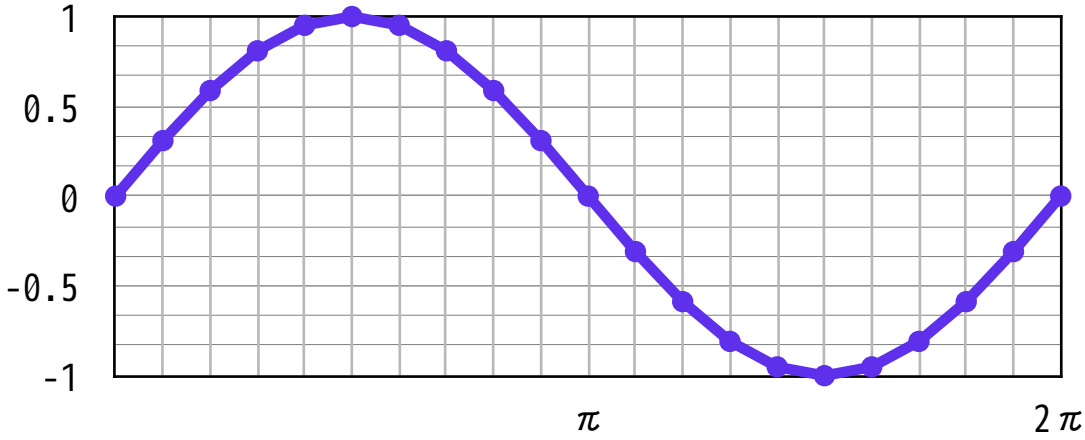
【解説】

ＬＦＯデータ定義の、tbl_rel=[]項目で使用する、テーブルデータを定義します。
「#MB:LFOTBL_REL」は、ノートオフ同期専用に使用されます。
定義内容と、定義できる数は、「#MB:LFOTBL_ATK」と同様です。

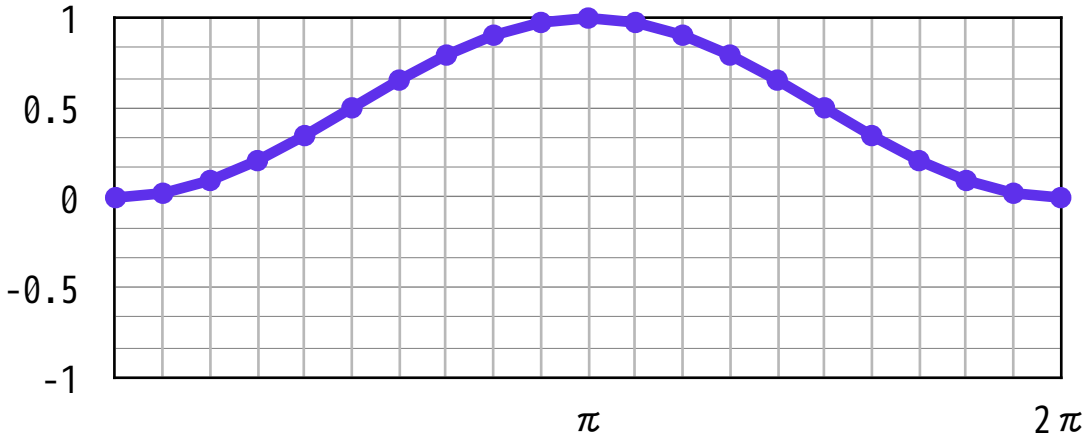
8.18. ＬＦＯ波形グラフ（１）（@lp/@lb/@lf）

ＬＦＯ波形グラフ（１）：@lp/@lb/@lfの変化パターン

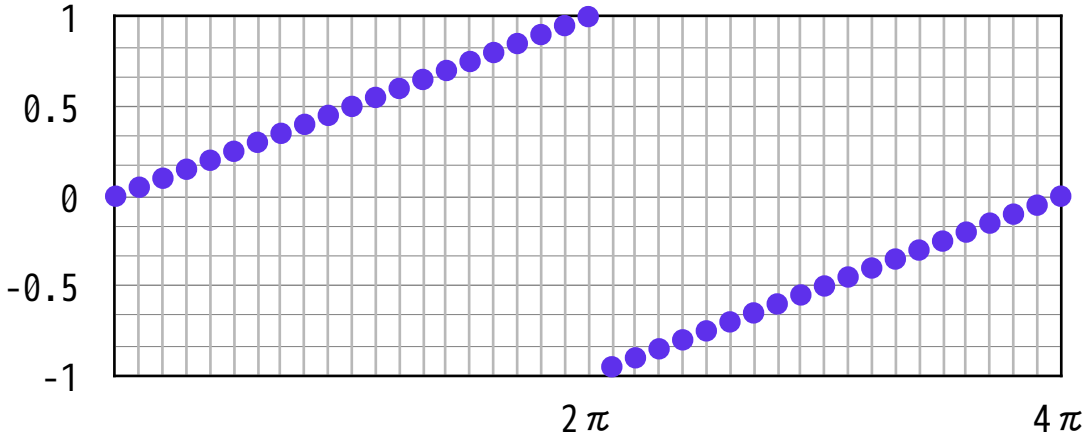
サイン波：form=sin



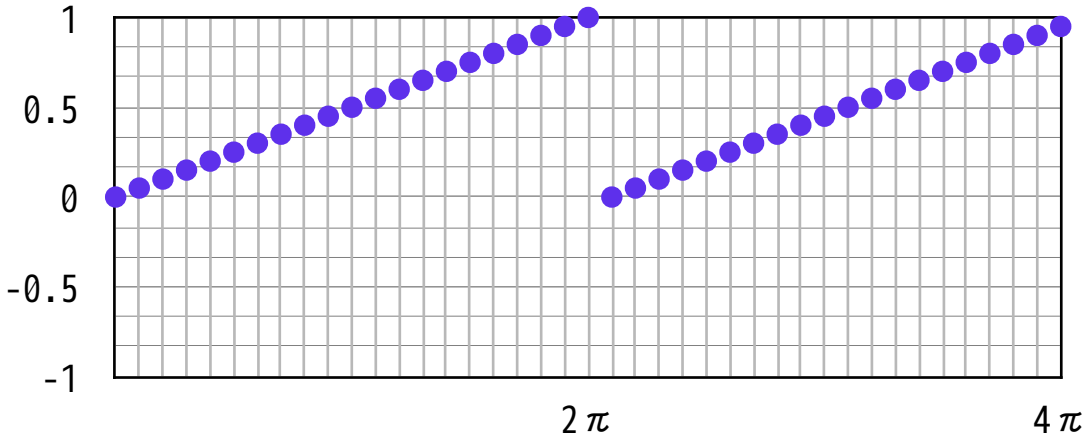
調整サイン波：form=sin_u



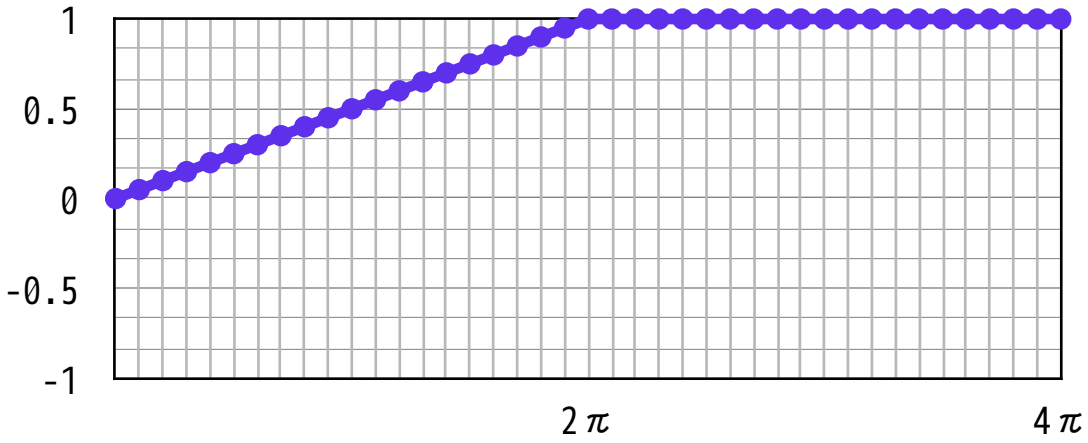
ノコギリ波：form=saw



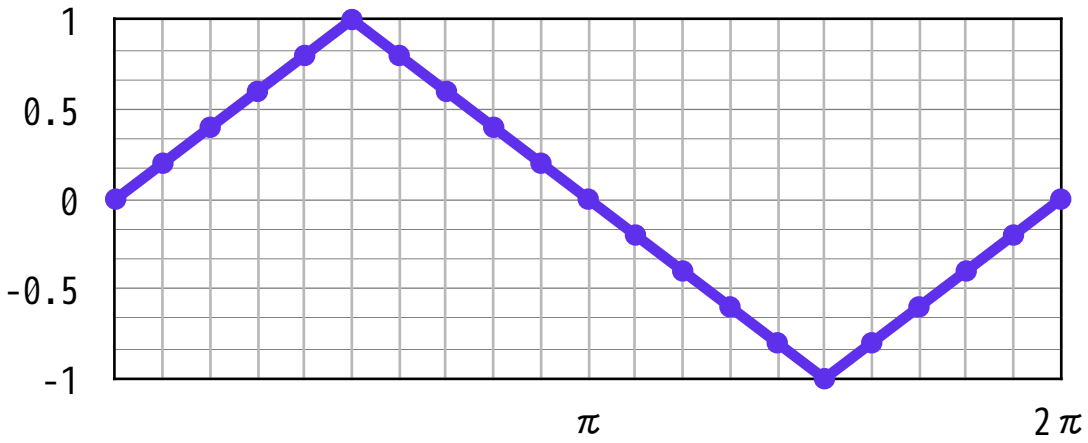
調整ノコギリ波：form=saw_u



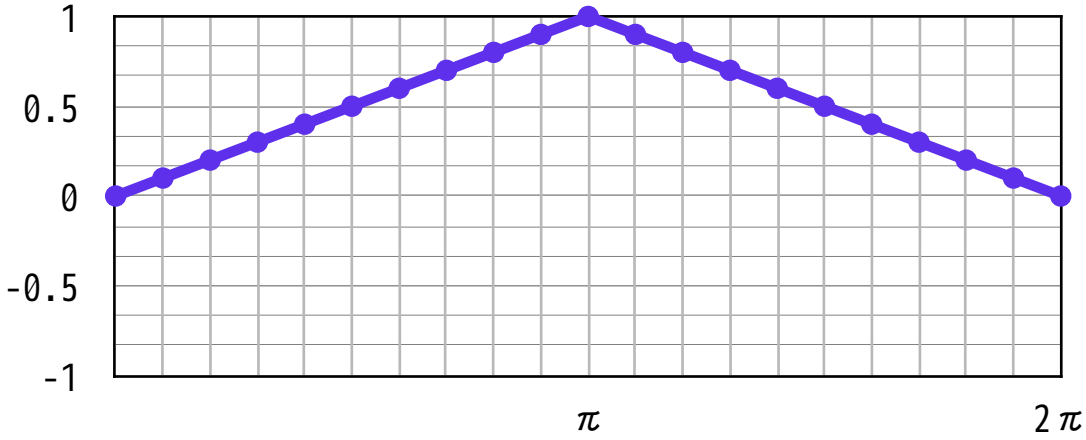
ノコギリ波ワンショット：form=saw_os



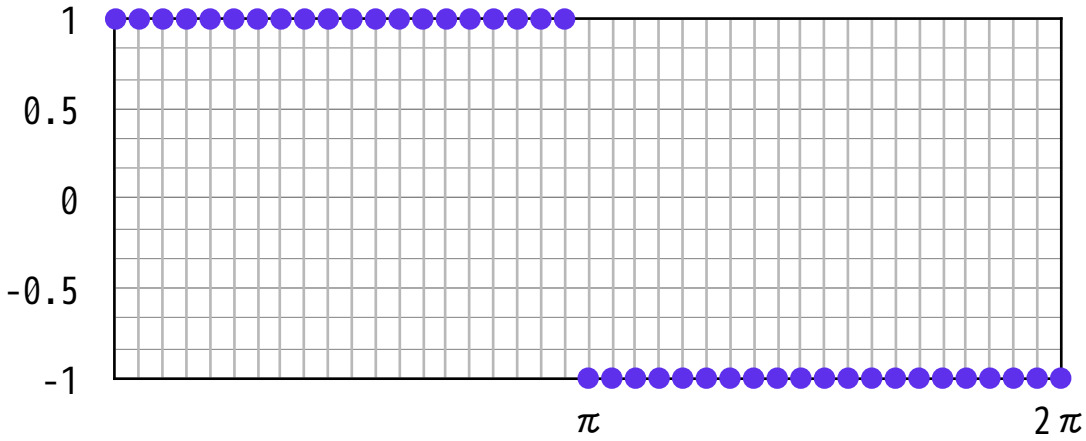
三角波：form=tri



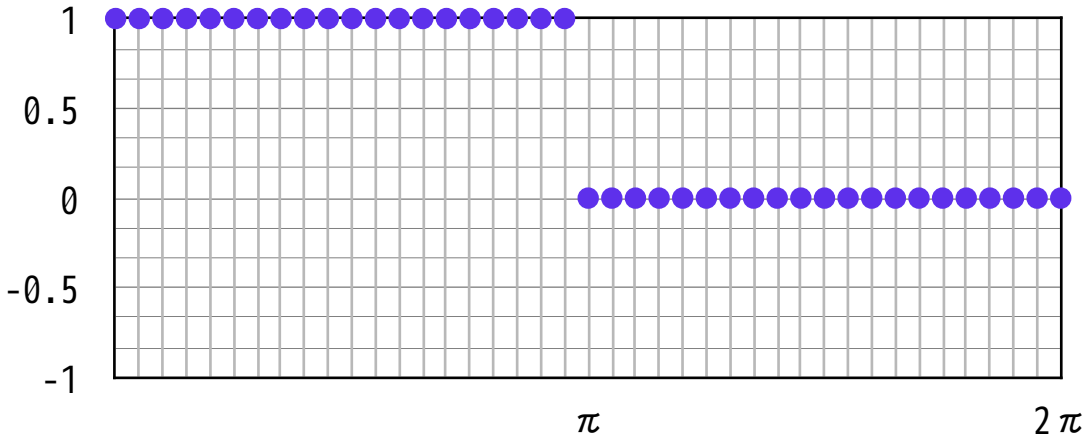
調整三角波：form=tri_u



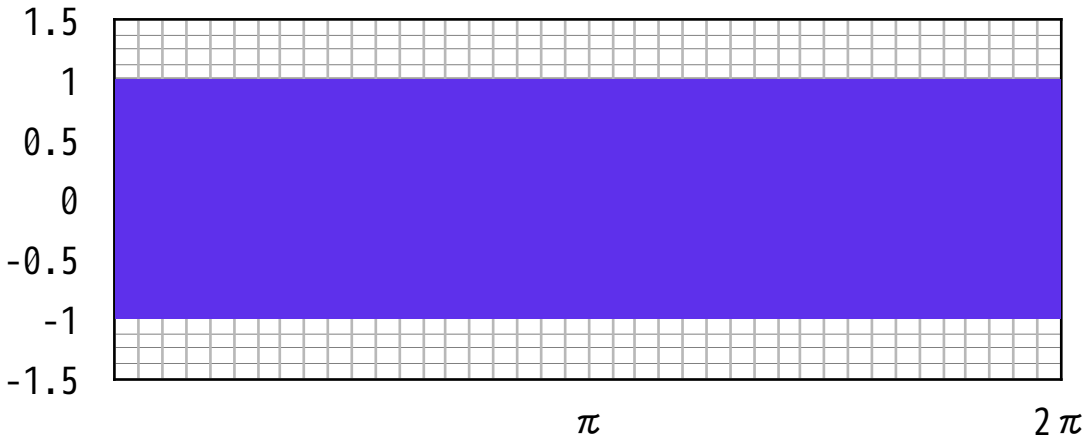
パルス波：form=pls



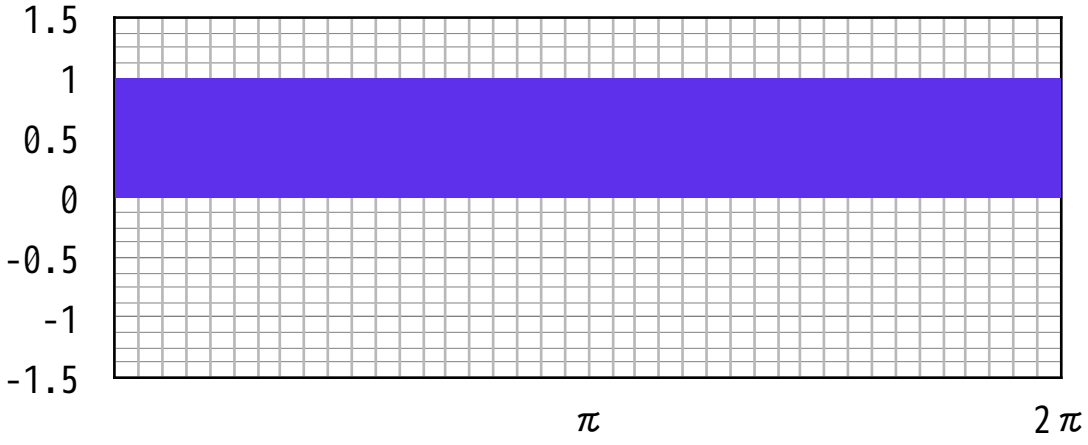
調整パルス波：form=pls_u



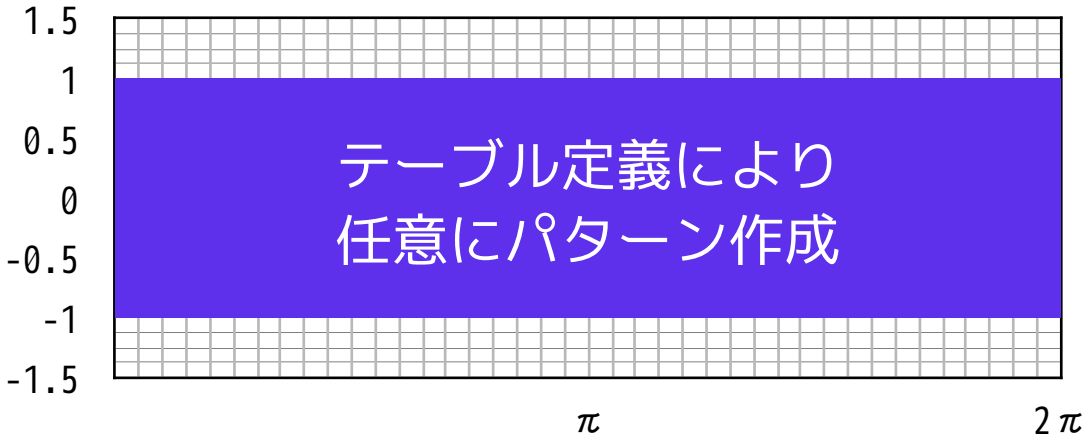
ホワイトノイズ：form=nzw



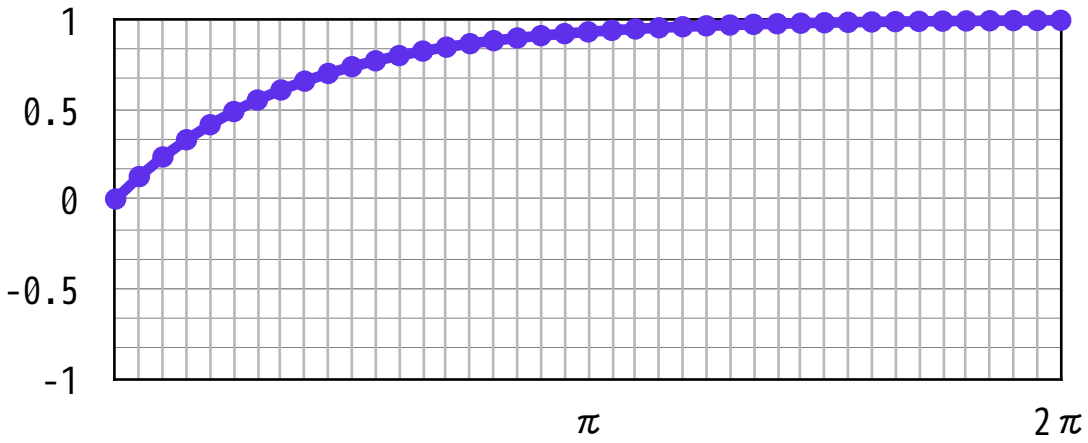
ホワイトノイズＢ：form=nzw_u



ユーザー定義テーブル：form=table



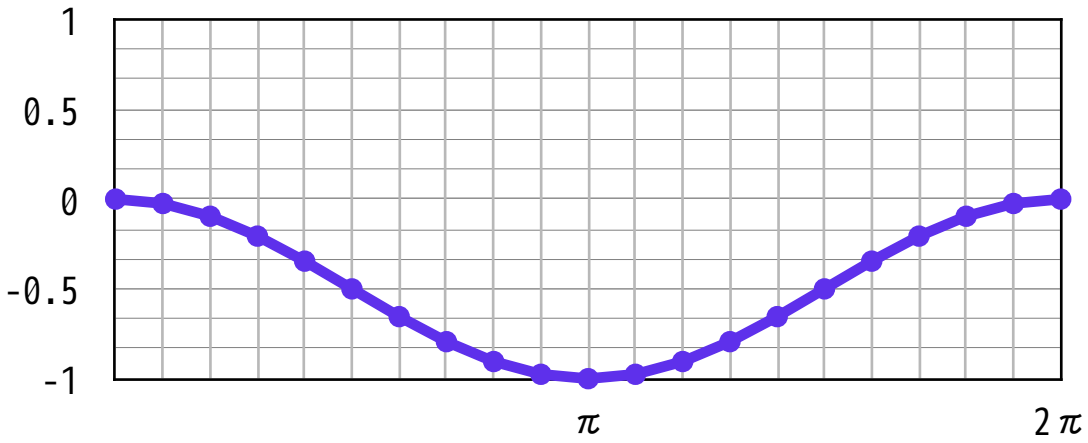
ノンリニアベンド：form=nlbend



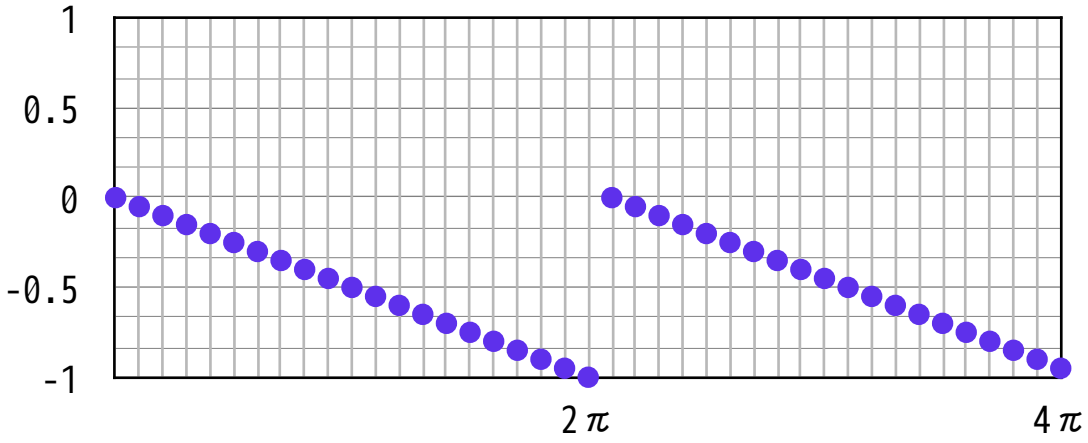
8.19. ＬＦＯ波形グラフ（２）（@la）

ＬＦＯ波形グラフ（２）：@laの変化パターン

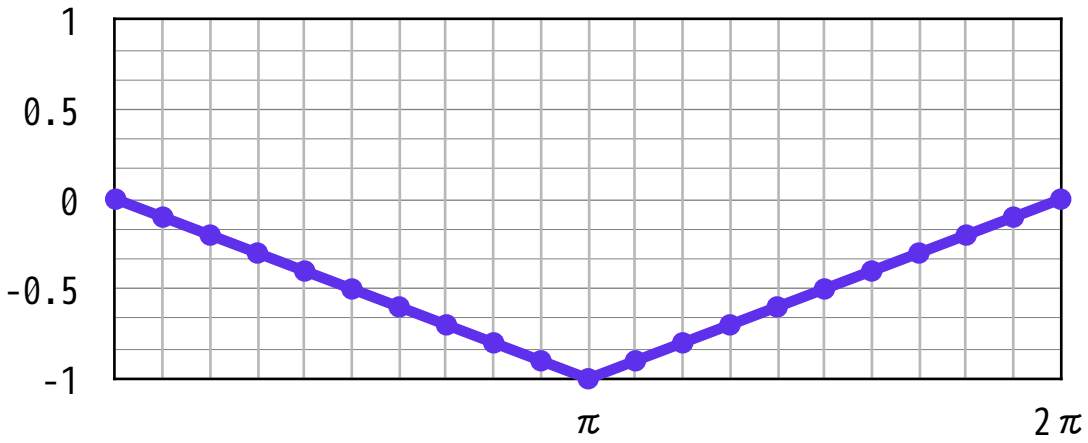
調整サイン波：form=sin_a



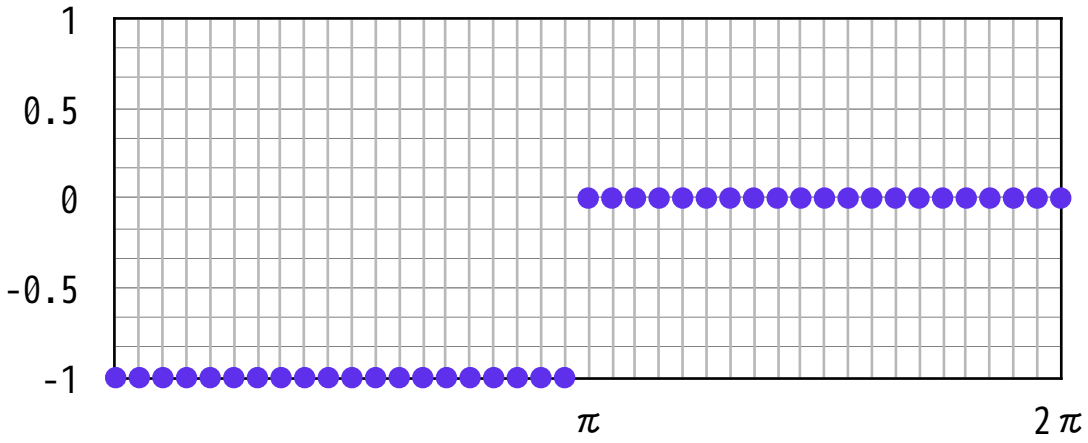
調整ノコギリ波：form=saw_a



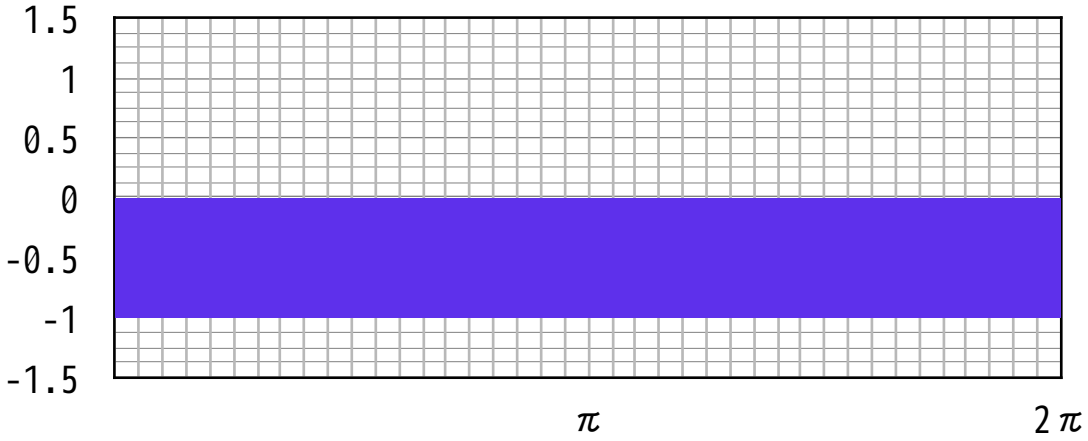
調整三角波：form=tri_a



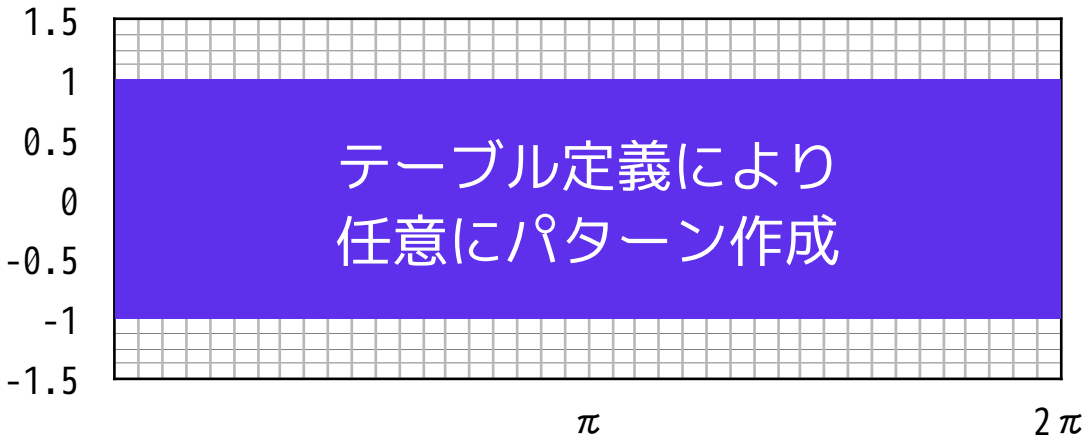
調整パルス波：form=pls_a



調整ホワイトノイズ：form=nzw_a



ユーザー定義テーブル：form=table



9.発音数関連

9.1. @pl[1],[2]：ポリフォニックモード設定

【記述例】

```
#MB:ENV_A @ea=1 { peak=15, init=& | n:1:15, r:250:0 }
@@"pls" @ea1 o4 l8
@pl4,0 dfa ceg brrr rrrr
@pl1,0 dfa ceg brrr;
```

@pl4,0の掛かった音符は、リリース音が重なりながら演奏されます。(4-poly)
@pl1,0の掛かった音符は、リリース音が重ならずに演奏されます。(mono)

【解説】
１トラックの記述で音が複数重なる「ポリフォニックモード」の設定を行います。

指定する引数は次の通りです。
引数[1]...最大発音数
引数[2]...音が重なる方式のモード番号
引数はカンマで区切って指定します。

引数[1]には、同時発音数を整数で記述します。
設定範囲は、1 ～ 64 です。
同時発音数が多すぎると処理落ちの原因になります。

引数[2]には、音が重なる方式のモード番号を、0 または 1 で指定します。

0 のとき：
ポリフォニックモードを上塗り方式にします。
まず、無音チャネルを検索し、見つかればそこで発音させます。
無音チャネルが見つからなかった場合、最も古い発音に、最も新しい発音を上塗りします。
最も古い発音が同時発音により複数存在する場合は、最初に見つかったものに機械的に上塗りされます。

1 のとき：
(1)ノートオフされて余韻状態で発音しているチャネルを検索し、その中から、これからノートオンしようとしている音程と同じ音程のものがあれば、新しい発音を上塗りします。
(2)(1)で見つからなければ、無音チャネルを検索し、見つかればそこで発音させます。
(3)(2)においても見つからなければ、最も古い発音に、最も新しい発音を上塗りします。
最も古い発音が同時発音により複数存在する場合は、最初に見つかったものに機械的に上塗りされます。
（このモードは、鍵盤楽器を意識したモードです）

【備考】
ポリフォニックモードが発動されたトラックでは、最大発音数の分だけ、内部的に発音チャネルを固定的に確保するので、メモリ使用量が増えます。
ポリフォニックモードが発動されてないトラックでは、多重チャネル化を行わないので、メモリ使用量は増えません。

9.2. [...]：和音記法

【解説】
和音記法です。１つのトラックで和音を記述することが出来ます。

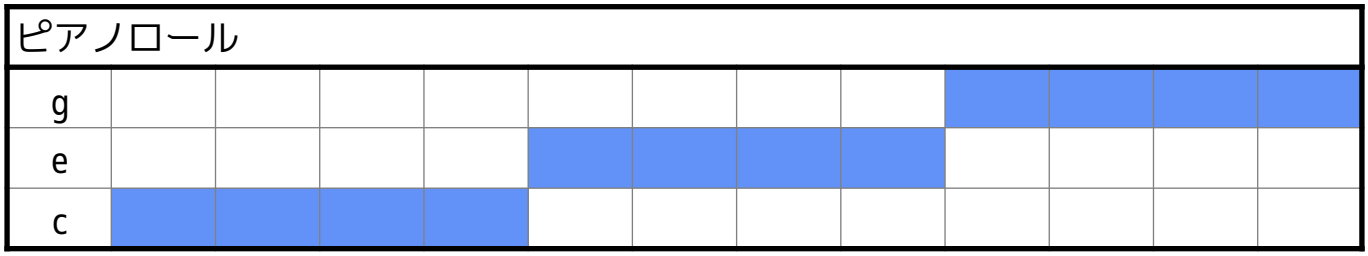
ポリフォニックモードが発動しているトラックで、音符を大カッコ［ ］で括ると、複数音を同時に鳴らすことが出来ます。
休符（rコマンド）によって、大カッコ内での発音タイミングを、ずらすことも出来ます。

ポリフォニックモードを発動するには、対象となるトラックで、@plコマンドを記述します。

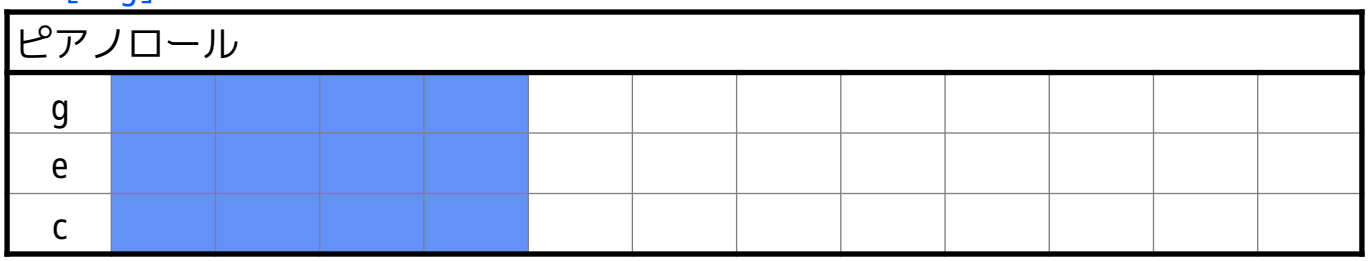
【機能制限】
・和音記法［ ］内では、スラーは使用できません。

【記述例】

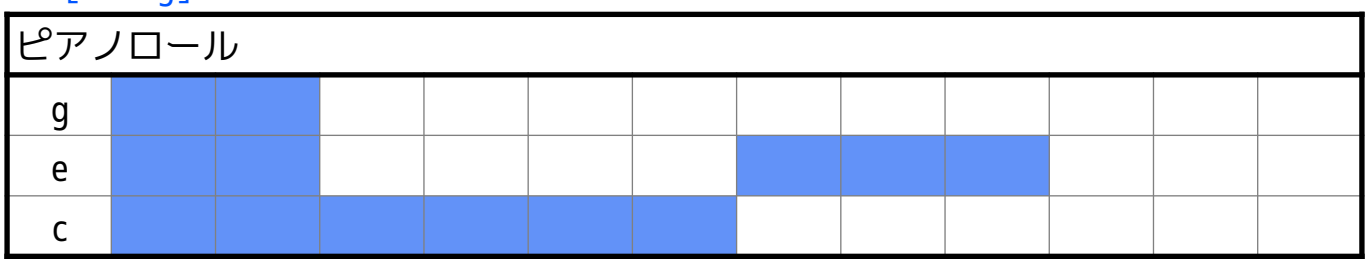
```
l4 ceg
```



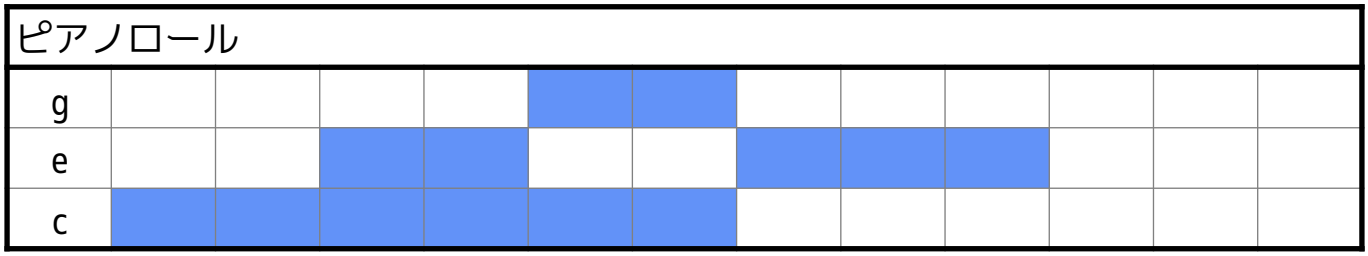

```
l4 [ceg]
```



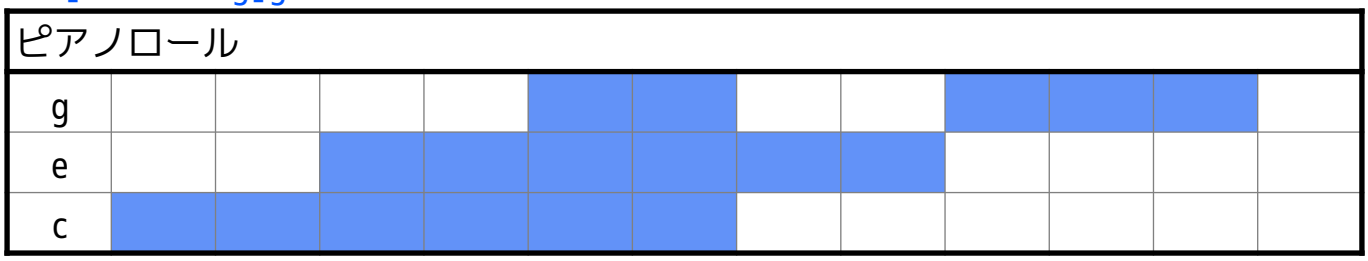

```
l8 [c4.eg]e.
```



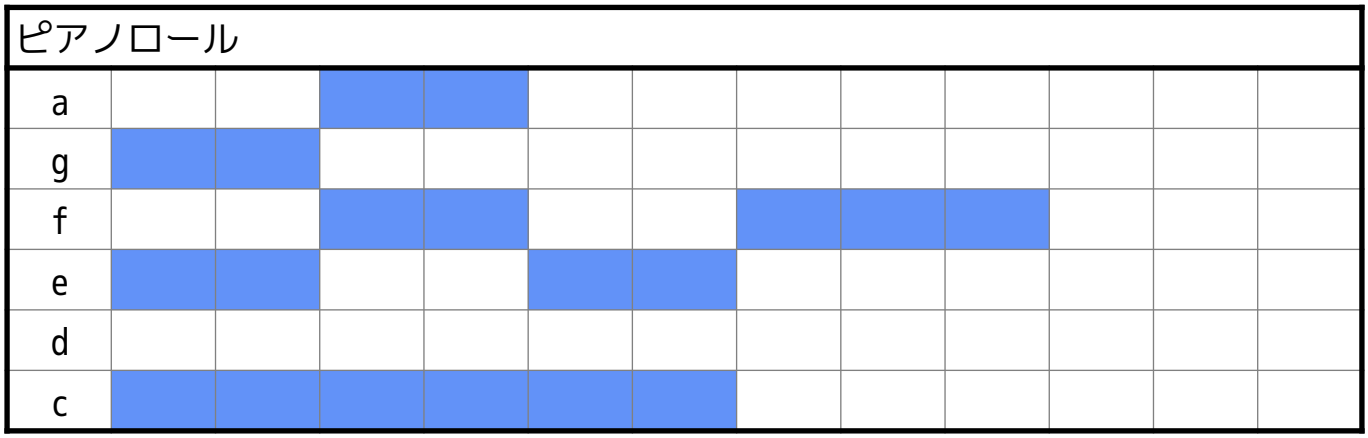
```
l8 [c4.rerg]e.
```




```
l8 [c4.re4.rg]g.
```




```
l8 [c4.egrfare]f.
```



10.装飾関連

10.1. xa[str],[1]：パラメータへの加算

【記述例】

```
$A=l8 @d0 cegefdgg
    @d300 cegefdg&g
;

$A;
xa"@d",8 $A;
```

１つのマクロ定義でトラックを２つ定義し、２トラック目全体のデチューン設定に＋８加算しています。

【解説】

宛先になるコマンドのパラメータ値に、指定値を加算します。
適用範囲は、当コマンド指定以降、当該トラック終了までです。

想定している用途は、既に書き上げたＭＭＬトラックの文字列を、マクロ参照でそのまま再利用することです。再利用時、トラック先頭に当コマンドを書き足して、デチューン値やサブフォーム番号などをずらすのに使用します。

指定する引数は次の通りです。
引数[str]...宛先
引数[1].....加算数値
引数はカンマで区切って指定します。
トラック先頭における初期設定は、全ての宛先に対し加算値０です。

引数[str]

加算対象となる宛先のコマンドを、文字列で指定します。

【xaコマンドの宛先となる識別子一覧】

宛先	内容
@	音源サブフォーム番号
mv	ミキシングボリューム
v	ボリューム
vl	ボリューム for @la
ns	ノートシフト
nt	ノートトランス
@d	デチューン
@ea	音量エンベロープ
@ef	フィルタエンベロープ
@lp	ピッチＬＦＯ
@la	音量ＬＦＯ
@lb	パンポットバランスＬＦＯ
@lf	フィルタＬＦＯ
@ly	yコマンドＬＦＯ
@fd	フィルタ（動的/dynamic）
@fs	フィルタ（静的/static）
@dl	ディレイ
yp	yコマンドパック

引数[1]

加算したい数値を指定します。
負数や、小数以下の指定も記述することもできます。
ただし、加算後のパラメータが有効な設定値におさまるようにしてください。
加算後のパラメータが無効な設定値の場合、エラーは xaコマンド ではなく、宛先のコマンドに対して表示されます。
整数のみ受け付けるコマンドでは、加算値の少数以下は切り捨てが行われます。

10.2. xm[str],[1]：パラメータへの乗算

【記述例】

```
$A= l8 @p100 cegefdgg
      @p-100 cegefdg&g
;

xm"@p",0  o5 $A;
xm"@p",-1 o3 $A;
```

１つのマクロ定義でトラックを２つ定義し、２トラック目全体のパンポット設定に(-1)を乗算し、左右反転しています。

【解説】

宛先になるコマンドのパラメータ値に、指定値を乗算します。
適用範囲は、当コマンド指定以降、当該トラック終了までです。

想定している用途は、既に書き上げたＭＭＬトラックの文字列を、マクロ参照でそのまま再利用することです。再利用時、トラック先頭に当コマンドを書き足して、デチューン値やサブフォーム番号などをずらすのに使用します。

指定する引数は次の通りです。
引数[str]...宛先
引数[1].....乗算数値
引数はカンマで区切って指定します。
トラック先頭における初期設定では、全ての宛先に対し乗算しません。

引数[str]

乗算対象となる宛先のコマンドを、文字列で指定します。

【xmコマンドの宛先となる識別子一覧】

宛先	内容
@p	パンポット
p	パンポット・レガシーモード

引数[1]

乗算したい数値を指定します。
負数や、小数以下の指定も記述することもできます。
ただし、乗算後のパラメータが有効な設定値におさまるようにしてください。
乗算後のパラメータが無効な設定値の場合、エラーは xmコマンド ではなく、宛先のコマンドに対して表示されます。
整数のみ受け付けるコマンドでは、乗算値の少数以下は切り捨てが行われます。

10.3. @kt[1]：キーイベント（トリガモード）

10.4. #MB:KEYEVENT_TRG：@kt用キーイベント定義

【記述例】

```
#MB:KEYEVENT_TRG @kt=1 {
   target=@p, start=0, loop=-1 |
   -60,-40,-20,0,20,40,60,60,
   -60,-40,-20,0,20,40,60,
}
t120 l8 @@"pls" @kt1
o4 cegefdgg cegefdg&g
cegefdgg cegefdg&g;
```

このように書くと、#MB:KEYEVENT_TRG と @ktコマンドによって、キーイベント（トリガモード）定義と適用を記述できます。
この場合、音符ごとに@pコマンドが、数値テーブル「-60,-40,-20...」の先頭から順に読み出されて発動されます。テーブルの最後に達したらループはせず、最後の状態が維持されます。

【解説】

キーイベント（トリガモード）コマンド（@kt[1]）：
キーイベント（トリガモード）定義（#MB:KEYEVENT_TRG）の定義番号を整数で指定します。
定義番号は 0 ～ 1023 の整数です。
未定義の番号を指定するとエラーになります。
トラック先頭における初期設定では、当機能は停止状態です。

キーイベント（トリガモード）とは、ノートオンの都度、事前に定義した「#MB:KEYEVENT_TRG」によるテーブルに従って順番に、指定種別のコマンドを発行する機能です。異なる宛先であれば、複数の発動も可能です。

【キーイベントの発動をやめたい場合】
発動済みのキーイベント機能の停止するには、
宛先ごとのコマンドを、あらためて発行します。

例えば、音源サブフォーム番号宛のキーイベント機能の発動と停止であれば、次のようになります。（#MB:KEYEVENT_TRG @kt=1 にて、target=@ の場合）

```
@kt1 cdefg @1 gfedc;
```

この場合、cdefg はキーイベント機能が有効、gfedc は宛先「@」のキーイベント機能が全て無効になって、@1 で演奏されます。

キーイベント（トリガモード）定義（#MB:KEYEVENT_TRG）：

キーイベント（トリガモード）コマンド（@kt[1]）で使用する、キーイベント（トリガモード）設定を定義します。

キーイベント（トリガモード）定義方法の概要は、
・行頭からの記述で、キーワード「#MB:KEYEVENT_TRG」を書く。
・続けて、スペースを置き、「@kt=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られた設定データを書く。
です。

定義番号：

定義番号は、@ktコマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 1023 です。

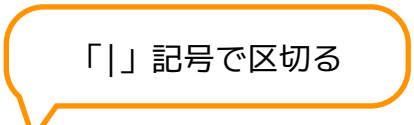
設定データ：

【記述例】

```
#MB:KEYEVENT_TRG @kt=1 {
   target=@p, start=0, loop=-1 |
   -60,-40,-20,0,20,40,60,60,
   -60,-40,-20,0,20,40,60,
}
```

【解説】

フォーマットは次の通りです。




```
#MB:KEYEVENT_TRG @kt=定義番号 {
     target=[], start=[], loop=[], |
     [],[],[]...
}
```

target：

target=[]では、キーイベント機能発動の宛先コマンドを文字列で指定します。

【#MB:KEYEVENT_TRG での target=[] の宛先一覧】

宛先	内容
@	音源サブフォーム番号
v	ボリューム
vl	ボリューム for @la
@p	パンポット
p	パンポット・レガシーモード
@n	ノイズサイクル
@w	パルス幅（デューティ比）
@d	デチューン
@ea	音量エンベロープ
@ef	フィルタエンベロープ
@lp	ピッチＬＦＯ
@la	音量ＬＦＯ
@lb	パンポットバランスＬＦＯ
@lf	フィルタＬＦＯ
@ly	yコマンドＬＦＯ
@fd	フィルタ（動的/dynamic）
@fs	フィルタ（静的/static）
@dl	ディレイ
yp	yコマンドパック

start：

start=[]では、「|」で区切られた後の数値テーブルの、何番目から参照開始するか、整数で指定します。
設定範囲は、0 ～（定義個数-1）です。

loop：

loop=[]では、「|」で区切られた後の数値テーブルの、何番目からループするか、整数で指定します。
設定範囲は、0 ～（定義個数-1）または -1 です。
-1 の場合、ループせず、テーブル終端に達したら、終端位置が維持されます。

「|」で区切られた後の数値テーブル：

数値テーブルでは、target=[]の宛先コマンドに送る数値を、カンマ区切りで記述します。
数値は少数以下の指定も受け付けますが、宛先コマンドが整数で受け付けるコマンドだった場合、少数以下は使用時に四捨五入で整数に丸められて処理されます。
数値テーブルの定義個数は 1 ～ 2048個の範囲になります。

10.5. @kc[1]：キーイベント（音程モード）

10.6. #MB:KEYEVENT_CODE：@kc用キーイベント定義

【記述例】

```
#MB:KEYEVENT_CODE @kc=1 {
   target=@p, |
   -75,-74,-72,-71,-70,-69,-67,-66,-65,-64,-62,-61,
   -60,-59,-57,-56,-55,-54,-52,-51,-50,-49,-47,-46,
   -45,-43,-42,-41,-40,-38,-37,-36,-35,-33,-32,-31,
   -30,-28,-27,-26,-25,-23,-22,-21,-20,-18,-17,-16,
   -14,-13,-12,-11, -9, -8, -7, -6, -4, -3, -2, -1,
     1,  2,  3,  4,  6,  7,  8,  9, 11, 12, 13, 14,
    16, 17, 18, 20, 21, 22, 23, 25, 26, 27, 28, 30,
    31, 32, 33, 35, 36, 37, 38, 40, 41, 42, 43, 45,
    46, 47, 49, 50, 51, 52, 54, 55, 56, 57, 59, 60,
    61, 62, 64, 65, 66, 67, 69, 70, 71, 72, 74, 75,
}
t120 l4 @@"pls" @kc1 o1 a>a>a>a>a>a>a>a;
```

このように書くと、#MB:KEYEVENT_CODE と @kcコマンドによって、キーイベント（音程モード）定義と適用を記述できます。
この場合、音程番号(0～119)ごとに決められた@pコマンドが発動されます。

【解説】

キーイベント（音程モード）コマンド（@kc[1]）：
キーイベント（音程モード）定義（#MB:KEYEVENT_CODE）の定義番号を整数で指定します。
定義番号は 0 ～ 1023 の整数です。
未定義の番号を指定するとエラーになります。
トラック先頭における初期設定では、当機能は停止状態です。

キーイベント（音程モード）とは、ノートオンの都度、事前に定義した「#MB:KEYEVENT_CODE」によるテーブルに従って、音程番号(0～119)ごとに決められた、指定種別のコマンドを発行する機能です。異なる宛先であれば、複数の発動も可能です。

【キーイベントの発動をやめたい場合】
発動済みのキーイベント機能の停止するには、
宛先ごとのコマンドを、あらためて発行します。

例えば、音源サブフォーム番号宛のキーイベント機能の発動と停止であれば、次のようになります。（#MB:KEYEVENT_CODE @kc=1 にて、target=@ の場合）

```
@kc1 cdefg @1 gfedc;
```

この場合、cdefg はキーイベント機能が有効、gfedc は宛先「@」のキーイベント機能が全て無効になって、@1 で演奏されます。

キーイベント（音程モード）定義（#MB:KEYEVENT_CODE）：
キーイベント（音程モード）コマンド（@kc[1]）で使用する、キーイベント（音程モード）設定を定義します。

キーイベント（音程モード）定義方法の概要は、
・行頭からの記述で、キーワード「#MB:KEYEVENT_CODE」を書く。
・続けて、スペースを置き、「@kc=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られた設定データを書く。
です。

定義番号：
定義番号は、@kcコマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 1023 です。

設定データ：

【記述例】

```
#MB:KEYEVENT_CODE @kc=1 {
   target=@p, |
   -75,-74,-72,-71,-70,-69,-67,-66,-65,-64,-62,-61,
   -60,-59,-57,-56,-55,-54,-52,-51,-50,-49,-47,-46,
   -45,-43,-42,-41,-40,-38,-37,-36,-35,-33,-32,-31,
   -30,-28,-27,-26,-25,-23,-22,-21,-20,-18,-17,-16,
   -14,-13,-12,-11, -9, -8, -7, -6, -4, -3, -2, -1,
     1,  2,  3,  4,  6,  7,  8,  9, 11, 12, 13, 14,
    16, 17, 18, 20, 21, 22, 23, 25, 26, 27, 28, 30,
    31, 32, 33, 35, 36, 37, 38, 40, 41, 42, 43, 45,
    46, 47, 49, 50, 51, 52, 54, 55, 56, 57, 59, 60,
    61, 62, 64, 65, 66, 67, 69, 70, 71, 72, 74, 75,
}
```

【解説】

フォーマットは次の通りです。

「|」記号で区切る

```
#MB:KEYEVENT_CODE @kt=定義番号 {
     target=[],                          |
     [],[],[]...      （音程番号0～119に対する120個のコマンド指定値）
}
```

※target=[]の後のカンマは無くても良いですが、将来拡張されて複数並ぶ際には必要になります。

target：

target=[]では、キーイベント機能発動の宛先コマンドを文字列で指定します。

【#MB:KEYEVENT_CODE での target=[] の宛先一覧】

宛先	内容
@	音源サブフォーム番号
v	ボリューム
vl	ボリューム for @la
@p	パンポット
p	パンポット・レガシーモード
@n	ノイズサイクル

宛先	内容
@w	パルス幅（デューティ比）
@d	デチューン
@ea	音量エンベロープ
@ef	フィルタエンベロープ
@lp	ピッチＬＦＯ
@la	音量ＬＦＯ
@lb	パンポットバランスＬＦＯ
@lf	フィルタＬＦＯ
@ly	yコマンドＬＦＯ
@fd	フィルタ（動的/dynamic）
@fs	フィルタ（静的/static）
@dl	ディレイ
yp	yコマンドパック

「|」で区切られた後の数値テーブル：

数値テーブルでは、target=[]の宛先コマンドに送る数値を、
カンマ区切りで１２０個記述します。（定義個数は必ず１２０個）
数値は少数以下の指定も受け付けますが、宛先コマンドが整数で受け付けるコマンドだった場合、少数以下は使用時に四捨五入で整数に丸められて処理されます。

１２０個の各数値の割り当ては、１個目がオクターブ０の「ド」が呼ばれた際に使用される値となり、２個目がオクターブ０の「ド＃」用の値になります。以降同様に半音ずつ上がっていき、１２０個目はオクターブ９の「シ」用の値になります。

１２平均律のピッチスケールなどで、１２０個の範囲外の音程が呼ばれた場合、例えばオクターブ０の「ド」よりも低い音程が呼ばれた場合は、オクターブ０の「ド」用の値で処理されます。また、オクターブ９の「シ」よりも高い音程が呼ばれた場合は、オクターブ９の「シ」用の値で処理されます。

10.7. @kl[1]：キーイベント（音長モード）

10.8. #MB:KEYEVENT_LEN：@kl用キーイベント定義

【記述例】

```
#MB:CONFIG {
   tempo_unit: note_ticks=192: beat_length=4,
}
#MB:KEYEVENT_LEN @kl=1 {
   target=@p, |
   case=48: set=7,   //l4
   case=36: set=75,  //l8.
   case=12: set=-68, //l16
   default: set=0,
}
t120 l16 @@"pls" @kl1 o4
c8.<g >c8.<g >c<g>c eg4
f8.d   f8.d  f d <b>d<g4;
```

このように書くと、#MB:KEYEVENT_LEN と @klコマンドによって、キーイベント（音長モード）定義と適用を記述できます。
この場合、次のように音長に応じて@pコマンドが発動されます。

４分音符→@p7
付点８分音符→@p75
１６分音符→@p-68
上記以外→@p0

【解説】

キーイベント（音長モード）コマンド（@kl[1]）：
キーイベント（音長モード）定義（#MB:KEYEVENT_LEN）の定義番号を整数で指定します。
定義番号は 0 ～ 1023 の整数です。
未定義の番号を指定するとエラーになります。
トラック先頭における初期設定では、当機能は停止状態です。

キーイベント（音長モード）とは、ノートオンの都度、

事前に定義した「#MB:KEYEVENT_LEN」によるテーブルに従って、
音長に応じた値で指定種別のコマンドを発行する機能です。
異なる宛先であれば、複数の発動も可能です。

【キーイベントの発動をやめたい場合】
発動済みのキーイベント機能の停止するには、
宛先ごとのコマンドを、あらためて発行します。

例えば、音源サブフォーム番号宛のキーイベント機能の発動と停止であれば、次のようになります。（#MB:KEYEVENT_LEN @kl=1 にて、target=@ の場合）

```
@kl1 cdefg @1 gfedc;
```

この場合、cdefg はキーイベント機能が有効、gfedc は宛先「@」のキーイベント機能が全て無効になって、@1 で演奏されます。

キーイベント（音長モード）定義（#MB:KEYEVENT_LEN）：
キーイベント（音長モード）コマンド（@kl[1]）で使用する、キーイベント（音長モード）設定を定義します。

キーイベント（音長モード）定義方法の概要は、
・行頭からの記述で、キーワード「#MB:KEYEVENT_LEN」を書く。
・続けて、スペースを置き、「@kl=定義番号」を書く。
・続けて、スペースを置き、中括弧「{}」で括られた設定データを書く。
です。

定義番号：
定義番号は、@klコマンドの引数で使用する番号と合致するように定義します。
定義番号の設定範囲は 0 ～ 1023 です。

設定データ：

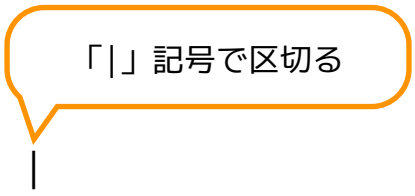
【記述例】

```
#MB:KEYEVENT_LEN @kl=1 {
   target=@p, |
   case=48: set=7,   //l4
   case=36: set=75,  //l8.
   case=12: set=-68, //l16
   default: set=0,
}
```

上記のtickカウント（48で４分音符など）になるのは、
#MB:CONFIG { tempo_unit: note_ticks=192: beat_length=4, }
この設定の場合になります。

【解説】

フォーマットは次の通りです。




```
#MB:KEYEVENT_LEN @kl=定義番号 {
      target=[],                    |
      case=[]: set=[],          （音長tickカウントに応じた設定値）
      case=[]: set=[],
       :
      default: set=[],
}
```

target：
target=[]では、キーイベント機能発動の宛先コマンドを文字列で指定します。

【#MB:KEYEVENT_LEN での target=[] の宛先一覧】

宛先	内容
@	音源サブフォーム番号
v	ボリューム
vl	ボリューム for @la
@p	パンポット
p	パンポット・レガシーモード
@n	ノイズサイクル
@w	パルス幅（デューティ比）
@d	デチューン
@ea	音量エンベロープ
@ef	フィルタエンベロープ
@lp	ピッチＬＦＯ
@la	音量ＬＦＯ
@lb	パンポットバランスＬＦＯ
@lf	フィルタＬＦＯ
@ly	yコマンドＬＦＯ

宛先	内容
@fd	フィルタ（動的/dynamic）
@fs	フィルタ（静的/static）
@dl	ディレイ
yp	yコマンドパック

「|」で区切られた後の定義：

「|」で区切られた後の定義では、「:」で区切られた２つの定義、

```
case=[]: set=[],
```

を１組とする設定を、カンマ区切りで記述します。
ただし、カンマ区切記述の最後は

```
default: set=[],
```

という１組の記述とします。
これらの「:」で区切られた定義は、1 ～ 1024個の範囲で定義します。

`case=[]: set=[],`：

case=[]では、tickカウントがいくつの音長だったときかを設定します。
set=[]では、case=[]に該当する場合に、target宛にセットするコマンド値を設定します。

`default: set=[],`：

defaultは、どのcase=[]にも該当しなかった場合を指す記述です。
defaultに対応した set=[]では、どのcase=[]にも該当しなかった場合にtarget宛にセットするコマンド値を設定します。

11.付加情報関連

11.1. #ML:TITLE：曲のタイトル

11.2. #ML:ARTIST：曲のアーティスト

11.3. #ML:COMMENT：曲のコメント

11.4. #ML:CODING：ＭＭＬ作成者名

【記述例】

```
#ML:TITLE    曲のタイトル
#ML:ARTIST   曲のアーティスト
#ML:COMMENT 曲のコメント
#ML:CODING   ＭＭＬ作成者名
```

【解説】

各種、曲のメタデータ文字列を定義します。
フォーマットは次の通りです。

```
#ML:TITLE 定義文字列
#ML:ARTIST 定義文字列
#ML:COMMENT 定義文字列
#ML:CODING 定義文字列
```

定義文字列には、任意の文字列（全角文字も可）を記述します。
改行で定義終了になります。

定義文字列の直前には、区切りとしてスペースが最低１文字は必要ですが、複数のスペースを置いても良く、置いたスペースは全て読み飛ばされます。
結果的に、定義文字列の先頭文字は、必ずスペース以外の文字になります。

「区切りとしてのスペース」とは、半角スペースまたはTAB文字になります。
全角スペースは通常文字扱いとなり、「区切りとしてのスペース」としては認識されません。

複数行定義したい場合には、例えば曲のタイトルであれば「#ML:TITLE」定義を複数、記述します。
ただし、原則として表示保証対象は最初の行のみになります。

[end of file]