Z-MUSIC システムver.2.0

Oh!X編集部編

X68000用音楽ドライバZMUSIC.X ver.2.0
X-BASIC用外部関数/C言語用ライブラリ
AD PCMツール ZPCNV.X/ZPLK.R
Oh!X標準/拡張サンプリングデータ
綴じ込み付録　5"2HD6枚組

SOFT BANK

X68000
PERSONAL WORKSTATION

Z-MUSICシステムver.2.0

CONTENTS

1 目次
2 Z-MUSICシステム導入の手引き
6 MEASURE1 イントロダクション
9 MEASURE2 ZMUSIC.Xのオプションスイッチ
12 MEASURE3 X-BASIC用外部関数MUSICZ.FNC
24 MEASURE4 ZMSコマンド
30 MEASURE5 MML
43 MEASURE6 AD PCMを扱う
47 MEASURE7 プレイヤーZP.R
50 MEASURE8 AD PCM加工ツールZVT.X
54 MEASURE9 MIDIデータを扱う
57 MEASURE10 Z-MUSICのファンクションコール
60 MEASURE11 エラーについて
72 MEASURE12 ZMD/ZPDフォーマット
77 MEASURE13 ワークエリア
83 MEASURE14 効果音モードと映像同期モード
91 MEASURE15 PCM8モード
92 MEASURE16 付録
94 用語解説

装　　丁　島村勝頼
表紙CG制作　江口響子
協　　力　進藤慶到
マニュアルデバッグ協力
NOVA, ぼぜ, M.ODAKI, LastTear, ゆきごん, あほみ他

Z-MUSICシステム導入の手引き

Z-MUSICとはなにか?

あなたのシステムディスクを立ち上げてください。付録ディスク1をBドライブに入れます。

B>¥ZMUSIC¥ZMUSIC

コマンドラインからこのように打ち込むとZ-MUSICシステムはあなたのシステムに登録されます。

あとは音楽データ(ZMS形式で記述されたファイルやZMD形式のデータ)を用意すれば,なにかの音楽をずっと流しながら作業をしたり,作成したゲームのBGMや効果音を鳴らしたり,そのほかMIDI楽器の音色やシーケンスデータなどを保存するなど,X68000を使っていて音楽に関わりのありそうなことはすべてこのドライバで制御できます。X68000の標準的なOPMファイルやX-BASICでの音楽命令書式さえわかっていれば,FM音源,AD PCM音源,MIDI音源のすべてを簡単に扱うことができます。

とりあえずインストール(ハードディスクの場合)

ドライバ本体のZMUSIC.Xと各種サポートツール,PCMデータを各自の環境にインストールします。

まず,ハードディスクの空きを確認します。フルセットでZ-MUSICシステムをインストールすると以下の28個のディレクトリを作成することになります。

ZMUSIC	198Kバイト	*
BASS	70Kバイト	*
SNARE	256Kバイト	*
TOMTOM	221Kバイト	*
ETHNIC	100Kバイト	*
ACCENT	64Kバイト	*
CYMBAL	177Kバイト	*
ADDITION	153Kバイト	*
RAP	95Kバイト	
EFFECTS	306Kバイト	
OHCH	1220Kバイト	
EXTRA_PERC	520Kバイト	*
BRASS	80Kバイト	
SYNTHE	69Kバイト	
GUITAR	209Kバイト	
STRINGS	68Kバイト	
CHOIR	79Kバイト	
PIANO	189Kバイト	
TR808SET	36Kバイト	*
BRASS_P16	179Kバイト	
DG_P16	291Kバイト	
PIANO_16	310Kバイト	
SAX_P16	245Kバイト	
CHOIR_P16	282Kバイト	
ORGAN_P16	184Kバイト	
STRINGS_P16	186Kバイト	
EBASS	216Kバイト	
SOUND_EFFECTS	1099Kバイト	

前半の2～11番までのディレクトリの内容はZ-MUSICシステムver.1.0のものと同じです。合計で約7Mバイトを占有します。普通に使っていく分には"*"印のついたものだけでことが足りると思われますので,それらのディレクトリだけをインストールしておき,足りないものがあったら追加する方式でもよいでしょう。この場合なら約2Mバイトの占有になります。

これらはそのディレクトリ単位であれば,これらはどのドライブに置いてもかまいません。以下ではすべてCドライブに置くものとして説明します。ファイルの転送は,

```
COPYALL ADDITION C:
COPYALL BASE C:
COPYALL EXTRA_PERC C:
        ⋮
```

のようにCOPYALLコマンドを使用してディレクトリ単位で行ってください。また,一部のファイルはLHAで圧縮されていますので,

```
LHA
```

のようにしてファイルを展開しておいてください(とりあえず使わない場合は不要です)。

なお,LHA.Xは今回のディスクには収録されておりません。Oh! Xの付録ディスクなどに収録されていますので,それを使用してください。

●ZMUSIC.Xとツールにパスを通す

実行ファイルで必要なものはすべてディレクトリZMUSICにまとめてあります。このようにツールなどは基本的に1カ所にまとめてパスを通しておけばそれでかまいません。AUTOEXEC.BAT中のパス指定を,

```
PATH=A:¥BIN;A:¥BASIC2 ……;C:¥ZMUSIC
```

のようにしておきます。

以下のような手順で転送するとよいでしょう。

```
MD C:ZMUSIC
COPYALL ZMUSIC C:ZMUSIC
```

続いてサンプリングデータをセットします。ハードディスクの残り容量次第ですが,最低限,

ディスク2すべて
ディスク4のEXTRA_PERC
(できればディスク1のADDITION,RAPも)

の各ディレクトリの内容をハードディスクに転送してください。場所はどこでもかまいません。

転送が終わったら,環境変数zmusicを設定するようにシステムのAUTOEXEC.BATを書き換えます。先ほど転送したディレクトリの場所を環境変数zmusicに列記します。指定の順番などは適当でかまいませんが,ファイルの検索はここに書かれている順番に行われますので,普段よく使うものを前のほうに書いておいたほうがいいでしょう。たとえば,

```
SET zmusic=D:¥EXTRA_PERC;D:¥SNARE;C:¥BASS;
C:¥TOMTOM;B:¥ETHNIC;…… B:¥ADDITION;D:¥ORCH
```

のような具合です。

また,

```
C:¥ZMUSIC¥ADPCM_DATA¥SNARE
C:¥ZMUSIC¥ADPCM_DATA¥BASS
C:¥ZMUSIC¥ADPCM_DATA¥TOMTOM
C:¥ZMUSIC¥ADPCM_DATA¥……
        ⋮
```

のような階層構造になったディレクトリにデータを格納したいというときは,環境変数が長くなりすぎてシステムに登録できないことがあります。そのような場合は,

```
SET zmusic=/0C:¥ZMUSIC;/1C:¥ZMUSIC¥P16
SET zmusic0=SNARE;BASS;TOMTOM;ETHNIC;ADDITION
```

のように補助的にzmusic0などといった環境変数を使用してパスの記述を簡略化できます。この場合は,zmusic0,zmusic1という環境変数で指定された内容が,環境変数zmusicでその番号に対応した指定パスのディレクトリ内からさらにパスを加えてサーチするようになります。

Z-MUSICのセットアップ(フロッピーディスクの場合)

Z-MUSICシステムはフロッピーディスクベースでも開発が行えるように工夫してありますが,サンプリング音がかなり大量にありますのでできるだけハードディスクの導入をおすすめします。

フロッピーベースで使用する際の注意点を注意点をまとめてみましょう。

まず,Z-MUSIC用のシステムディスクを用意します。これは普段使っているものでもかまわないのですが,システムディスクに十分な空き容量がない人は付録ディスクのディスク1をベースにすることをおすすめします。

ディスク1から必要のなさそうなファイルをどんどん削除してく

ださい。ただし，Z-MUSIC関係のツールだけは残しておきます。整理すればZMUSICディレクトリも90Kバイトくらいになるはずです。次に普段使う基本的なコマンドを転送してください（ED.XやCOPYALL.Xなど）。続いてAUTOEXEC.BATからDSHELL3.Xの行を削除します。

サンプリングデータのうち特に使用頻度の多いものはディスク2にまとめてありますから，ディスク2はそのまま使用してください。ディスク4のEXERA_PERCをシステムディスクに転送します。余裕があればTR808SETもシステムディスクに入れてしまいましょう。

```
COPYALL B:EXTRA_PERC A:
COPYALL B:TR808SET A:
```

RAPやADDITIONは余裕がなければ削ってしまいます。基本的にはハードディスクでの最小構成をなんとかフロッピーディスク上に構築できればいいわけです。

そのほかのものは特殊な用途に使うものがほとんどです。楽器音や効果音はめったに使うことはありませんから，当座のところはこれで大丈夫でしょう。

環境変数の設定は以下のようになります。

```
SET　zmusic=B:¥BASS;B:¥SNARE;B:¥TOMTOM;B:¥SYMBAL;B:¥ETHNIC;B:¥ACCENT;B:¥EFFECT;A:¥EXTRA_PERC;A:¥TR808SET
```

このようにCOMMAND.Xでのパス設定と同様にパス名を連続して記述できます。

ミュージックデータによってはここにある音以外のものも使用しているかもしれません。ここで足りなかったものはしかたありませんからほかのディスクからシステムディスクに拾ってきます。それほど使用頻度は高くないはずですので使い終わったら削除しておきましょう。

●ピアノの音しか鳴らないけど……

Z-MUSICシステムは基本的にOPMDRVコンパチですが，省メモリ化のため，OPMDRVが内蔵している64音色はオプションになっています。起動時にはすべての音色がピアノになっています。OPMDRV用に作成された曲データで内蔵音色を使用しているものを演奏するときには，ETCディレクトリにあるX68K_SND.ZMSをあらかじめ演奏しておいてください。起動時に自動的に読み込むようにするためには，

```
ZMUSIC-SA:X68K_SND.ZMS
```

のようにドライバ組み込み時のセットアップファイルとして指定してやります。いちいち指定するのが面倒だという場合には，AUTOEXEC.BAT内で環境変数を次のように設定しておくとよいでしょう。

```
SET zm_opt=-SA:X68K_SND.ZMS
```

これでZMUSIC.X起動時に自動的にオプションスイッチが設定された状態で立ち上がります（ただしオプション指定がまったくされてない場合のみ）。なお，CONFIG.SYSからOPMDRV.Xなどをはずしておくのをわすれないでください(Z-MUSICが常駐できません)。

Z-MUSICシステムが扱うファイル

Z-MUSICシステムで扱うデータの形態として，

```
＊.ZMS
＊.ZMD
＊.CNF
＊.ZPD
＊.MDD
＊.JUK
```

のような種類のファイルがあります。これらの拡張子を持ったファイルはZ-MUSICシステムでは特別な働きを持ったものとして扱われます。それぞれについて解説しておきましょう。

●ZMSファイル

MMLの並びをテキスト形式で記述したものです。テキストエディタで自由に変更可能です。演奏の際には内部データに変換されます。

●ZMDファイル

MMLデータを始めから内部データ形式に変換したものです。ドライバが直接扱えるのでゲームのBGMなどはこの形式に変換しておくことが望ましいといえます。この形式のままではデータの変更はできません。

●CNFファイル

Z-MUSICシステムで扱うAD PCM音を列記したものです。単にどのファイルを読み込むかだけでなく，さまざまな加工の指定をすることができます。

●ZPDファイル

CNFで指示されたAD PCM音を作成してひとまとめにしたものです。

●MDDファイル

MIDIが出力する信号を16進数のダンプリストにしたものです。

●JUKファイル

ZP.Xが提供するジュークボックス機能を使うためのファイルです。いくつかの曲を指定しておけば，ほかの作業をしているあいだも順に演奏を続けていきます。演奏する曲名が列記されています。

これらのファイルは以下のような動作をします。

```
COPY MUSIC.ZMS OPM　→　MUSIC.ZMSが演奏される
COPY MUSIC.ZMD OPM　→　MUSIC.ZMDが演奏される
COPY BEEP.PCM PCM　→　BEEP.PCMが鳴る
COPY DRUM.ZPD OPM
　　→　DRUM.ZPDがシステムにセットされる
COPY DUMP.MDD MIDI
　　→　DUMP.MDDがMIDI楽器に送られる
COPY MIDI DUMP.MDD
　　→　MIDIの信号がDUMP.MDDでセーブされる
```

音楽ドライバの動作のしくみ

簡単に動作の仕組みを解説しておきましょう。

Z-MUSICはMMLを処理します。MMLというのは，

```
O4CE8D&F
```

のようなデータを音楽ドライバが解釈してそのとおりに音源を駆動する，というのが基本です。

・演奏チャンネル

X68000で扱えるそれぞれの音源には，いくつの演奏を並列に処理できるかによってチャンネル数というものが定められています。それぞれ，

```
FM音源　　8
AD PCM音源　1（8）
MIDI　　16
```

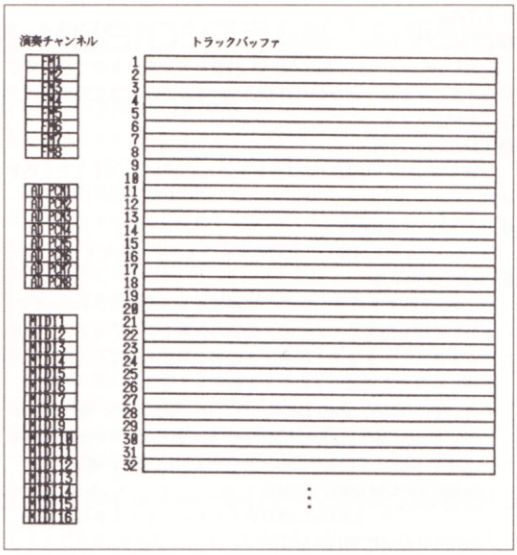
のようになります。MIDI楽器はたくさんの楽器をどんどんつなぐこともできますが，同時に演奏できるのは16チャンネルだけです。Z-MUSICシステムではこれらすべてを同時に制御（シーケンス）することができます。

それぞれのチャンネルに対応したそれぞれのMMLデータを用意して並列に処理すればよいことがわかりますね。

・トラックバッファ

Z-MUSICでは演奏データ（MML）はいったんトラックバッファというワークエリアに格納されます。通常，1パートあたり1本のトラックバッファを使用します(1パートで複数のトラックを使ってもかまいません)。Z-MUSICでは80本のトラックバッファが用意されており，それぞれをどのチャンネルに割り当てるかを指定できます。

これは図のようなイメージです。



・演奏の仕組み

音楽データの記述は，

初期化

トラックバッファの確保

チャンネルのアサイン（割り当て）

MMLをトラックバッファに格納

演奏

という手順になります。

トラックバッファの中にはMMLデータが格納されています。実際にはドライバが扱いやすいように符号化されていますがだいたい下図のようなイメージになっています。

```
Trk1 | T120 L8 V15@15 O3C4&E4CDG.……
```

トラックバッファはBASICの配列のようなものです。多少やっかいなことに使用する前に大きさを指定して置かなければなりません。ユーザーはまずドライバにメモリの割り付け(ALLOCATE)を指示しなければなりません。これにはX-BASICなら，

M_ALLOC()

ZMSなら，

(m)

といったコマンドを使用します。これは各トラックごとに指定することが必要です。

しかし，これから演奏データを作ろうというときに完成時のサイズがわかるわけはありませんから，たいていの場合は大きめに指示することになります。これらの合計が262144を超えない限り少々大きめでもなんの問題もないでしょう。

「音楽データを作る」ということは演奏トラックを用意することにほかなりません。Oh!Xに掲載されている音楽データを見ても，すべて同じ構成を持っていることがわかるでしょう。演奏トラックはそれぞれ独立で動作していますから，それぞれのトラックで時間ずれがないように気をつけてください。

音源を理解する

X68000で音楽ドライバを使うにはざっと2つの道があります。つまり，

内蔵音源を使うか

MIDI楽器を使うか

の2つです。

内蔵音源を使う場合，特別な機器は必要ありませんから誰でもできるというメリットがあります。当然作成されたデータも誰にでも聞かせることができます。

MIDI楽器を使う場合はなんといっても，音質がいいということが挙げられます。

MIDI対応というのは，演奏データをMIDIで規定されたコマンドに直して送ってやるということです。MIDIのコマンド自体は汎用性を備えたものなので，たとえ楽器が間違っていても，とりあえずやってきたコマンドに従って楽器は動作します。しかし，データは各楽器に最適化されていることが多いので，データの対応範囲は狭くなります。直接音源のレジスタ制御ができる内蔵音源と違い，MIDIのコマンドを介した間接的な制御になるのであまり複雑なことはできません。

・FM音源とは

FM音源とはサイン波の発信器を組み合わせて，出力の加算や乗算を繰り返して出力波形を決定する，という純数学的な構成を持ったシンセサイザです。たった50個くらいの簡単なパラメータ指定で多彩な音色が出せるという特徴を持っています。しかし，裏を返せば，それだけ個々のパラメータが複雑に絡みあっているということを示しています。

4つのオペレータでそれぞれ指定された波形で互いにモジュレーション（変調）をかけたり出力を合成していくのですから，パラメータから音色を感覚的にとらえることは困難です。

初心者がまずぶつかるのも，どのようにして新しい音を作っていくのかということでしょう。音色エディタというものもありますが，多くの人はそのようなものを一切使用せずに直接パラメータを記述していきます。

これはもう慣れの問題だけというしかないのですが，あらかじめ用意されている音色を使ってそれをいじっているうちに新しい音ができてくるのでしょう。Z-MUSICシステムでは従来のOPMDRV.Xで定義されていた68音と，X1シリーズで採用されたVIPでの200音を用意しています。

FM音源の欠点としては，「音が薄い」ことが挙げられます。これは，

エフェクタを内蔵していないこと

源波形が単純であること

などが原因です。

しかし，それを克服して音を分厚くするためにさまざまなテクニックが生み出されてきました。

左右のチャンネル指定を複雑に組み合わせて空間的に広がりのある音場を作るようにしたり，1つひとつの音色を聞くと変な音が多いのに，演奏が始まると相互に補いあって艶のある音になったりと芸術的な技巧の世界があります。

応答が非常に高速であることから，ソフトウェア制御により多彩な効果音を作ることも可能です。

・AD PCM音源とは

自然音を録音しておいて加工しながら再生するというのがいわゆるPCM音源なのですが，AD PCMも一応それらの仲間です。ただし，世の中のPCM音源が「データ長16ビット，サンプリング周波数44kHz」という次元なのに対して，「データ長4ビット，サンプリング周波数15kHz」というかたちに圧縮されたデータ構造を持っています。X68000ではそれが1音しかありません。高音部と低音部は再現できませんし，複雑な波形だとノイズがかなり多くなってしまいます。

音楽用に使うにはキツイ音源ですが，ソフトウェア処理によりデータをリアルタイムに合成してしまったり，元データをできるだけクリアなものにしようとする努力によって，最近ではなかなかの音質が得られるようになっています。ただし，それなりに手間がかかり，メモリをかなり消費しますし，CPUパワーも必要ですので気軽にというわけにはいきませんが……。

●AD PCMバンクってなに？

AD PCM音はノート番号で管理されます。たとえば，オクターブ2のCを鳴らすとバスドラムが鳴るといった具合です。

また，Z-MUSICではマルチサンプリングされた楽器音を用意すれば，AD PCMでメロディラインをシーケンスできます。こういった場合，実際の音程と登録ノート番号を一致させたいと思うのは当然でしょう。AD PCMはPCM8.Xのおかげで8声まで対応できますが，楽器音を複数使用するとどうしても登録ノート番号がぶつかることが予想されます。そこで，楽器音ごとに別々のノート番号を与えようというのがAD PCMバンクの考え方です。

AD PCMバンクは音色切り換えコマンドで切り換えられます。た

とえば，ドラムは音色番号の1，ピアノは音色番号の2，オーケストラヒットは音色番号の3といったふうに指定します（AD PCMバンクは4つまで設定されています）。

あらかじめそれぞれに対応したAD PCM音を登録しておかなければなりませんが，これにより普通の楽器と同じ感覚でAD PCMでメロディラインをシーケンスすることができるようになります。

・MIDI楽器を使う

MIDIを導入すれば手軽に高品質な音楽制作ができるようになります。その気になればプロミュージシャンと同等の機材を揃えることも困難ではありません。もちろん，それなりの投資が必要になってきますが。

MIDI楽器にもいろいろありますが，現在X68000で標準的に使用されているのはローランドSC-55です。ユーザーも多く，最近ではMIDIを使った曲のほとんどがSC-55用になってきています。

SC-55を使うということのメリットはなんでしょうか。

SC-55はシンセサイザではなく，再生専用音源モジュールという性格の強い音源です。まず，最初から使える音が揃っていますので音色のエディットが必要ありません。SC-55では多少の音色エディットはできますが，エディットしても劇的な効果はないようです。

シンセサイザでの音色作成という部分がなくなったため，初心者でも扱いやすいものになっているといえるでしょう。

●コントロールチェンジ

MIDIでは演奏表現のためのさまざまなコマンドが用意されています。これらはコントロールチェンジと呼ばれています。演奏に表情を与えるために使われ，

1　モジュレーション
5　ポルタメントタイム
7　ボリューム
10　パンポット
11　エクスプレッション
64　ホールド1（ダンパー）
65　ポルタメント
66　ソステヌート
67　ソフト
84　レガート
91　エフェクト（リバーブ）
93　エフェクト（コーラス）

のようなものがあります。

Z-MUSICではYコマンドでこれを制御します。

```
Y91,40
```

ではリバーブのかかり具合を40に設定します（0～127）。

コントロールチェンジは単純な音程指定だけでなく，演奏表現にかかわるものを実現するためのものです。たとえば，音量や音程の細かな制御やピアノのダンパーペダルやソフトペダルのような表現も可能にしています。

コントロールチェンジは音を発音する前に指定してください。Z-MUSICはMMLは逐次処理ですから，ある音を発音しているあいだはほかの処理は実行できません（基本的には）。

MIDIでは音を鳴らしているあいだにコントロールチェンジを送って音の表情を変えるといった操作も多用されます。たとえば，ピッチベンドは単にコマンドをひとつ送るだけでは音程をちょっとずらせるだけですので「ぎゅい～ん」と連続的に音程を上げたりすることはできません。このような操作には音程変化の値に対応して連続的にデータを送り続けることが必要なのです。

最悪の場合でも，細かい音符で区切り，コントロールチェンジを挟み込みながら“&”で接続するといった操作で実現できますが，特に多用されるピッチベンドとモジュレーションについては専用命令で定形処理に対応しています。

●NRPNの使い方

MIDI楽器はコントロールチェンジで音色やボリュームの設定ができますが，これはMIDI規格で仕様が決められています。そのなかにNRPNという機能の限定されていないコントロールチェンジがあります。SC-55ではこれを使って音色を操作することができます。

```
(t1)　@y1,$20,70　　/ TVF カットオフフリクエンシー
(t1)　@y1,$21,70　　/ TVF レゾナンス
```

ドラムの特定ノートナンバーのピッチ（音程）や音量も変わります。

```
(t1)　@y$18,35,68　/ ドラムピッチ
(t1)　@y$38,35,127　/ TVA レベル
```

この例ではノートナンバー35のKICK DRUM 2のピッチを68にTVAレベル（音量）を127にしています。パラメータの取りうる範囲や初期値は楽器のマニュアルを参照してください。

●楽器を2台同時に使う場合（例：SC-55とCM-64）

SC-55は16パート，CM-64はLAとPCMあわせて15パートの演奏を行います。MIDIチャンネルとは関係なく，パートそのものは固定でデフォルトでは各楽器の全パートそれぞれがMIDIチャンネル1～16を埋めるように割り当てられています（CM-64ではMIDI1だけ空いてますが）。このような場合，パートのMIDI受信チャンネルを変えてやれば任意のMIDIチャンネルに任意の楽器のお好みのパートを割り当てることができます。

SC-55にはパート設定専用の命令がありますのでそれを使うと，

```
.sc55_part_setup 1= {1}
.sc55_part_setup 2= {17}
```

となります。

これでSC-55のパート1をMIDIチャンネル1にアサインできます。デフォルトでは1～16パートがチャンネル1～16を受信してしまうのであとのパートがいらない場合はOFF（17と書いておけばいい）。例ではパート2しかOFFしていないので，使うチャンネルの数によって臨機応変に対応してください。

CM-64には専用命令はありませんがエクスクルーシブメッセージで設定できます。

```
/ PCM SOUND PART
.roland_exclusive 16,22 = {$52,0,10
        1,16,16,16,16,16}          / MIDI ch#
```

これはCM-64のPCMパート1をMIDIチャンネル2（1ではないことに注意）へ割り当てることの意を表します。あとのチャンネルは全部OFFです。ひととおりの設定が終わったら，あとは，

```
(m1,1000) (aMidi1,1)
(m2,1000) (aMidi2,2)
```

とやればトラック1でSC-55のパート1，トラック2でCM-64のPCMパート1を操作できます。

この例ではSC-55とCM-64についての解説しかしていませんが，同様の設定はどの楽器でも可能です。

ZPDデータでディスクが圧迫されている

Z-MUSICでは演奏時にコンフィギュレーションファイルを読み込むことで，個別のAD PCMファイルを持たなくてもその場で必要なAD PCM音を組み立てて演奏をすることができます。しかし，読み込み，加工の手間がかかることから，個別にZPDファイルを用意しておくのが一般的になったようです。

ドラムなどは一般的に使われる音が限られているのですから，曲ごとに同じようなサンプリングデータをたくさん持っているのは，ずいぶん無駄なことになります。共通で使える部分だけでもまとめることはできないかと考えるのは自然の成り行きでしょう。

今回のZ-MUSICではZPDデータを複数個指定することが可能です。たとえば，ZMSファイルのAD PCMブロックデータの指定部分で，

```
.adpcm_block_data   B:STD_SET.ZPD
.adpcm_block_data   B:POWER_SET.ZPD
.adpcm_block_data   B:TEST.ZPD
```

のように指定することで，STDSET，POWERSET，TESTの3種類のZPDファイルを読み込み結合します。AD PCMバッファは指定したZPDファイルの合計だけ必要です。ちなみに，割り当てがぶつかった場合にはあとから指定されたものが優先されます。STD_SETは一般的なMIDI楽器のドラムキット配列を参考にして作成されています。環境変数zmusicを適正にSETしたうえで，

```
A>ZPCNV STD_SET
```

のようにするとSTD_SET.CNFからSTD_SET.ZPDが作成されます。ここで使われているサンプリング音はほとんどがEXTRA_PERCから採用されています。

MEASURE 1
イントロダクション

ここではZ-MUSICシステムの概要を簡単に説明します。

1.1. はじめに

コンピュータミュージックを楽しむにはいくつかの方法があります。

1) 実際に弾いたものをコンピュータ，シーケンサのメモリに記憶しておくもの

2) コンピュータ，シーケンサのメモリに直接演奏データを書き込んでいくもの

3) 1)と2)の複合的な方法

などです。2)はコンピュータのキーボードを叩くことから「打ち込み」と呼ばれます。3)はつまり，人間が弾いたものをあとでコンピュータを用いて修正するといったものです。

ちょっと昔までは3)の方法が圧倒的にメジャーでしたが，音源の進歩とミュージックツールの進化により最近は2)の方法もメジャー化してきています。最近は譜面も起こさずに直接「打ち込み」を始める作曲者も少なくないようですし，スタジオのマニピュレータ(平たくいえば打ち込み人)のなかには実際に弾いたものよりも人間臭さを表現する人もいると聞きます。

さて，本書でお話しするのは2)の方法です。2)の方法の魅力といえば，自分が楽器の演奏ができなくてもイメージどおりの演奏が可能という点でしょう。さらに，一度仕上げた曲は，ディスクなどの記憶装置に保存しておけばいつでも再現可能というのもこの方法の魅力です。

打ち込みの方法にも実はいろいろな方法があります。いちばん原始的なものは完全な数値入力です。音階，発音時間などすべてを細々と入力していくものです。最近は「ステップ入力」と呼ばれます。数年前にMZやPC-8001用に出ていたCMU800シリーズはこの方法でした。これを進化させたものが譜面入力による「打ち込み」です。要するに五線譜に音符を置いていくタイプのものですね。楽器の演奏はおろか譜面を読むことができない人にでも簡単に音楽を作ることができるため初心者層には圧倒的な人気を誇ります。また印刷機能が充実していれば美しい譜面を得ることができますので，プロでもこのタイプを使っている人が多くいます。

さて，「Z-MUSIC」では現在MML（注1）という「打ち込み」方式を採用しています。これはいわば「音楽記述言語」というもので「音楽をプログラムする」というイメージが正しいでしょう。完全な数値入力よりはずっとドキュメント性が高いうえ，入力する量も少ないのでパソコンを基盤として音楽を楽しむ人にはピッタリのものといえます。欠点としては，小，中学校程度の音楽知識が必要なこと，ある程度コンピュータを扱える人(DIRやCOPYがわかればOK)でないと駄目なこと，ドキュメント性が譜面入力よりは低いこと，などです。まあ，ドレミが読めて，ディスクをフォーマットしたことがあれば素質十分です。

MMLの利点としては，慣れればたいていの方式よりも高速に1曲を仕上げることができること，細かいニュアンスなどに凝ることができることなどが挙げられます。また，移植性が高い点も見逃せません。ほとんど方言のような感覚で他機種の演奏データやほかの音楽ドライバ用のデータを，自分の使っているMMLに修正して利用することができるのです。すでに多くの音楽ドライバ用のデータがZ-MUSICシステム上で演奏可能となっています。

さあ，「Z-MUSIC」を懸け橋にしてあなたもコンピュータミュージックの世界に浸りませんか。

注1) 今後，違った入力方式に対応した支援プログラムが発表されれば，別の入力方式（たとえば譜面入力方式）も可能です。

1.2. ZMUSIC.Xとは

ZMUSIC.XとはX68000本体付属のOPMDRV.Xを改造したりするものではなく，まったく新しくゼロから開発されたミュージックプログラム開発言語です。X68000の内蔵音源であるFM音源8声とADPCM1声(PCM8.X(C)江藤啓を使用すれば8声まで)，MIDIボードが接続されていればMIDI楽器も同時にコントロールすることができます。

一般のほかのドライバではひとつのMIDI楽器をX68000の外部音源という感じで扱っていましたが，「ZMUSIC.X」ではX68000をホストに複数のMIDI楽器をコントロールすることができるように設計されているので，どんどん自分のMIDIシステムを広げていくことが可能です。

ここでZMUSIC.Xの特長を列挙してみます。

1) OPMDRV.Xに上位コンパチ

2) FM音源，AD PCM，MIDI楽器を同時に演奏可能

3) もちろんMIDI楽器を持っていない人でも多彩なコマンドを内蔵音源に対して使用可能

4) 汎用トラックを80本持ち，そのうち最大32トラックを同時に演奏可能

5) AD PCMを音階MMLでコントロール可能。独自のAD PCMドライバ搭載

6) ポルタメントやオートベンド，和音やLFOなどの特殊コマンド系がFM/MIDIの両方で使用可能

7) AD PCMコンフィギュレーションファイルで，AD PCMデータの音程とボリュームが変更可能。合成，エンベロープの変更も可能

8) 任意の時間に演奏中の任意のトラックに任意の演奏データを割り込ませて演奏することが可能（効果音モード）

9) 演奏処理がOPMDRV.Xの平均3倍以上高速。さらにテンポずれが起こりにくい設計

10) MT-32のほかにU220/M1/SC-55などに対応した楽器個別のコマンドを装備

11) MIDI楽器側の音色や設定をファイルに吸い出す機能を装備

12) オブジェクトレベルの演奏データを得ることが可能(コンパイル機能)

13) あらゆる命令が機械語レベルで操作可能。ゲームなどのBGMドライバとして威力を発揮

14) 専用A/Dコンバータによってサンプリングされた高音質のAD PCMデータライブラリの標準装備

15) 全情報公開。ライセンスフリー

16) PCM8.X(C)江藤啓に完全対応。AD PCM音を最大同時に8声までシーケンス可能

17) 複数のX68000による同期演奏

とまあ，大袈裟なところもありますがこんな感じです。

同人ソフトなどの制作者やグループにとっては8)，9)，12)，13)，15)あたりが魅力でしょうか。

ではまず，ディスク1の中身を確認してみましょう。

1.3. ディスク1

ディスク1には「ZMUSIC.X」のほかに「ZP.R」「ZPCNV.R」「ZVT.X」「ZPLK.X」「MUSICZ.FNC」「ZMUSIC.L」「PCM8.X」といったプログラムが収録されています。それぞれを簡単に説明します。

●**ZMUSIC.X**……Z-MUSICシステムの本体です(詳しくはMEASURE2参照)。

●**ZMSC.X**……ZMUSIC.Xのタイニー版です(詳しくはMEASURE2参照)。

●**ZP.R**……ZMUSIC.Xは内部コール，ワークなどのすべてを公開しますので支援プログラムの開発が容易に行えます。その一例としてX68000のOS「COMMAND.X」から簡単に音楽が演奏できるレコードプレイヤー的なものを制作してみました。機能としては「ZMUSIC.X」用に作られた音楽データの演奏，ジュークボックス，音楽データの制作を援助するデバッグ機能などがあります(詳しくはMEASURE7参照)。

●**ZPCNV.R**……外部記憶が十分にある場合，演奏するたびにAD PCMデータを1ファイルずついちいち読み込んでいたのでは非能率的です。そこでその曲に必要なAD PCMデータ群をひとまとめにしてしまうのがこのプログラムです(詳しくはMEASURE6参照)。

●**ZVT.X**……自分でAD PCMデータをサンプリングしたり，既存の

AD PCMデータを編集したりするためのプログラムです(詳しくはMEASURE8参照)。

●ZPLK.R……AD PCMファイル（または16ビットPCMファイル）を加工したり，ほかのAD PCMファイルと結合分離したりするためのツールです(詳しくはMEASURE6参照)。

●MUSICZ.FNC……「ZMUSIC.X」用のX-BASIC外部関数です。X-BASICから「ZMUSIC.X」を使うときに使用します(詳しくはMEASURE3参照)。

●ZMUSIC.L……C言語からZ-MUSICの機能を使用したり，MUSICZ.FNCを使って作成されたプログラムをコンパイルするときに使用します(詳しくはMEASURE16参照)。

●PCM8.X……江藤啓氏によって作成された常駐型AD PCMドライバです。本来なら単音のX68000のAD PCM音源を高度なソフトウェア処理によって制御し同時に８和音まで鳴らすことのできる画期的なアプリケーションです。Z-MUSICシステムはこのpcm8.Xに対応しており，これによってAD PCM８チャンネル分のシーケンス（あるいは８音までのポリフォニック演奏）が可能です。

また，ディスク２，３に入り切らなかった標準サンプリングデータとして，

・RAP……英語の掛け声
・ADDITION……その他のAD PCMデータ

が収録されています。

そのほか，OPMD.Xコンパチのデータを作るためのコンフィギュレーションファイル「BOS.CNF」(MEASURE6参照)，KORG M1のプリセット音のベンド幅を１オクターブに変更するためのデータファイル「M1_ZM_STD.MDD」(MEASURE9参照)，サンプル数曲が収録されています。

ソースリストについて

ディスク１には今回収録されているZ-MUSIC関連のプログラムのすべてのソースが収録されています。アセンブラで記述されていますが拡張子'.HAS'はパソコン通信ネットワークなどで入手可能なフリーソフトのハイスピードアセンブラ「HAS.X」用に記述したものでシャープの「AS.X」ではアセンブルできません。HAS.Xは満開製作所製の月刊ディスクマガジン「電脳倶楽部」1991年６月号付録の「自由軟盤壱號」などにも収録されています。

1.4. ディスク２，３

ディスク２，３にはAD PCMデータが収録されています。これまでOh!Xに掲載されていたZ-MUSIC用の音楽プログラムはほとんどがこの２枚のディスクに収録されているものを使っていました。ここにあるAD PCMデータが標準となります。

さて，今回収録されたAD PCMデータの多くはNF-ELECTRONIC INSTRUMENTSのA/D，D/Aコンバータ「FV-665」やローランドS-330を使用してサンプリングしたものです。X68000本体でサンプリングするとどうしても高・低音部分がカットされてしまったり，「サー」というノイズが入ってしまったりと，原音を忠実に再現できませんでした。そこで今回こういった手法をとったわけです。ただし，原音をなるべく高音質で得るために音量を音割れしない程度の最大音量で収録しています。どの音も音量が大きめなのはそういうわけです。でも「ZMUSIC.X」ではAD PCMの音量が変更可能なので問題はありません。

サンプリングデータは主に以下のように種類別に分類されて収録されています。

・SNARE……スネアドラムです
・BASS……バスドラムです。オーケストラのコンサートバスドラムもあります
・TOMTOM……タムタムです。エレクトリックタムもあります
・CYMBAL……シンバルのほかにハイハットもあります
・ETHNIC……ラテン楽器や和風楽器があります
・EFFECTS……効果音的なものが入っています。ゲームの効果音を作ったり，前衛的な音楽を作るのに役立ちます
・ACCENT……ハンドクラップやリムショットなど曲のメリハリをつけるためのものが収録されています。オーケストラで使用されるティンパニもあります
・ORCH……オーケストラヒットです。アタックの弱いものから強いもの，クワイアとユニゾンしたもの，エレクトリック味のものなどさまざまなタイプを収録しています。ファイルはまとめて圧縮されていますので，使用する際にはLHA.Xを使って展開してください。

また，ディスク３には，

・PIANO_P16……ピアノの音を16ビットPCM形式でサンプリングしたものです

が含まれています。

1.5. ディスク４，５，６

ディスク４～６はver.2.0になって拡張されたサンプリング音です。

●ディスク４
・EXTRA_PERC……リズムキットの拡張用に用意されたドラムなどのサンプリング音です
・BRASS……管楽器の音をサンプリングしたものです
・SYNTHE……シンセサイザ音をサンプリングしたものです
・GUITAR……エレクトリックギターやアコースティックギターのサンプリングデータです
・STRINGS……弦楽器をサンプリングしたものです
・CHOIR……人声のコーラスをサンプリングしたものです
・PIANO……ピアノの音をサンプリングしたものです。
・TR808SET……ローランドの名作リズムマシンTR808の音をサンプリングしたものです

●ディスク５
・DG_P16……ディストーションギターの音色を16ビットサンプリングしたものです
・SAX_P16……アルトサックスとテナーサックスを16ビットサンプリングしたものです
・CHOIR_P16……人声のコーラスを16ビットサンプリングしたものです
・ORGAN……パイプオルガンの音を16ビットサンプリングしたものです
・STRINGS_P16……弦楽器の音を16ビットサンプリングしたものです

●ディスク６
・SOUND_EFFECTS.LZH……各種効果音を集めたものです
・EBASS……ベースの音を集めたものです
・BRASS_P16……管楽器を16ビットサンプリングしたものです

これらのデータのうち，16ビットサンプリングされたものは，音色のアタック部分とループ部分を分離して収録してあります。PCM8.Xを組み込んだうえで，ZPLKなどのツールで加工したデータを使えば，AD PCMを使用してメロディラインをシーケンスすることも不可能ではありません（メモリが多少多めに必要。CPUパワーも多少必要。10MHzでは苦しいこともある)。

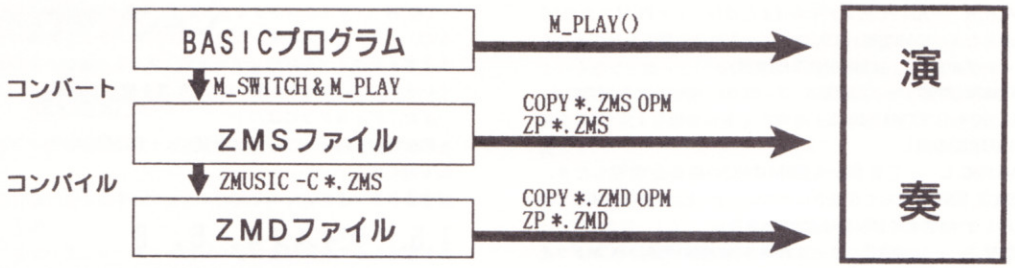
最後に，本書についてくるディスクはコピーフリーです。自由にバックアップを取って構いません。

1.6. 著作権について

法律上，日本では著作権の放棄ができませんので，プログラムの著作権は作者西川善司に保留されます。しかし，プログラムの性質上，「ZMUSIC.X」のオリジナルを開発した私，西川善司は「ZMUSIC.X」およびこれらを支援するプログラム(サブルーチンなども含む)すべての使用に関するライセンス権を放棄します。よって特に断らずに商的利用ができます。つまり市販だろうが同人だろうが勝手に「ZMUSIC.X」を組み込んだソフトを売ってもいいということです。すでに多くのソフトに組み込まれていますので安心して使ってください。

ただし，ひとつだけ守ってほしいことがあります。それは「ZMUSIC.X」の改造についてです。改造はもちろん各自に自由で行って構わないのですが，データの完全な互換性が保持されないような改造を施した際には「ZMSC.HAS」のバージョンID番号を$F0以上にしてほ

図1　Z-MUSICにおけるデータの流れ



しいのです(オリジナルは$20です。ソースリストZMSC.HAS参照)。また,オリジナルのZMUSIC.Xで演奏できないデータフォーマットにした場合はファイルの拡張子も変更しておいてください。これは混乱を防止するためです。今後のバージョンアップ版との兼ねあいもありますし,改造を施したものではオリジナルバージョン用のデータを頭っからはじいてほしいのです。ぜひご協力ください。

移植に関しても自由に行って構いません(ただし,完全にコンパチでない場合はバージョン番号をやはり$F0以上にしてください)。移植版をOh!X編集部へ送ってくれればTHE USER'S WORKSなどのコーナーで紹介したいと思います。

1.7. ZMUSIC.Xの概要

ZMUSIC.X上で演奏するデータを作るためには2通りの開発方法があります。

ひとつは「MUSICZ.FNC」を組み込んだX-BASIC上で開発する方法です(MEASURE3参照)。基本的には「MUSICZ.FNC」に設けられたZMUSIC.X制御命令を記述していくことにより演奏プログラムを作成していきます。同じ処理を繰り返したいときは「for～next」,条件分岐をさせたいなら「if ～ then ～」……というようにX-BASICに用意された多彩な命令を織り交ぜて開発を行うことができます。また,文字変数などをうまく使えば少ない手間で長い曲も作れそうです。さらに,曲データ以外の音楽関係のツール,たとえば音色エディタなども作ることができるでしょう。

2番目の方法はCOMMAND.X上からエディタを使ってMMLファイルを作成する方式です(MEASURE4参照)。Z-MUSICシステムではこのMMLファイルを,拡張子「.ZMS」をつけることを前提に特にZMSファイルと呼ぶことにします。この形式ではZMSコマンド(MEASURE4参照)と呼ばれるZMUSIC.X制御命令を1行1行に記述していくことで演奏プログラムを作成していきます。X-BASICのような制御文はないので一見開発が不便そうですが,エディタに備えられたマクロ機能や複写機能を駆使すればBASIC以上の開発環境になりえます。また,X-BASICで作られた演奏プログラムもこのZMSファイルに変換することができます。これには'm_switch()'命令を使用します(MEASURE3参照)。

実はX-BASICで記述された演奏プログラム,エディタによって記述されたZMSファイルがZMUSIC.Xへ受け渡されるとZMUSIC.Xは内部で$47,$30,$30……のような16進数のデータ列で構成されたオブジェクトに変換しています。演奏しながらいちいち命令語を認識していたのでは処理が遅くなってしまうので,演奏直前に必要な計算をすべて済ませてしまうわけです。X-BASICで記述された演奏プログラム,エディタによって記述されたZMSファイルを演奏するとき,ほんの一瞬待たされますが,このときそのオブジェクトデータへ変換しているのです。Z-MUSICシステムではこのオブジェクトデータを特にZMDデータと呼ぶことにします。

通常はZMDデータを意識する必要はないのですが,あえてこのZMDデータを取り出す方法があります。ZMUSIC.Xに備えられているコンパイル機能がそれです。具体的には,

A>zmusic -c ZMSファイル名

のようにします(ZMUSIC.Xはアセンブラ やCコンパイラのようにも機能します)。するとZMSファイルの主ファイル名に拡張子'.ZMD'を添付したファイルが生成されます。これをZ-MUSICシステムでは特にZMDファイルと呼びます。

このZMDファイルも演奏可能で,その際は当然ZMDデータに変換する作業がありませんので,まったく待たされずに演奏が開始されます。高速応答が要求される場合(たとえば自作ゲームに曲を入れる場合など)は演奏データをZMDデータで管理したほうがよいでしょう。ただし,データや環境の変更が困難ですので,音楽データの配布を目的とした場合はZMD形式は避けるべきです。

以上のことを図にまとめると図1のようになります。

Oh!Xへの投稿は

Oh!Xのミュージックプログラムコーナーではzmusic.Xを使用した音楽プログラムを募集しています。掲載時にはソフトバンク規定の原稿料が支払われます。なにかできたらボツを恐れずじゃんじゃん送ってきてください。では,応募要項を以下に示します。

プログラムのデータ形式はBASIC,ZMS形式どちらでも構いません。ただしZMS形式をコンパイルしてできる「ZMD」形式のみでの投稿は受け付けません。

また,AD PCMデータを使用する場合にはできるだけディスク1～3に収録されているもののなかから選んでください。ディスク4～6に収録されているデータを使った場合はその旨明記していただけると助かります。独自でサンプリングしたものは特殊な状況を除き掲載することはできません。AD PCMのコンフィギュレーションファイル(MEASURE6参照)もお忘れなく。ZPCNV.Rを通してできるZPD形式のAD PCMデータのみでの投稿は受け付けられません。

MIDI対応の場合は楽器名と特殊なセッティングが必要であればそれについての解説を添えてください(ボリュームつまみの位置とか音量バランスについて)。複数の楽器を使用していたり,一般的でない楽器を使用している場合にはカセットテープに録音したものを添えておくと有利です。

まとめると,

1.「BASIC」または「ZMSファイル(OPMファイル)」形式。ZMDのみでは不可。
2. AD PCMを使用しているならばコンフィギュレーションファイルも添える。
3. プログラムや曲に関する解説があれば文書,あるいは文書ファイルを添える。

ご協力ください。

MEASURE 2
ZMUSIC.Xのオプションスイッチ

ここではZMUSIC.Xのコマンドオプションについて解説します。

2.1. はじめに

さて,ZMUSIC.Xというプログラムには大きく分けて2つの機能があります。ひとつはシステムに常駐して音楽演奏をする機能，もうひとつは前章で述べたZMDファイルの生成機能です。よってZMUSIC.Xのコマンドオプション/コマンドスイッチも大きく2つの機能に分けられます。ひとつは常駐する際のパラメータとしての機能，もうひとつは演奏データをコンパイルする際にこれを支援するものです。

各スイッチは「/」「−」の後ろにこれから解説するコマンドスイッチのアルファベットを記述し，さらに必要であればその直後に数値などのパラメータを記述することによって指定します。具体的には，

A>zmusic −または/[各スイッチの英数字][パラメータ]

のようになります。スイッチは（例外を除き）順不同でいくつでも記述することができます。また，

A>zmusic −h　または　−?

でコマンドスイッチの簡単な説明を見ることもできます。

それではそれぞれのスイッチを解説していきましょう。

2.2. ドライバ組み込み時のスイッチ

-A

割り込みにタイマAを使用する。デフォルトではタイマBを使用する。

タイマAはテンポ77から300までを有効範囲とし，かなり正確なテンポをキープできる。外部シーケンサ/リズムマシンとの同期演奏にはこちらのモードをすすめる。

よくわからない人は特に設定する必要はない。

-Bファイルネーム

AD PCMブロックデータ（ZPDデータ）をドライバ起動時に読み込む。後述の'-P'スイッチでAD PCMバッファのサイズを指定していない場合は自動的にそのブロックデータのサイズがバッファサイズになる。ファイルネームの拡張子を省略すると自動的に'.ZPD'が添付される(MEASURE6参照)。

-E

外部シーケンサ/リズムマシンとの同期演奏を行う。X68000をホストにしたMUSICシステムならばこのスイッチは設定する必要はない。複数のX68000を同期させて演奏させるにはこのスイッチを設定すること。

具体的には，演奏開始時にスタートメッセージ$FA，演奏再開時にコンティニューメッセージ$FB，演奏停止時にストップメッセージ$FCを送信するようになる。また,演奏データ中で指定したテンポにあわせてタイミングクロック$F8を送信する。

-G

ZMUSIC.X起動時にロゴや常駐を報告するメッセージを表示しない。

-J

MIDI楽器へコントロールチェンジを送出したあとに若干のウェイトを入れる。ウェイトの量は−Xで指定してあるものと同じ値が使用される。極端に反応の遅いMIDI楽器を使用する際に有効である。通常は指定する必要はない。

-M

MFPの多重割り込み対応モードにする。ラスタースクロールなどの処理をジャマしない。高度な多重割り込みを駆使したアプリケーションと併用する場合にはこのスイッチを指定するとよい。通常使用時やよく意味のわからない人は設定する必要はない。

-N

初期化なしモードにする。このスイッチを指定してZMUSIC.Xを常駐させると以後，演奏データの変わり目などで楽器などやワークの初期化を行わない。これはゲームなどのBGMを機械語レベルで演奏するときに曲の変わり目の処理を軽くするためのものである。通常は設定する必要はない。

-On

PCM8.Xが先に組み込んであることが前提のスイッチである。従来のAD PCM 1声の曲データをPCM8.Xを用いて無理やりポリフォニックで鳴らしてしまおうというものである(図1参照)。ただし，本来のPCM8モードとは違うものなので注意すること(2.5.参照)。パラメータのnは音楽演奏に8声あるAD PCMチャンネルを何声音楽用に確保するかを設定するものである。nは1≦n≦8でn省略時は自動的にn=8が採択される。余ったチャンネル数は効果音専用とみなして音楽演奏には使用しない。たとえば，

A>zmusic -o6

とした場合，チャンネル1～6をBGM専用にし，チャンネル7，8を効果音専用とする。

-Pn

AD PCMデータバッファとしてnKバイト確保する。'-P'スイッチを設定しないとAD PCMバッファサイズは256となる。0と設定した場合は,以後AD PCMに関係するコマンドは無効になる(MEASURE6参照)。

-R

常駐しているZMUSIC.Xをシステムから削除する。コマンドラインから常駐させた場合のみ有効で,'CONFIG.SYS'から'DEVICE='で組み込んだ場合は常駐解除はできない。

-Sファイルネーム

MIDIのダンプデータ,AD PCMデータのコンフィギュレーションファイルなどのセットアップファイルを読み込みドライバ起動時に実行する。AD PCMコンフィギュレーションファイル中，データの加工を行うように設定してある場合には必要なだけのワークを確保しなければならない(MEASURE6参照)。ファイルネームは拡張子の省略は認められず，さらにMIDIダンプデータは拡張子'.MDD'のときのみ有効である。

さて，ここに演奏データのファイル名を書けばドライバ起動時に曲を演奏させることも可能。もちろんその際，AD PCMを使用していたりするならばそれらに関係するバッファも確保しなければならない。

例

A>zmusic.x -bdrum.zpd -w30 -p100 -S music.zmd

-Tn
#n

トラックバッファをnKバイト確保する。デフォルトでは128Kバイト確保する。サイズの大きいデータを演奏する際には状況に応じて設定すべきである。

-Wn

ワークエリアとしてnKバイト確保する。ワークエリアはAD PCMデータの加工やコンフィギュレーションファイルのバッファ,MIDI楽器の設定の取り込み用バッファ，波形メモリバッファと多目的に使用されるため，作業内容に応じて設定すべきである。AD PCMのデータ加工用のワークとして設定する場合の目安は，加工するデータのうち最長のものの長さを4倍した大きさである。AD PCMデータ登録をすべてZPD(MEASURE6参照)で行うというのであれば指定は不要である。デフォルトは12Kバイト。

-Xn

EOX（エンドオブエクスクルーシブ）メッセージを送ったあと，n/60秒のウェイトを与える。デフォルト値は3で，通常は設定する必要はないが，極端に応答の遅いMIDI楽器に対しては設定するとよい（U110/220,D70，JW-50……など）。

2.3. CONFIG.SYSからの組み込み

ZMUSIC.Xはデバイスドライバでもあるので，もちろんCONFIG.SYSから組み込むことができます。書式ははかのデバイスドライバと同様に，

DEVICE＝ZMUSIC.X　オプションスイッチ…

のようにします。

また，

DEVICE＝ZMUSIC.X -s曲ファイル名

のようにすればデバイスドライバ組み込み時から曲を流すことができます。もちろん演奏に必要なバッファは確保する必要があります。

2.4. コンパイル支援系

-C　ファイル名1　ファイル名2

ファイル名1をコンパイルし，生成されるオブジェクトをファイル名2でセーブする。ファイル名2は省略可能で，省略時はファイル名1に拡張子'.ZMD'をつけたファイル名でセーブされる。

例

A>zmusic -c music.opm　→　music.ZMD

この'-C'スイッチは必ずコマンドラインの最後に書かなければならない。'-C'スイッチの後ろに書かれたスイッチは無視される。

例

A>zmusic -c music.opm -w100　←'-w100'は無視される

また大文字の'C'でこのスイッチを設定した場合は最適化処理（空トラックの削除処理）を行わずにコンパイル動作を行う。

-Q　ファイル名1　ファイル名2

コンパイル後，その演奏データのトータルステップカウントを計算する。MML命令[do]～[loop]以外（たとえば[coda]～[tocoda]など）を用いて無限ループを構成していると半永久的に計算を繰り返してしまうので注意すること。チェックサム的な役割を果たすのでOh!Xなどの雑誌メディアに演奏データを投稿する際はこのトータルステップカウントを原稿などに添えること。

-Tn
#n

トラックバッファを確保する。デフォルトでは128Kバイト確保するが，コンパイル中'out of memory'が出たときに設定するとよいだろう。常駐時と同じイメージでコンパイルが行えるよう，あえてこういう仕様にした。

-Wn

コンパイルされたデータを整理する際にこのエリアを使用する。デフォルトで128Kバイト確保しているのでほとんどの場合は設定する必要はない。極端に長い曲をコンパイルする際には設定の必要がある。

2.5. PCM8.Xモードについて

PCM8.Xとは，江藤啓氏により作成された，X68000本体の改造をせずにAD PCM音を8和音までリアルタイムに発音できるアプリケーションです。このPCM8.Xを事前に組み込んでからZMUSIC.Xを組み込むとZMUSIC.Xは自動的にPCM8.Xモードで動作します。ZMUSIC.Xを組み込んだあとにPCM8.Xを組み込むことはできません（エラーが発生します）。

さて，ZMUSIC.XのPCM8モードには2通りあります。

ひとつはAD PCM音源を8つのほぼ独立したチャンネルとして使用するモードです。このモードにするにはZMUSIC.Xを組み込むときに特別なスイッチを設定する必要はありません（AD PCMバッファの確保は必要ですが）。このモードを特にPCM8独立チャンネルモードと呼ぶことにします。

もうひとつのモードはPCM8.Xを組み込んだあと'-O'スイッチをつけてZMUSIC.Xを組み込むと設定されるモードです。このモードはPCM8.X用ではない曲，つまりAD PCM 1声用の曲データのAD PCM音をむりやりポリフォニックに演奏するモードなのです（図1）。

たとえばシンバルを叩いたあと，このシンバルが鳴り終わらないうちにスネアを叩いたとすると，AD PCM 1声の従来の曲ではシンバルの音がブツリと切れてスネアの音に切り換わっていました。ここをブツ切りにしないで（PCM8.Xを用いて）ちゃんと前後の音を重ねて演奏してしまおうというのがこのモードです。このモードはPCM8ポリモードと呼ぶことにします。

PCM8独立チャンネルモード時の具体的な操作方法に関しては後述します（MEASURE15参照）。

2.6. 3タイプのZMUSIC.X

Z-MUSICシステムには，

●シャープ純正（または互換）のMIDIボードに対応した標準的なUNIVERSALバージョン

●RS-232Cポートを使ってMIDI出力を行うRS-232Cバージョン

●ネオコンピュータシステム（NCS）製のサブボード，POLYPHONを使用してMIDI出力を行うPOLYPHONバージョン

の3つのタイプが存在します。

3タイプとも内蔵音源（FM音源，AD PCM音源）の制御部分は，まったく同じプログラムで構成されており，違うのはMIDI出力に関する点のみです。もちろん3タイプともX68000/X68030の両方に対応しています。

●RS-232Cバージョン

RS-232CバージョンはX68000本体後面パネルのRS-232C端子に市販のRS-232C-MIDIアダプタ（例：COME ON MUSIC製 MA01：10,000円）を装着し，これを使用してMIDI制御を行うZMUSIC.Xです。標準のUNIVERSALバージョンとは以下の相違点があります。

・MIDI出力の処理速度がUNIVERSALバージョンと比べて少し遅い。

・純正MIDIボードを装着していてもRS-232C-MIDIを強制的に選択する。

・外部同期モード（オプション'-E'）がない。

●POLYPHONバージョン

POLYPHONバージョンはサブボード「POLYPHON」を同梱のプログラム「PCM8SB.X」で制御し，MIDI出力を行うZMUSIC.Xです。ZMUSIC.Xの組み込みよりも先にポリフォンに同梱のプログラム「PCM8SB.X」を組み込まなくてはMIDI制御が行えません。PCM8SB.Xを組み込まずにZMUSIC.Xのみを組み込んだ場合は内蔵音源の制御のみ可能となります。標準のUNIVERSALバージョンとは以下の相違点があります。

・MIDI出力の処理速度がUNIVERSALバージョンと比べて少し遅い。

・MIDIボードを装着していてもPOLYPHON-MIDIを強制的に選択する。

・POLYPHON-MIDIによるMIDI-INはサポートされない。

・外部同期モード（オプション'-E'）がない。

2.7. 機能縮小版ZMSC.X

MMLコンパイル処理部分などを省いたZMD再生専用のZMUSIC.XがZMSC.Xです。ZMSを扱う機能はまったく持っていないため，ZMSファイルを演奏したりすることはできません。しかし，常駐サイズはZMUSIC.Xの半分程度で済むため，ゲームやその他のアプリケーションから使用する場合には適しています。

ZMUSIC.X同様にUNIVERSALバージョン，RS-232Cバージョン，POLYPHONバージョンの3タイプがありますがディスクにはUNIVERSALバージョンだけが収録されています。その他のバージョンはアセンブルしなおすことで生成されます。

それぞれ，付属のバッチファイルを使って，

AZ_3　　(RS-232C版)

AZ_4　　(POLYPHON版)

のようなスイッチをつけてアセンブルしてください。

以下にZMUSIC.XとZMSC.Xとの相違点を述べます。

・ファンクション$01 m_alloc
ファンクション$02 m_assign
ファンクション$03 m_vget
ファンクション$06 m_trk
ファンクション$07 m_free
ファンクション$3e set_fm_master_vol
はなんの機能も実行しない(削除されている)。

・ファンクション$3c get_play_workは,戻り値のd0.lが意味をなさなくなる(a0.lにはZMUSIC.X同様の正しい戻り値を返す)。

・ファイル名'OPM'への出力による制御も行えるが,ZMDまたはZPD以外を出力した場合はなんの機能も実行しない(ZMSコマンドの使用もできない)。

・ファイル名'MIDI'によるMIDI楽器のデータ吸い取り機能はそのまま使用できる。MDDの転送も行える。

・IOCSコール_OPMDRVによるコントロールはできない。TRAP #3でのみファンクションコールを実行できる。

・ZMDの演奏やZPDの変更登録はZMUSIC.X同様にZP.Rで行える。ただし,ZMSの再生は行えない。

・MUSICZ.FNCは使用できるが,
'm_alloc'
'm_assign'
'm_vget'
'm_trk'
'm_trk2'
'm_free'
'fm_master'
'm_debug'
はなんの機能も実行しない(削除されている)。

・デフォルトでトラックバッファを32Kバイト,ワークエリアを0Kバイト,AD PCMバッファを256Kバイト確保して常駐する。ZMUSIC.Xとはデフォルトが違うので注意すること。

環境変数zmusicについて

Z-MUSICでは,カレントよりファイルが見つからない場合は環境変数'zmusic'に書かれたパスに従ってファイルを検索します。なお,Z-MUSICシステムではAD PCMデータに限らず,ファイルを読み込むときは必ずこのルールに従うので,ハードディスク上など多くのディレクトリに分散されて必要ファイルが収められているときは,なるべく設定しておくべきです。

設定例

```
set zmusic=a:¥drums;a:¥pcmfiles;b:¥rythm¥snare;
      ↑小文字のみ可
```

環境変数はHuman68kの仕様から255文字を超えて設定することはできません。255文字を超えてZ-MUSICのファイルパスを記述したい場合には,環境変数zmusic0〜zmusic9に分けて設定してください。この時環境変数zmusicは木でいう幹に,環境変数zmusic-0〜zmusic9は枝の扱いになります。幹から枝への参照は,環境変数zmusic中に'/'とその後ろに補助環境変数zmusic0〜zmusic9までを表す数値'0'〜'9'を書き,さらにその後ろにベースとなるパスを書くことによって指定できます。たとえばzmusic標準のADPCMライブラリのディレクトリ名であるBASS, SNARE, TOMTOM, CYMBAL, ETHNIC, EFFECTS, ACCENT, ORCH, RAP, ADDITIONを検索させる場合を考えます。これらのディレクトリが,FドライブのADPCM_LIBの下に存在する場合,

```
SET zmusic=/0F:¥ADPCM_LIB;
          ↑ zmusic0を示す数値
          補助環境変数の検索を指示するコマンド記号'/'
          ('-'で代用も可能)
SET  zmusic0= BASS ; SNARE ; TOMTOM ; CYMBAL ; ETHNIC ;
EFFECTS;ACCENT;ORCH;RAP;ADDITION
```

と指定すればよいことになります。これで

F:¥ADPCM_LIB¥BASS, F:¥ADPCM_LIB¥SNARE, F:¥ADPCM LIB¥TOMTOM,
F:¥ADPCM_LIB¥CYMBAL, F:¥ADPCM_LIB¥ETHNIC, F:¥ADPCM LIB¥EFFECTS,
F:¥ADPCM_LIB¥ACCENT, F:¥ADPCM_LIB¥ORCH, F:¥ADPCM LIB¥RAP,
F:¥ADPCM_LIB¥ADDITION

を検索するようになります。通常の表記との混在ももちろん可能です。

設定例

```
SET  zmusic=A:¥ZPD_DATA;A:¥PCMFILES;F:¥BOSPCM;/0F:
¥ADPCM_LIB;/1F:¥MUSIC_DATA;
SET  zmusic0= BASS ; SNARE ; TOMTOM ; CYMBAL ; ETHNIC ;
EFFECTS;ACCENT;ORCH;RAP;ADDITION
SET zmusic1=SC55_DATA;CM64_DATA;INTERNAL_DATA;
```

また,補助環境変数zmusic0〜zmusic9をカテゴリに分類して検索パスを設定すれば簡潔にデータ格納ディレクトリ検索パスを記述することができます。上はzmusic0をADPCMライブラリ,zmusic1を曲データのライブラリとして管理した例です。

MEASURE 3
X-BASIC用外部関数MUSICZ.FNC

X68000本体付属のBASIC「X-BASIC」からZ-MUSICを使う方法について述べます。

3.1. はじめに

X-BASICからZ-MUSICを使うには外部関数「MUSICZ.FNC」を組み込んだX-BASICを起動しなければなりません。具体的には，BASICディレクトリ中のBASIC.CNFの内容のうち，

FUNC＝MUSIC

の部分を，

FUNC＝MUSICZ

のように変更し，さらにこのディレクトリにMUSICZ.FNCをコピーします。そして，

A>basic

でX-BASICを起動すればいいのです。しかし，各自のシステムの環境によっては多少の食い違いもあるので，よくわからない人は「BASICマニュアル」を読みましょう。また，ディスク1を起動している人は，

A>basic

でMUSICZ.FNCを組み込んだX-BASICを立ち上げることができます。

また，このZ-MUSICシステムは1990年10月号のOh!Xに発表された「ZMUSIC.FNC」とはまったく無関係で互換性もありません。

MEASURE1で述べたように，ZMUSIC.Xは一部を除いてX68000本体付属の「OPMDRV.X」と「MUSIC.FNC」との互換性を保っています。このことからMUSICZ.FNCのコマンドの一部はMUSIC.FNCのものと仕様が同じになっています。そのため，ここでは「BASICマニュアル」にあるコマンドも重複して説明がなされています。しかし，MUSICZ.FNCでは仕様が拡張変更されたものもあるので注意してください。

また，従来のMUSIC.FNCでもZ-MUSICは操作可能です。しかし，もちろんZMUSIC.Xの性能を100%発揮することはできません。また，MUSICZ.FNCでOPMDRV.Xは操作できないので注意してください。

3.2. コマンドの使用にあたっての注意

MIDI楽器専用のコマンドは，場合によっては楽器のメモリを書き替えます。楽器側のマニュアルをよく読んでから実行するようにしましょう。楽器側に大切なデータが存在する場合は楽器のメモリの内容をファイルに保存しておきましょう(MEASURE9参照)。

3.3. MUSICZ.FNC命令一覧

●初期化

m_init()

機能　音源の初期化，ドライバのワークの初期化
引数　なし
戻り値　なし
備考　MIDIボード装着時MIDI楽器に対して以下のコマンドメッセージを送出する。

システムリセット($FF)
リセットオールコントローラーズ($Bn,$79,$00)
オムニモードオン($Bn,7D,$00)
モノモードオフ($Bn,$7F,$00)
ローカルオン($Bn,$7A,$7F)
マスターチューン微調整＝中央
($Bn,$65,$00,$Bn,$64,$01,$Bn,$06,$40,$Bn,$26,$00)
マスターチューンコース＝中央
($Bn,$65,$00,$Bn,$64,$02,$BN,$06,$40)

m_count(n)

機能　全音符の絶対音長指定
引数　n＝全音符の絶対音長(char：1≦n≦254)
戻り値　なし
備考　'm_init()'でデフォルト値である192が設定される。通常は設定する必要なし。ゲーム用のBGMを作成するときなど，割り込み回数を減らす目的で使うとよい。192，144，128，96といった数値が一般的。

●トラック確保/チャンネルアサイン

m_alloc(tr,size)

機能　トラックバッファの確保
引数　tr：トラック番号(char：1～80)
size：バッファサイズ(int：100～65535)
戻り値　なし

m_assign(ch,tr)

機能　チャンネル番号chをトラック番号trに割り当てる
引数　ch：チャンネル番号(char：1～32)
tr：トラック番号(char：1～80)
戻り値　なし
備考　複数のトラックをひとつのチャンネルに割り当てることもできる。チャンネル番号は後述の'm_ch()'コマンドによって対象デバイスが変化する。ただし，PCM8.X組み込み時は'm_ch()'によらずチャンネル番号26～32がADPCM2～8になる。

m_assign2(ch,tr)

機能　チャンネル番号chをトラック番号trに割り当てる
引数　ch：チャンネル番号
(str："FM1～8"，"MIDI1～16"，"ADPCM"，"ADPCM1～8")
tr：トラック番号(char：1～80)
戻り値　なし
備考　ひとつのチャンネルを複数のトラックに割り当てることができる。
例　m_assign2(1,"FM1")
m_assign2(1,"MIDI10")
"ADPCM2～8"が実用できるのはPCM8モード時のみである。

m_ch(dv)

機能　ベースチャンネルを変更する
引数　dv：デバイス名(str："FM" or "MIDI")
戻り値　なし
備考　"FM"を設定するとチャンネル番号が1～8はFM音源，9がAD PCMチャンネル1，10～25がMIDIになる。"MIDI"を設定するとチャンネル番号1～16がMIDI，17～24がFM音源，25がAD PCMチャンネル1になる。
また，PCM8.X組み込み時は'm_ch()'によらず，チャンネル番号26～32がADPCM2～8になる。
例　m_ch("MIDI")
m_ch("FM")

m_free(tr)

機能　トラックバッファのフリーエリアを得る
引数　tr＝トラック番号(char：1～80)
戻り値　int

●演奏制御

m_play(tr_1,tr_2,…,tr_{10})

機能　演奏の開始
引数　tr1～10：トラック番号(char：1～80)
すべて省略すると全トラックを演奏する
戻り値　なし

m_stop(tr1,tr2,…,tr10)

機能　演奏の停止
引数　tr1～10:トラック番号(char:1～80)
　すべて省略すると全トラックの演奏を停止する
戻り値　なし

m_cont(tr1,tr2,…,tr10)

機能　演奏の再開
引数　tr1～10:トラック番号(char:1～80)
　すべて省略すると全トラックの演奏を再開する
戻り値　なし

m_solo(ch1,ch2,…ch10)

機能　演奏チャンネルのマスク（その1）
引数　ch1～10＝チャンネル番号（char：1～32)
戻り値　なし
備考　指定したチャンネルのみを演奏しそれ以外のチャンネル演奏を一時停止する。演奏中に実行可能。またパラメータを省略し，
　'm_solo()'
とすると通常演奏に戻る。
　m_ch("FM")のときはチャンネル番号1～8がFM音源，9がADPCMチャンネル1，10～25がMIDIに対応する。
　m_ch("MIDI")のときはチャンネル番号1～16がMIDI，17～24がFM音源，25がAD PCMチャンネル1に対応する
　また，PCM8.X組み込み時は'm_ch()'によらず，チャンネル番号26～32がADPCM2～8に対応する。
　演奏が停止したトラックも内部処理は続行するため，マスクを解除したときは，ほかのトラックに遅れをとらずになにごともなかったように演奏が再開される。
　ただし，このコマンド実行時は効果音モードが正常に動作しない。

m_mute(ch1,ch2,…ch10)

機能　演奏チャンネルのマスク（その2）
引数　ch1～10＝チャンネル番号（char：1～32)
戻り値　なし
備考　指定したチャンネルの演奏を一時停止し，それ以外のチャンネルは通常に演奏を行う。演奏中に実行可能。またパラメータを省略し，
　'm_mute()'
とすると通常演奏に戻る。
　m_ch("FM")のときはチャンネル番号1～8がFM音源，9がADPCMチャンネル1，10～25がMIDIに対応する。
　m_ch("MIDI")のときはチャンネル番号1～16がMIDI，17～24がFM音源，25がAD PCMチャンネル1に対応する。
　また，PCM8.X組み込み時は'm_ch()'によらず，チャンネル番号26～32がADPCM2～8に対応する。
　演奏が停止したトラックも内部処理は続行するため，マスクを解除したときは，ほかのトラックに遅れをとらずになにごともなかったように演奏が再開される。
　ただし，このコマンド実行時は効果音モードが正常に動作しない。

m_fadeout(spd)

機能　演奏中のデータをフェードアウト/インする
引数　spd＝フェードアウト/インスピード(int:－85～85)
戻り値　なし
備考　spdを省略するとデフォルト値16が設定される。
　spdは正値がフェードアウト，負値がフェードインの指定となる。絶対値が大きいほど音量の増減スピードは速くなる。

m_stat(ch)

機能　演奏状態のチェック
引数　ch＝チャンネル番号(char:1～25)　(注1)
備考　すべて省略すると全チャンネルの演奏状態をチェックできる。
戻り値　int
　チャンネル番号を指定した場合，そのチャンネルが演奏中ならば1，演奏中でないなら0。チャンネル番号を省略した場合は，演奏中のチャンネル番号に対応したビットが1に，演奏していないチャンネル番号に対応したビットは0になる。
　m_ch("FM")の場合はビット0～7がFMチャンネル1～8に，ビット8がAD PCMチャンネル1に，ビット9～24がMIDIチャンネル1～16に対応する。
　m_ch("MIDI")の場合はビット0～15がMIDIチャンネル1～16に，ビット16～23がFMチャンネル1～8に，ビット24がAD PCMチャンネル1に対応する。
　また，PCM8.Xの組み込み時には'm_ch()'によらず，ビット25～31が2～8チャンネルに対応する。
注1)　チャンネル番号は'm_ch()'コマンドで対応するデバイスが変動する。'm_ch()'コマンドの項を参照。

●開発補助/その他

m_debug(sw)

機能　MMLの'[@]','[!]','[end]'コマンドの有効/無効化
引数　sw＝数値(char:0＝off,0≠on)
戻り値　なし

m_total()

機能　各トラックにセットされたMMLの総ステップタイム数を計算し，結果を画面に出力する
引数　なし
戻り値　なし
備考　ステップタイムデータは，
　トラック番号：ループ外の総カウント数　ループ内の総カウント数
というフォーマットで出力される。X-BASICは独自の仮想画面で画面を管理しているのに対して本命令はむりやり表示を行っているため，画面下部で本命令を実行すると表示の一部がずれることがある。

zmd_play(fn)

機能　コンパイルされたデータの演奏
引数　fn＝ファイル名(str)
戻り値　なし
備考　拡張子を省略した場合は自動的に'.ZMD'を付け足す。fnがカレントより見つからない場合は環境変数'zmusic'のパスを参照する。

m_pcmplay(nt,pn,fr)

機能　登録してあるノート番号ntのAD PCM音を鳴らす
引数　nt＝ノート番号(int:0～511)
　pn＝パンポット(char:0～3)
　fr＝再生周波数(char:0～4)
戻り値　なし
備考　fr:　0 ＝ 3.9kHz
　1 ＝ 5.2kHz
　2 ＝ 7.8kHz
　3 ＝ 10.4kHz
　4 ＝ 15.6kHz
例　m_pcmplay(10,3,4)

m_switch(sw,fn)

機能　ZMSファイル（OPMファイル）のジェネレートスイッチ
引数　sw＝スイッチ(char:0＝off,　1＝on)
　fn＝ファイルネーム(str)
戻り値　なし
備考　fnを省略するとデフォルトファイル名として'ZMUSIC.ZMS'がカレントに作成される。
　なお本命令はステートメントでなくコマンドであるため，プログラム中に書くことはできない。書いた場合はエラーとなる。

fm_ver()

機能　ZMUSIC.XとPCM8.Xの常駐チェック
引数　なし
戻り値　ドライバの常駐状態を返す

上位16ビット＝0　　PCM8は常駐していない
上位16ビット＝－1　PCM8が常駐している
下位16ビット＝0　　ZMUSICは常駐していない
下位16ビット≠0　　ZMUSICが常駐している

備考　ZMUSIC.Xが常駐している場合にはそのバージョンIDを返す。

上位バイト　バージョン整数部，バージョン小数部第1桁
下位バイト　対応バージョンコード，バージョン小数部第2桁
対応バージョンコード
0　UNIVERSAL VERSION
1　16bit VERSION
2　RS-MIDI VERSION
3　POLYPHON VERSION

例　POLYPHON ver.1.49　→　バージョンID＝$1439

fm_master(v)

機能　FM音源のトラックのマスターボリューム
引数　v:ボリューム値(char:最小0～最大255)
戻り値　なし
備考　コンパイル時や中間言語変換時にのみ有効なコマンドなので，演奏中に指定しても無意味。用途としてはAD PCM，MIDI楽器との最終的な音量バランスを決定するときなど。通常は演奏データよりも先に書くとよいだろう。'm_init()'でデフォルトとして255が設定される。
増減は線形でなく指数関数的であるので注意のこと。

m_wave_form(wv,lt,lp,ary)

機能　波形メモリの登録
引数　wv＝登録先波形番号(char: 8～31)
lt＝ループタイプ(char: 0～2)
lp＝ループポイント(int: 0～65535)
ary＝波形データ(dim int ary(0～65535)={.....})
戻り値　なし
備考　波形番号8～31に登録できる(波形番号0～7はプリセット波形などが設定されておりリザーブ)。ループタイプとは波形を最後まで処理し終えたあと，どうループさせるかを設定するもの。

0 →　　ワンショット
(波形を一度実行したら最後の値を継続する)
1 →→→→....　リピート
(波形の終点までいったらループポイントに戻る)
2 →←→←....　オルタネイティブ
(波形のループポイントから終点まで交互に反復する)

ループポイントは0～65535までが有効。これは何番目のデータをループポイントに設定するかを決めるパラメータで省略するとループポイントは0，すなわちデータの先頭がループポイントとみなされる。
波形データは符号付き16ビット整数で構成する(－32768～＋32767($8000～$7fff))。データの個数は65535個まで。それ以上は設定できない。
波形メモリの具体的な使用方法についてはMEASURE5参照。

m_print(ms)

機能　単なる文字表示
引数　ms＝文字列(str:96文字以内)
戻り値　なし

exec_zms(zms)

機能　ZMSコマンドの実行
引数　zms＝ZMSコマンド文字列(str)
戻り値　なし

adpcm_to_pcm(adpcm_data_buffer,size,pcm_data_buffer)

機能　AD PCMデータを符号つき16ビットPCMデータへ変換する
引数　adpcm_data_buffer:入力AD PCMデータ格納バッファ
(dim char ary(0～65535))
size:入力AD PCMデータの個数(int:0～65535)
pcm_data_buffer:出力PCMデータ格納用バッファ
(dim int ary(0～65535))
戻り値　なし
備考　出力PCMデータ格納用配列のサイズは入力AD PCMデータ格納配列のサイズの2倍以上必要。

pcm_to_adpcm(pcm_data_buffer,size,adpcm_data_buffer)

機能　符号つき16ビットPCMデータをAD PCMデータへ変換する
引数　pcm_data_buffer:入力PCMデータ格納バッファ
(dim int ary(0～65535))
size:入力PCMデータの個数(int:0～65535)
adpcm_data_buffer:出力AD PCMデータ格納用バッファ
(dim int ary(0～65535))
戻り値　なし
備考　出力AD PCMデータ格納用配列のサイズは入力PCMデータ格納配列のサイズの半分以上は必要。

●テンポ

m_tempo(tm)

機能　テンポの設定
引数　tm＝トラック番号(int:20～300)
戻り値　int (tmを省略すると現在のテンポ値を返す)
備考　タイマAモード(MEASURE2参照)で動作中時は，テンポ77以下は強制的に77になる。

●MMLの書き込み

m_trk(tr,str)

機能　MMLをトラックバッファへ書き込む
引数　tr:トラック番号(char: 1～80)
str:MMLデータ(str)
戻り値　なし
備考　MMLに関してはMEASURE5を参照のこと。

m_trk2(str,tr1,tr2,tr3,tr4,tr5,tr6,tr7,tr8)

機能　MMLを複数のトラックバッファへ書き込む
引数　str:MMLデータ(str)
tr1～8:トラック番号(char:1～80)
戻り値　なし
備考　tr2～tr8は省略可能。MMLに関してはMEASURE5を参照のこと。

●FM音源の音色設定

m_vset(vn,ary)

機能　FM音源の音色登録
引数　vn:音色番号(char: 1～200)
ary:音色データ(dim char ary(4,10))
戻り値　なし
備考

```
10 /* 設定例
20 dim char v(4,10)
30 /*  AF  OM  WF SYC SPD PMD AMD PMS AMS PAN
40 v={ 59,  15,  2,  1, 200, 127,  0,  0,  0,  3,  0,
50 /*  AR  DR  SR  RR  SL  OL  KS  ML DT1 DT2 AME
60      31,  8,  1,  8,  7, 20,  2,  1,  5,  3,  0,
```

```
 70     31,  8,  8,  7,  5, 24,  1,  2,  1,  1,  0,
 80     31,  3,  7,  8,  1, 21,  1,  1,  3,  0,  0,
 90     31,  0,  0,  9,  0,  0,  2,  8,  5,  2,  0}
100 m_vset(1,v)
```

パラメータ概説

	0		1～4
(オペレータ)			
0	AF(フィードバック/アルゴリズム)	(0～63)	AR(アタックレート) (0～31)
1	OM(スロットマスク)	(0～15)	1DR(1stディケイレート) (0～31)
2	WF(ウェーヴフォーム)	(0～3)	2DR(2ndディケイレート) (0～31)
3	SYC(シンクロ)	(0,1)	RR(リリースレート) (0～15)
4	SPD(スピード)	(0～255)	1DL(1stディケイレベル) (0～15)
5	PMD	(0～127)	TL(トータルレベル) (0～127)
6	AMD	(0～127)	KS(キースケーリング) (0～3)
7	PMS	(0～7)	MUL(フェーズマルチプライ) (0～15)
8	AMS	(0～3)	DT1(ディチューン1) (0～7)
9	PAN	(0～3)	DT2(ディチューン2) (0～3)
10	DUMMY		AME(AMSイネーブル) (0,1)

パラメータの機能や詳しい解説は専門書や「BASICマニュアル」のM_VSET()やM_VGET()の項を参照。

m_vget(vn,ary)

機能　FM音源の音色データの取り出し
引数　vn:音色番号(char:1～200)
　　ary:音色バッファ(dim char ary(4,10))
戻り値　ary(4,10)に音色データが格納される。

m_fmvset(vn,ary)

機能　FM音源の音色登録(AL/FB分離形式)
引数　vn:音色番号(char:1～200)
　　ary:音色データ(dim char ary(4,10))
戻り値　なし
例

```
10 dim char v(4,10)
20 /*  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME
30 v={ 31,  0,  2,  0,  0, 21,  0,  1,  0,  0,  0,
40     31,  0,  0,  8,  0,  3,  0,  3,  0,  0,  0,
50     31,  0,  0,  8,  0,  3,  0,  1,  0,  0,  0,
60     31,  0,  0,  8,  0,  3,  0,  1,  0,  0,  0,
70 /* CON  FBL   OM PAN   WF  SYC  SPD PMD AMD PMS AMS
80      5,   7,  15,  3,  0,  0,  0,  0,  0,  0,  0}
90 m_fmvset(1,v)
```

備考　AL/FB(CON/FB)分離形式とはFM音源のパラメータであるアルゴリズム(またはコネクション)とフィードバックレベルを従来の書式ではひとつにまとめていたのをわかりやすく分割して指定できるようにしたもの。

●AD PCMコンフィギュレーション

m_pcmset(nt,fn,pt,vl,mx,dl,ct,rv,fd)

機能　ノート番号ntにファイル名fnのAD PCM音をセットする
引数　nt=ノート番号(int:0～511)
　　fn=ファイル名(str)
　　pt=ピッチシフトパラメータ
　　(int:0～11=ピッチダウン，12=ニュートラル，
　　13～24=ピッチアップ)
　　vl=ボリュームパラメータ(int:1%～300%)
　　mx=ミックス対象ノート番号(int:0～511)
　　dl=ミックスディレイパラメータ (int:0～65535)
　　ct=カットパラメータ(int:0～$ffffffff)
　　rv=リバーススイッチ(char:0=normal/1=reverse)
　　fd=フェードイン/アウトパラメータ(int:0～$ffffffff)
戻り値　なし
備考　カットパラメータ=オフセット
　　(0～65535)×65536+カットサイズ(0～65535)
　　フェードイン/アウトパラメータ
　　=オフセット(0～65535)×65536
　　+モード(－1=フェードイン，+1=フェードアウト)×256
　　+フェードイン/アウトレベル(0～127)
　　fnがカレントより見つからない場合は環境変数'zmusic'のパスを参照する。
　　pt,vl,mx,dl,ct,rv,fdはそれぞれ省略可能。
　　pt,vl,mx,dl,ct,rv,fdのいずれかを設定した場合はAD PCMデータの加工を行うので多少の時間を要する。
　　また，加工には相当量のメモリを必要とするためドライバ起動時の'-W'スイッチでワークエリアを十分に確保していないとエラーとなる。
　　パラメータの具体的な説明はMEASURE6を参照のこと。
例　m_pcmset(10,"snare.pcm",13,120,8,1000,&h10001000,1,&h1000ff01)

m_pcmcnf(fn)

機能　AD PCMコンフィギュレーションファイルで，AD PCM音をドライバに登録する
引数　fn=ファイル名(str)
戻り値　なし
備考　fnがカレントより見つからない場合は環境変数'zmusic'のパスを参照する。

m_adpcm_block(fn)

機能　AD PCMブロックデータを読み込む
引数　fn=ファイル名(str)
戻り値　なし
備考　fnがカレントより見つからない場合は環境変数'zmusic'のパスを参照する。AD PCMバッファがブロックデータの大きさより小さい場合はエラーとなる。

●MIDIデータ出力

m_out(data1,data2,…,data10)

機能　MIDIデータを送信する
引数　data1～10=MIDIデータ(char)
戻り値　なし
備考　data2以降は省略可，X-BASICの制約からデータは10個まで。
例　m_out(&HC0,10)

m_dirout(ary,size)

機能　1次元配列に格納されたデータを送信する
引数　ary=1次元配列(dim char ary(n))
　　size=送信データ数(int:1～65536)
戻り値　なし
備考　sizeを省略した場合は配列内すべてのデータを送信する。

m_exc(ary,size)

機能　1次元配列に格納されたデータをエクスクルーシブデータとして送信する
引数　ary=1次元配列(dim char ary(n))
　　size=送信データ数(int:1～65536)
戻り値　なし
備考　sizeを省略した場合は配列内すべてのデータを送信する。
　　頭に$f0,後尾に$f7を自動的に付けて送信するだけで機能的にはほとんど'm_dirout()'と同じ。

m_roland(dev,mdl,ary,size)

機能　1次元配列に格納されたデータをRolandエクスクルーシブデータとみなし送信する
引数　dev=転送先の楽器のデバイスID(char)

mdl＝転送先の楽器のモデルID(char)
ary＝1次元配列(dim char ary(n))
size＝送信データ数(int：1～65536)

戻り値　なし

備考　sizeを省略した場合は配列内すべてのデータを送信する。エクスクルーシブヘッダやチェックサムのデータは自動発信する。

m_trans(fn)

機能　MIDIダンプファイル（MDD）をMIDI楽器へ送信する

引数　fn＝ファイル名(str)

戻り値　なし

備考　fnがカレントより見つからない場合は，環境変数'zmusic'のパスを参照する。MEASURE9参照。

●MIDIデータ入力

m_inp(mode)

機能　1バイトのMIDIデータを受信する

引数　mode＝入力モード(char：0,1～255)
mode＝0
入力データがない場合はエラーコードを持って帰還
mode＝1～255
データが受信されるまでmode×(1/60)秒間待つ

戻り値　0～255：入力されたデータ
-1：エラー

m_rec()

機能　MIDI楽器よりデータを受信するモードにする

引数　なし

戻り値　なし

備考　MEASURE9参照。

m_rstop()

機能　MIDI楽器からのデータを受信するモードを解除する

引数　なし

戻り値　なし

備考　MEASURE9参照。

m_save(fn)

機能　'm_rec()'で受信したMIDIデータをセーブする

引数　fn=ファイル名(str)

戻り値　なし

備考　MEASURE9参照。

●SC-55/CM-300専用命令

パラメータの意味などは楽器のマニュアルを参照してください。

sc55_init(id)

機能　SC-55の初期化を行う

引数　id＝SC-55のデバイスID(char)

戻り値　なし

備考　IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。

sc55_v_reserve(ary,id)

機能　SC-55のボイスリザーブ

引数　ary＝各パートのボイスリザーブ値
(dim char ary(15)＝{vr1,…,vr16})
id＝SC-55のデバイスID(char)

戻り値　なし

備考　配列は必ず1次元，添え字は15でリザーブ値は16パート分書かなければならない。

ary(0)＝パート1，ary(1)＝パート2…，ary(15)＝パート16というふうに対応する。

IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。

sc55_reverb(ary,id)

機能　SC-55のリバーブパラメータ設定

引数　ary＝リバーブパラメータ
(dim char ary(6)＝{rv1,…,rv7})
id＝SC-55のデバイスID(char)

戻り値　なし

備考　配列は必ず1次元，添え字は6以内で配列の内容全部がSC-55へ送信される。

IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。

ary(0)：REVERB MACRO 00：Room1,01：Room2,02：Room3,03：Hall1,04：Hall2
05：Plate,06：Delay,07：Panning Delay(FB)

ary(1)：REVERB CHARACTER	($00～$07)
ary(2)：REVERB PRE-LPF	($00～$07)
ary(3)：REVERB LEVEL	($00～$7F)
ary(4)：REVERB TIME	($00～$7F)
ary(5)：REVERB DELAY FEEDBACK	($00～$7F)
ary(6)：REVERB SEND LEVEL TO CHORUS	($00～$7F)

(SC-55マニュアルp79参照)

sc55_chorus(ary,id)

機能　SC-55のコーラスパラメータ設定

引数　ary＝コーラスパラメータ
(dim char ary(7)＝{co1,…,co8})
id＝SC-55のデバイスID(char)

戻り値　なし

備考　配列は必ず1次元，添え字は7以内で配列の内容全部がSC-55へ送信される。

IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。

ary(0)：CHORUS MACRO 00：Chorus1,01：Chorus2,02：Chorus3,03：Chorus4
04：Feedback Chorus,05：Flanger,06：Short Delay,07：Short Delay(FB)

ary(1)：CHORUS PRE-LPF	($00～$07)
ary(2)：CHORUS LEVEL	($00～$7F)
ary(3)：CHORUS FEEDBACK	($00～$7F)
ary(4)：CHORUS DELAY	($00～$7F)
ary(5)：CHORUS RATE	($00～$7F)
ary(6)：CHORUS DEPTH	($00～$7F)
ary(7)：CHORUS SEND LEVEL TO REVERB	($00～$7F)

(SC-55マニュアルp79参照)

sc55_part_setup(pt,ary,id)

機能　SC-55のパートパラメータ設定

引数　pt＝パートナンバー(char：1～16)
ary＝パートパラメータ
(dim char ary(118)＝{pp1,…,pp119})
id＝SC-55のデバイスID(char)

戻り値　なし

備考　配列は必ず1次元，添え字は118以内で配列の内容全部がSC-55へ送信される。

IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合は，ドライバ内のデフォルト値が選択される。

ary(0)：Rx CHANNEL	(1～16,17＝OFF)
ary(1)：Rx PITCH BEND	(0～1：OFF/ON)
ary(2)：Rx CH PRESSURE(CAf)	(0～1：OFF/ON)
ary(3)：Rx PROGRAM CHANGE	(0～1：OFF/ON)
ary(4)：Rx CONTROL CHANGE	(0～1：OFF/ON)

ary(5):Rx POLY PRESSURE(PAf) (0~1:OFF/ON)
ary(6):Rx NOTE MESSAGE (0~1:OFF/ON)
ary(7):Rx RPN (0~1:OFF/ON)
ary(8):Rx NRPN (0~1:OFF/ON)
ary(9):Rx MODURATION (0~1:OFF/ON)
ary(10):Rx VOLUME (0~1:OFF/ON)
ary(11):Rx PANPOT (0~1:OFF/ON)
ary(12):Rx EXPRESSION (0~1:OFF/ON)
ary(13):Rx HOLD1 (0~1:OFF/ON)
ary(14):Rx PORTAMENTO (0~1:OFF/ON)
ary(15):Rx SOSTENUTO (0~1:OFF/ON)
ary(16):Rx SOFT (0~1:OFF/ON)
ary(17):MONO/POLY MODE (0~1:MONO/POLY)
ary(18):ASSIGN MODE (0:SINGLE,1:LIMITED-MULTI,2:FULL-MULTI)
ary(19):USE FOR RHYTHM PART (0:OFF,1:MAP1,2:MAP2)
ary(20):PITCH KEY SHIFT ($28~$58:-24~+24[semitone])
ary(21):PITCH OFFSET FINE(upper) ($08~$F8:-12.0~+12.0[Hz])
ary(22):PITCH OFFSET FINE(lower) 〃
ary(23):PART LEVEL (0~127)
ary(24):VELOCITY SENSE DEPTH (0~127)
ary(25):VELOCITY SENSE OFFSET (0~127)
ary(26):PART PANPOT (0:RANDOM,01(LEFT)~64~127(RIGHT))
ary(27):KEY RANGE LOW (0~127:C-1~G9)
ary(28):KEY RANGE HIGH (0~127:C-1~G9)
ary(29):CC1 CONTROL NUMBER (0~127)
ary(30):CC2 CONTROL NUMBER (0~127)
ary(31):CHORUS SEND DEPTH (0~127)
ary(32):REVERB SEND DEPTH (0~127)
ary(33):Vibrato rate ($0E~$72:-50~+50)
ary(34):Vibrato depth ($0E~$72:-50~+50)
ary(35):TVF cutoff freq. ($0E~$50:-50~+16)
ary(36):TVF resonance ($0E~$72:-50~+50)
ary(37):TVF&TVA Env.attack ($0E~$72:-50~+50)
ary(38):TVF&TVA Env.decay ($0E~$72:-50~+50)
ary(39):TVF&TVA Env.release ($0E~$72:-50~+50)
ary(40):Vibrato delay ($0E~$72:-50~+50)
ary(41):SCALE TUNING C ($00~$7F:-64~+63[cent])
ary(42):SCALE TUNING C♯ ($00~$7F:-64~+63[cent])
ary(43):SCALE TUNING D ($00~$7F:-64~+63[cent])
ary(44):SCALE TUNING D♯ ($00~$7F:-64~+63[cent])
ary(45):SCALE TUNING E ($00~$7F:-64~+63[cent])
ary(46):SCALE TUNING F ($00~$7F:-64~+63[cent])
ary(47):SCALE TUNING F♯ ($00~$7F:-64~+63[cent])
ary(48):SCALE TUNING G ($00~$7F:-64~+63[cent])
ary(49):SCALE TUNING G♯ ($00~$7F:-64~+63[cent])
ary(50):SCALE TUNING A ($00~$7F:-64~+63[cent])
ary(51):SCALE TUNING A♯ ($00~$7F:-64~+63[cent])
ary(52):SCALE TUNING B ($00~$7F:-64~+63[cent])
ary(53):MOD PITCH CONTROL ($28~$58:-24~+24[semitone])
ary(54):MOD TVF CUTOFF CONTROL ($00~$7F:-9600~+9600[CENT])
ary(55):MOD AMPLITUDE CONTROL ($00~$7F:-100.0~+100.0[%])
ary(56):MOD LFO1 RATE CONTROL ($00~$7F:-10.0~+10.0[Hz])
ary(57):MOD LFO1 PITCH DEPTH ($00~$7F:0~600[cent])
ary(58):MOD LFO1 TVF DEPTH ($00~$7F:0~2400[cent])
ary(59):MOD LFO1 TVA DEPTH ($00~$7F:0~100.0[%])
ary(60):MOD LFO2 RATE CONTROL ($00~$7F:-10.0~+10.0[Hz])
ary(61):MOD LFO2 PITCH DEPTH ($00~$7F:0~600[cent])
ary(62):MOD LFO2 TVF DEPTH ($00~$7F:0~2400[cent])
ary(63):MOD LFO2 TVA DEPTH ($00~$7F:0~100.0[%])
ary(64):BEND PITCH CONTROL ($28~$58:-24~+24[semitone])
ary(65):BEND TVF CUTOFF CONTROL ($00~$7F:-9600~+9600[CENT])
ary(66):BEND AMPLITUDE CONTROL ($00~$7F:-100.0~+100.0[%])
ary(67):BEND LFO1 RATE CONTROL ($00~$7F:-10.0~+10.0[Hz])
ary(68):BEND LFO1 PITCH DEPTH ($00~$7F:0~600[cent])
ary(69):BEND LFO1 TVF DEPTH ($00~$7F:0~2400[cent])
ary(70):BEND LFO1 TVA DEPTH ($00~$7F:0~100.0[%])
ary(71):BEND LFO2 RATE CONTROL ($00~$7F:-10.0~+10.0[Hz])
ary(72):BEND LFO2 PITCH DEPTH ($00~$7F:0~600[cent])
ary(73):BEND LFO2 TVF DEPTH ($00~$7F:0~2400[cent])
ary(74):BEND LFO2 TVA DEPTH ($00~$7F:0~100.0[%])
ary(75):CAf PITCH CONTROL ($28~$58:-24~+24[semitone])
ary(76):CAf TVF CUTOFF CONTROL ($00~$7F:-9600~+9600[CENT])
ary(77):CAf AMPLITUDE CONTROL ($00~$7F:-100.0~+100.0[%])
ary(78):CAf LFO1 RATE CONTROL ($00~$7F:-10.0~+10.0[Hz])
ary(79):CAf LFO1 PITCH DEPTH ($00~$7F:0~600[cent])
ary(80):CAf LFO1 TVF DEPTH ($00~$7F:0~2400[cent])
ary(81):CAf LFO1 TVA DEPTH ($00~$7F:0~100.0[%])
ary(82):CAf LFO2 RATE CONTROL ($00~$7F:-10.0~+10.0[Hz])
ary(83):CAf LFO2 PITCH DEPTH ($00~$7F:0~600[cent])
ary(84):CAf LFO2 TVF DEPTH ($00~$7F:0~2400[cent])
ary(85):CAf LFO2 TVA DEPTH ($00~$7F:0~100.0[%])
ary(86):PAf PITCH CONTROL ($28~$58:-24~+24[semitone])
ary(87):PAf TVF CUTOFF CONTROL ($00~$7F:-9600~+9600[CENT])
ary(88):PAf AMPLITUDE CONTROL ($00~$7F:-100.0~+100.0[%])
ary(89):PAf LFO1 RATE CONTROL ($00~$7F:-10.0~+10.0[Hz])
ary(90):PAf LFO1 PITCH DEPTH ($00~$7F:0~600[cent])
ary(91):PAf LFO1 TVF DEPTH ($00~$7F:0~2400[cent])
ary(92):PAf LFO1 TVA DEPTH ($00~$7F:0~100.0[%])
ary(93):PAf LFO2 RATE CONTROL ($00~$7F:-10.0~+10.0[Hz])
ary(94):PAf LFO2 PITCH DEPTH ($00~$7F:0~600[cent])
ary(95):PAf LFO2 TVF DEPTH ($00~$7F:0~2400[cent])
ary(96):PAf LFO2 TVA DEPTH ($00~$7F:0~100.0[%])
ary(97):CC1 PITCH CONTROL ($28~$58:-24~+24[semitone])
ary(98):CC1 TVF CUTOFF CONTROL ($00~$7F:-9600~+9600[CENT])
ary(99):CC1 AMPLITUDE CONTROL ($00~$7F:-100.0~+100.0[%])
ary(100):CC1 LFO1 RATE CONTROL ($00~$7F:-10.0~+10.0[Hz])
ary(101):CC1 LFO1 PITCH DEPTH ($00~$7F:0~600[cent])
ary(102):CC1 LFO1 TVF DEPTH ($00~$7F:0~2400[cent])
ary(103):CC1 LFO1 TVA DEPTH ($00~$7F:0~100.0[%])
ary(104):CC1 LFO2 RATE CONTROL ($00~$7F:-10.0~+10.0[Hz])
ary(105):CC1 LFO2 PITCH DEPTH ($00~$7F:0~600[cent])
ary(106):CC1 LFO2 TVF DEPTH ($00~$7F:0~2400[cent])
ary(107):CC1 LFO2 TVA DEPTH ($00~$7F:0~100.0[%])
ary(108):CC2 PITCH CONTROL ($28~$58:-24~+24[semitone])
ary(109):CC2 TVF CUTOFF CONTROL ($00~$7F:-9600~+9600[CENT])
ary(110):CC2 AMPLITUDE CONTROL ($00~$7F:-100.0~+100.0[%])
ary(111):CC2 LFO1 RATE CONTROL ($00~$7F:-10.0~+10.0[Hz])
ary(112):CC2 LFO1 PITCH DEPTH ($00~$7F:0~600[cent])
ary(113):CC2 LFO1 TVF DEPTH ($00~$7F:0~2400[cent])
ary(114):CC2 LFO1 TVA DEPTH ($00~$7F:0~100.0[%])
ary(115):CC2 LFO2 RATE CONTROL ($00~$7F:-10.0~+10.0[Hz])
ary(116):CC2 LFO2 PITCH DEPTH ($00~$7F:0~600[cent])
ary(117):CC2 LFO2 TVF DEPTH ($00~$7F:0~2400[cent])
ary(118):CC2 LFO2 TVA DEPTH ($00~$7F:0~100.0[%])

(SC-55マニュアルp75,p79~80参照)

sc55_drum_setup(mp,nt,ary,id)

機能 SC-55のドラムパラメータ設定

引数 mp=マップナンバー(char:0~1)
nt=ノート番号(char:0~127)
ary=ドラムパラメータ(dim char ary(7)={dr1,…,dr8})
id=SC-55のデバイスID(char)

戻り値 なし

備考 配列は必ず1次元,添え字は7以内で配列の内容全部がSC-55へ送信される。IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。

ary(0):PLAY KEY NUMBER (0~127)

ary(1):LEVEL (0～127)
ary(2):ASSIGN GROUP NUMBER (0:NON,1～127)
ary(3):PANPOT
(0:RANDOM,01(LEFT)～64～127(RIGHT))
ary(4):REVERB DEPTH (0～127:0.0～1.0)
ary(5):CHORUS DEPTH (0～127:0.0～1.0)
ary(6):Rx NOTE OFF (0～1:OFF/ON)
ary(7):Rx NOTE ON (0～1:OFF/ON)
(SC-55マニュアルp75,p82参照)

sc55_print(ms,id)

機能 SC-55のディスプレイに文字列を表示する
引数 ms＝文字列(str:32文字以内)
id＝SC-55のデバイスID(char)
戻り値 なし
備考 IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。

sc55_display(ary,id)

機能 SC-55のディスプレイにドットパターン(16×16)を表示する
引数 ary＝ドットパターンデータ
(dim int a(15)＝{dt1,…,dt16})
id＝SC-55のデバイスID(char)
戻り値 なし
備考 配列は必ず1次元，int型で添え字は必ず15でなければならない。IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。
例

```
 5 /*SC-55の画面に"善"を表示する
10 dim int d(a5)={ %0001000000010000,
20                 %0000100000100000,
30                 %0111111111111100,
40                 %0000000100000000,
50                 %0011111111111000,
60                 %0000000100000000,
70                 %0111111111111100,
80                 %0001000100010000,
90                 %0000100100100000,
100                %1111111111111110,
110                %0000000000000000,
120                %0011111111111000,
130                %0010000000001000,
140                %0010000000001000,
150                %0011111111111000,
160                %0010000000001000}
170 sc55_display(d,&h10)
```

●MT-32/CM-32L専用命令

パラメータの意味などは楽器のマニュアルを参照してください

mt32_init(id)

機能 MT-32の初期化を行う
引数 id＝MT-32のデバイスID(char)
戻り値 なし
備考 IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。
注意 MT-32用の命令はCM-32L/CM-64/CM-500にも使用できる。以下同様

mt32_p_reserve(ary,id)

機能 MT-32のパーシャルリザーブを行う
引数 ary＝各パートのパーシャルリザーブ値
(dim char ary(8)＝{pr1,……,pr9})
id＝MT-32のデバイスID(char)
戻り値 なし
備考 配列は必ず1次元，char型で添え字は必ず8でなければならない。9番目のデータ(添え字で8番目)はリズムパートに相当する。IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。

mt32_reverb(ary,id)

機能 MT-32のリバーブパラメータの設定
引数 ary＝リバーブパラメータ
(dim char ary(2)＝{pr1,pr2,pr3})
id＝MT-32のデバイスID(char)
戻り値 なし
備考 配列は必ず1次元，添え字は2以下で配列の内容全部がMT-32へ送信される。IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。
ary(0):REVERB MODE
(0:Room,1:Hall,2:Plate,3:Tap Delay)
ary(1):REVERB TIME (0～7)
ary(2):REVERB LEVEL (0～7)
(MT-32マニュアルp35,CM-64マニュアルp30参照)

mt32_part_setup(ary,id)

機能 MT-32の各パートのMIDIチャンネルの設定
引数 ary＝各パートのMIDIチャンネル値 (1～16,17:OFF)
(dim char ary(8)＝{pr1,……,pr9})
id＝MT-32のデバイスID(char)
戻り値 なし
備考 配列は必ず1次元，添え字は8以下で配列の内容全部がMT-32へ送信される。9番目のデータ(添え字で8番目)はリズムパートに相当する。IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。

mt32_drum_setup(nt,ary,id)

機能 MT-32のリズムパートの設定
引数 nt＝設定を変更するリズムノートナンバー(char:24～87)
ary＝リズムパートのセットアップパラメータ
(dim char ary(3)＝{pr1,……,pr4})
id＝MT-32のデバイスID(char)
戻り値 なし
備考 配列は必ず1次元，添え字は3以下で配列の内容全部がMT-32へ送信される。IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。
ary(0):TIMBRE (0～63:i1～64,64～127:r1～64)
ary(1):OUTPUT LEVEL (0～100)
ary(2):PANPOT (0～14)
ary(3):REVERB SWITCH (0～1:OFF/ON)
(MT-32マニュアルp35,CM-64マニュアルp30参照)

mt32_common(tm,nm,ary,id)

機能 MT-32のティンバーコモンパラメータの設定
引数 tm＝ティンバー番号(char:1～64)
nm＝ティンバー名(str:10文字以内)
ary＝コモンパラメータ
(dim char ary(3)＝{pr1,……,pr4})
id＝MT-32のデバイスID(char)
戻り値 なし

備考　配列は必ず1次元，添え字は3以下で配列の内容全部がMT-32へ送信される。IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。

ary(0)：Structure of Partial#1&2　(0～12：1～13)
ary(1)：Structure of Partial#3&4　(0～12：1～13)
ary(2)：PARTIAL MUTE　(0～15)
ary(3)：ENV MODE　(0～1：NORMAL，NO SUSTAIN)
(MT-32マニュアルp34，CM-64マニュアルp29参照)

mt32_patch(pt,ary,id)

機能　MT-32のパッチパラメータの設定
引数　pt＝パッチ(program)番号(char：1～128)
ary＝パッチパラメータ(dim char ary(6)={pr1,……,pr7})
id＝MT-32デバイスID(char)
戻り値　なし
備考　配列は必ず1次元，添え字は6以下で配列の内容全部がMT-32へ送信される。IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。

ary(0)：TIMBRE GROUP　(0～3：a，b，i，r)
ary(1)：TIMBRE NUMBER　(0～63)
ary(2)：KEY SHIFT　(0～48：－24～＋24)
ary(3)：FINE TUNE　(0～100：－50～＋50)
ary(4)：BENDER RANGE　(0～24)
ary(5)：ASSIGN MODE　(0～3：POLY1～4)
ary(6)：REVERB SWITCH　(0～1：OFF/ON)
(MT-32マニュアルp35，CM-64マニュアルp30参照)

mt32_partial(tm,pt,ary,id)

機能　MT-32の音色のパーシャルパラメータの設定
引数　tm＝ティンバー番号(char：1～64)
pt＝パーシャル番号(char：1～4)
ary＝パッチパラメータ
(dim char ary(57)＝{pr1,……,pr58})
id＝MT-32のデバイスID(char)
戻り値　なし
備考　配列は必ず1次元，添え字は57以下で配列の内容全部がMT-32へ送信される。IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。

ary(0)：WG PITCH COARSE　(0～96：C1，C#1，～，C9)
ary(1)：WG PITCH FINE　(0～100：－50～＋50)
ary(2)：WG PITCH KEYFOLLOW
(0～16：－1，－1/2，－1/4，0，1/8，1/4，3/8，1/2，5/8，3/4，7/8，1，5/4，3/2，2，s1，s2)
ary(3)：WG PITCH BENDER SW　(0～1：OFF/ON)
ary(4)：WG WAVEFORM/PCM BANK　(0～3：SQU1，SAW1，SQU2，SAW2)
ary(5)：WG PCM WAVE　(0～127：1～128)
ary(6)：WG PULSE WIDTH　(0～100)
ary(7)：WG PW VELO SENS　(0～14：－7～＋7)
ary(8)：P-ENV DEPTH　(0～10)
ary(9)：P-ENV VELO SENS　(0～3)
ary(10)：P-ENV TIME KEYF　(0～4)
ary(11)：P-ENV TIME 1　(0～100)
ary(12)：P-ENV TIME 2　(0～100)
ary(13)：P-ENV TIME 3　(0～100)
ary(14)：P-ENV TIME 4　(0～100)
ary(15)：P-ENV LEVEL 0　(0～100：－50～＋50)
ary(16)：P-ENV LEVEL 1　(0～100：－50～＋50))
ary(17)：P-ENV LEVEL 2　(0～100：－50～＋50)
ary(18)：P-ENV SUSTAIN LEVEL　(0～100：－50～＋50)
ary(19)：END LEVEL　(0～100：－50～＋50)
ary(20)：P-LFO RATE　(0～100)
ary(21)：P-LFO DEPTH(0～100)
ary(22)：P-LFO MOD SENS　(0～100)
ary(23)：TVF CUTOFF FREQ　(0～100)
ary(24)：TVF RESONANCE　(0～30)
ary(25)：TVF KEYFOLLOW　(0～14：－1，－1/2，－1/4，0，1/8，1/4，3/8，1/2，5/8，3/4，7/8，1，5/4，3/2，2)
ary(26)：TVF BIAS POINT/DIR　(0～127)
ary(27)：TVF BIAS LEVEL　(0～14：－7～＋7)
ary(28)：TVF ENV DEPTH　(0～100)
ary(29)：TVF ENV VELO SENS　(0～100)
ary(30)：TVF ENV DEPTH KEYF　(0～4)
ary(31)：TVF ENV TIME KEYF　(0～4)
ary(32)：TVF ENV TIME 1　(0～100)
ary(33)：TVF ENV TIME 2　(0～100)
ary(34)：TVF ENV TIME 3　(0～100)
ary(35)：TVF ENV TIME 4　(0～100)
ary(36)：TVF ENV TIME 5　(0～100)
ary(37)：TVF ENV LEVEL 1　(0～100)
ary(38)：TVF ENV LEVEL 2　(0～100)
ary(39)：TVF ENV LEVEL 3　(0～100)
ary(40)：TVF ENV SUSTAIN LEVEL　(0～100)
ary(41)：TVA LEVEL　(0～100)
ary(42)：TVA VELO SENS　(0～100：－50～＋50)
ary(43)：TVA BIAS POINT 1　(0～127)
ary(44)：TVA BIAS LEVEL 1　(0～12：－12～0)
ary(45)：TVA BIAS POINT 2　(0～127)
ary(46)：TVA BIAS LEVEL 2　(0～12：－12～0)
ary(47)：TVA ENV TIME KEYF　(0～4)
ary(48)：TVA ENV TIME V_FOLLOW　(0～4)
ary(49)：TVA ENV TIME 1　(0～100)
ary(50)：TVA ENV TIME 2　(0～100)
ary(51)：TVA ENV TIME 3　(0～100)
ary(52)：TVA ENV TIME 4　(0～100)
ary(53)：TVA ENV TIME 5　(0～100)
ary(54)：TVA ENV LEVEL 1　(0～100)
ary(55)：TVA ENV LEVEL 2　(0～100)
ary(56)：TVA ENV LEVEL 3　(0～100)
ary(57)：TVA ENV SUSTAIN LEVEL　(0～100)
(MT-32マニュアルp34，CM-64マニュアルp29参照)

mt32_print(ms,id)

機能　MT-32のコンソールに文字列を表示する
引数　ms＝文字列(str：20文字以内)
id＝MT-32のデバイスID(char)
戻り値　なし
備考　IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。

●U220/U20専用命令

パラメータの意味などは楽器のマニュアルを参照してください

u220_setup(ary,id)

機能　U220のセットアップパラメータの設定
引数　ary＝パラメータ(dim char ary(6)＝{pr1,……,pr7})
id＝U220のデバイスID(char)
戻り値　なし
備考　配列は必ず1次元，char型で添え字は必ず6でなければならない。IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。

ary(0)：Chorus SW　(0～1：OFF/ON)
ary(1)：Reverb SW　(0～1：OFF/ON)

ary(2):Rx Cntrl Channel (1～17:1～16,OFF)
ary(3):Patch Change (0～5:MAP1～MAP4,Dir,OFF)
ary(4):Timbre Change (0～5:MAP1～MAP4,Dir,OFF)
ary(5):Rhythm Change (0～5:MAP1～MAP4,Dir,OFF)
ary(6):R.Inst Assign (0～5:MAP1～MAP4,Dir,OFF)
(U220マニュアルp46,p146～148参照)

注意 U220用の命令はU20に対しても使用できる。以下同様。

u220_part_setup(pt,ary,id)

機能 U220のテンポラリパートセットアップパラメータの設定

引数 pt＝パート番号(char:1～6)
ary＝パラメータ(dim char ary(12)＝{pr1,……,pr13})
id＝U220のデバイスID(char)

戻り値 なし

備考 配列は必ず1次元，char型で添え字は必ず12でなければならない。IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。

ary(0):Timbre Number (1～128)
ary(1):Voice Reserve (0～30)
ary(2):Receive Channel (1～17:1～16,OFF)
ary(3):Key Range Low (0～127:C-1～G9)
ary(4):Key Range High (0～127:C-1～G9)
ary(5):Velo Level (0～1:Above/Below)
ary(6):Velo Threshold (0～127)
ary(7):Output Assign (0～4:Dry,Rev,Cho,Dir1,Dir2)
ary(8):Level (0～127)
ary(9):Pan (0～15:L7～M～R7)
ary(10):Rx Volume (0～1:OFF/ON)
ary(11):Rx Pan (0～1:OFF/ON)
ary(12):Rx Hold (0～1:OFF/ON)
(U220マニュアルp58,p149参照)

u220_common(ary,id)

機能 U220のテンポラリパッチコモンパラメータの設定

引数 ary＝パラメータ(dim char ary(17)＝{pr1,……,pr18})
id＝U220のデバイスID(char)

戻り値 なし

備考 配列は必ず1次元，char型で添え字は必ず17でなければならない。IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。

ary(0):Chorus Type (0～4:Chorus1,Chorus2,FB-Chorus,
Flanger,Short Delay)
ary(1):Chorus Out Mode (0～1:Pre Rev,Post Rev)
ary(2):Chorus Level (0～31)
ary(3):Chorus Delayl (0～31)
ary(4):Chorus Ratel (0～31)
ary(5):Chorus Depthl (0～31)
ary(6):Chorus Feedbackl (1～63:－31～0～＋31)
ary(7):Reverb Type (0～7:Room1～3,Hall1～2,Gate
Delay,CrossDelay)
ary(8):Reverb Timel (0～31)
ary(9):Reverb Levell (0～31)
ary(10):Reverb Delay Feedbackl (0～31)
ary(11):Reverb Pre Delay Feedback (0～31)
ary(12):Param1 # (0～63:0～5,7～31,64～95,OFF)
ary(13):Param1 Param (0:Timbre Level,1:Env Attack
2:Env Decay,3:Env Sustain
4:Env Release,5:A.Bend Depth
6:A.Bend Rate,7:Detune Depth
8:Vib Rate,9:Vib Wave Form
10:Vib Depth,11:Vib Delay
12:Vib Rise Time,13:Vib Mod Depth
14:Chorus Level,15:Chorus Rate
16:Chorus Feedback,17:Reverb Level,
18:Delay Feedback)
ary(14):Param2 # (Same as Param1 #)
ary(15):Param2 Param (Same as Param1 Param)
ary(16):Param3 # (Same as Param1 #)
ary(17):Param3 Param (Same as Param1 Param)
(U220マニュアルp54,p149参照)

u220_timbre(tm,name,ary,id)

機能 U220のティンバーパラメータの設定

引数 tm＝ティンバー番号(char: 1～128)
name＝ティンバーの名前(str:12文字以内)
ary＝パラメータ(dim char ary(25)＝{pr1,……,pr26})
id＝U220のデバイスID(char)

戻り値 なし

備考 配列は必ず1次元，char型で添え字は必ず25でなければならない。IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。

ary(0):Tone Media (0～31:I,1～31)
ary(1):Tone Number (1～128)
ary(2):Timbre Level (0～127)
ary(3):Velocity Sens (1～15:－7～＋7)
ary(4):Channel Press Sens (1～15:－7～＋7)
ary(5):Env Attack Rate (1～15:－7～＋7)
ary(6):Env Decay Rate (1～15:－7～＋7)
ary(7):Env Sustain Level (1～15:－7～＋7)
ary(8):Env Release Rate (1～15:－7～＋7)
ary(9):Pitch Shift Coarse (8～56:－24～＋24)
ary(10):Pitch Shift Fine (14～114:－50～＋50)
ary(11):Bend Range Lower (0～15:－36,－24,－12～0)
ary(12):Bend Range Upper (0～12)
ary(13):Channel After Sens (0～27:－36,-24,-12～＋12)
ary(14):Poly After Sens (0～27:－36,－24,－12～＋12)
ary(15):Auto Bend Depth (0～27:－36,－24,－12～＋12)
ary(16):Auto Bend Rate (0～15)
ary(17):Detune Depth (0～15)
ary(18):Rate (0～63)
ary(19):Waveform (0～8)
ary(20):Depth (0～15)
ary(21):Delay (0～15)
ary(22):Rise Time (0～15)
ary(23):Modulation Depth (0～15)
ary(24):Ch After Sens (0～15)
ary(25):Poly After Sens (0～15)
(U220マニュアルp5,p49参照)

u220_drum_setup(ary,id)

機能 U220のテンポラリドラムセットアップパラメータの設定

引数 ary＝パラメータ(dim char ary(6)＝{pr1,……,pr7})
id＝U220のデバイスID(char)

戻り値 なし

備考 配列は必ず1次元，char型で添え字は必ず6でなければならない。

IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。

ary(0):Rhythm Setup # (0～3:1～4)
ary(1):Voice Reserve (0～30)
ary(2):Receive Channel (1～17:1～16,OFF)
ary(3):Level (0～127)
ary(4):Level Boost Sw (0～1:OFF/ON)
ary(5):Rx Volume (0～1:OFF/ON)
ary(6):Rx Hold (0～1:OFF/ON)

(U220マニュアルp63,p149参照)

u220_drum_inst(nt,ary,id)

機能　U220のドラムインストパラメータの設定

引数　nt＝ノートナンバー(char:35～99)
ary＝パラメータ(dim char ary(19)＝{pr1,……,pr20})
id＝U220のデバイスID(char)

戻り値　なし

備考　配列は必ず1次元，添え字は19以下で配列の内容全部がU220へ送信される。

IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。

ary(0):Tone Media	(0～31:I,1～31)
ary(1):Tone Number	(0～127:1～128)
ary(2):Source Key	(0～127:C-1～G9)
ary(3):Mute Inst	(34～98:OFF,B1～D7)
ary(4):Inst Level	(0～31)
ary(5):Velocity Sens	(0～15)
ary(6):Env Mode	(0～1:Sustain,No Sustain)
ary(7):Env Attack Rate	(1～15:－7～＋7)
ary(8):Env Decay Rate	(1～15:－7～＋7)
ary(9):Env Release Rate	(1～15:－7～＋7)
ary(10):Pitch Shift Coarse	(0～27:-36,-24,-12～＋12)
ary(11):Pitch Shift Fine	(14～114:－50～＋50)
ary(12):Channel After Sens	(0～27:-36,-24,-12～＋12)
ary(13):Poly After Sens	(0～27:－36,－24,－12～＋12)
ary(14):Random	(0～15)
ary(15):Auto Bend Depth	(0～27:－36,－24,－12～＋12)
ary(16):Auto Bend Rate	(0～15)
ary(17):Detune Depth	(0～15)
ary(18):Output Assign	(0～3:Dry,Rev,Cho,Dir1)
ary(19):Pan	(0～15:L7～M～R7)

(U220マニュアルp72,p149参照)

u220_print(ms,id)

機能　U220のコンソールに文字列を表示する

引数　ms＝文字列(str:12文字以内)
id＝U220のデバイスID(char)

戻り値　なし

備考　IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。

注2)　M1専用命令は途中でM1関係以外の命令が入ると正常なデータが楽器側へ送信されないので注意。

●M1/M1 EX/M1R/M1R EX専用命令

パラメータの意味などは楽器のマニュアルを参照してください。

m1_midi_ch(ary)　(注2)

機能　M1のシーケンサのSONG0の各トラックのMIDIチャンネルを設定する

引数　ary＝各パートのチャンネル値(1～16,17:OFF)
(dim char ary(7)＝{ch1,……,ch8})

戻り値　なし

備考　配列は必ず1次元，char型で添え字は必ず7でなければならない。

注意)　M1用の命令はM1 EX/M1R/M1R EXに対しても使用できる。以下同様。

m1_part_setup(ary)　(注2)

機能　M1のシーケンサのSONG0の各トラックのパラメータを設定する

引数　ary＝各パートのパラメータ
(dim char ary(39)＝{pr1,……,pr40})

戻り値　なし

備考　配列は必ず1次元，char型で添え字は必ず39でなければならない。

TRACK 1

ary(0):PROGRAM NUMBER	(0～199:I00～C99)
ary(1):OUTPUT LEVEL	(0～99)
ary(2):KEY TRANSPOSE	($F4～$0C:－12～12)
ary(3):DETUNE	($CE～$32:－50～＋50)
ary(4):PAN	(0～13:10:0～0:10,C,C＋D,D)
ary(5～39):TRACK 2～8	

以下同様
(M1マニュアルp126参照)

m1_effect_setup(ary)　(注2)

機能　M1のシーケンサのSONG0のエフェクトパラメータを設定する

引数　ary＝エフェクトパラメータ
(dim char ary(24)＝{pr1,……,pr25})

戻り値　なし

備考　配列は必ず1次元，char型で添え字は必ず24でなければならない。

ary(0):Effect 1 Pattern No.	(0～33:1～32,Thru)
ary(1):Effect 2 Pattern No.	(0～33:1～32,Thru)
ary(2):Effect 1 L-Ch E.Balnc	(0～100)
ary(3):Effect 1 R-Ch E.Balnc	(0～100)
ary(4):Effect 2 L-Ch E.Balnc	(0～100)
ary(5):Effect 2 R-Ch E.Balnc	(0～100)
ary(6):Output 3 Pan	(0～101:OFF,100:0～0:100)
ary(7):Output 4 Pan	(0～101:OFF,100:0～0:100)
ary(8):Effect I/O	(BIT0:Effect 1 L-Ch 0=OFF/1=ON
	BIT1:Effect 1 R-Ch 0=OFF/1=ON
	BIT2:Effect 2 L-Ch 0=OFF/1=ON
	BIT3:Effect 2 R-Ch 0=OFF/1=ON)
	BIT4:Effect 2 0=PARALLEL/1=SERIAL)
ary(9～16):Effect 1 Parameter	
ary(17～24):Effect 2 Parameter	

Structure of Effect Parameter　(ofs＝9 or 17)

1～3:Hall (4,5:Room,6:Live Stage)

ary(ofs＋0):Reverb Time	(0～97:0.2～9.9),(0～48:0.2～5.0)
ary(ofs＋1):Dummy	0
ary(ofs＋2):High Damp	(0～99)
ary(ofs＋3):Pre Delay	(0～200)
ary(ofs＋4):E/R Level	(0～99)
ary(ofs＋5):Dummy	0
ary(ofs＋6):EQ High	($F4～$0C:－12～＋12)
ary(ofs＋7):EQ Low	($F4～$0C:－12～＋12)

7～9:Early Reflection 1,2,3

ary(ofs＋0):E/R Time	(0～70:100～800)
ary(ofs＋1):Pre Delay	(0～200)
ary(ofs＋2):Dummy	0
:	:
ary(ofs＋5):Dummy	0
ary(ofs＋6):EQ High	($F4～$0C:－12～＋12)
ary(ofs＋7):EQ Low	($F4～$0C:－12～＋12)

10:Stereo Delayt,11:Cross Delay

ary(ofs＋0):Delay Time L (L)	(0～500)
ary(ofs＋1):Delay Time L (H)	
ary(ofs＋2):Feedback	($9D～$63:－99～99)
ary(ofs＋3):High Damp	(0～99)
ary(ofs＋4):Delay Time R (L)	(0～500)
ary(ofs＋5):Delay Time R (H)	
ary(ofs＋6):EQ High	($F4～$0C:－12～＋12)

ary(ofs+7):EQ Low ($F4~$0C:−12~+12)
12~13:Stereo Chorus 1~2 (14~15:Flanger)
ary(ofs+0):Depth (0~99)
ary(ofs+1):Speed (0~99:0.03~3.00 0.03step
100~199:3.1~13.0 0.1step
200~216:14.0~30.0 1.0step)
ary(ofs+2):LFO Status (BIT0:Waveform=0:Sin, =1:Tri
BIT1:Phase=0:0.0° , =1:180°
BIT2:Wave Shape=0:Normal
=1: Flanger)
ary(ofs+3):Feedback ($9D~$63:−99~+99)
ary(ofs+4):Delay Time (0~200), (0~50)
ary(ofs+5):Dummy 0
ary(ofs+6):EQ High ($F4~$0C:−12~+12)
ary(ofs+7):EQ Low ($F4~$0C:−12~+12)
16~17:Phase Shifter 1~2
ary(ofs+0):Depth (0~99)
ary(ofs+1):Speed (0~99:0.03~3.00 0.03step
100~199:3.1~13.0 0.1step
200~216:14.0~30.0 1.0step)
ary(ofs+2):LFO Status (BIT0:Waveform=0:Sin, =1:Tri
BIT1:Phase=0:0.0° , =1:180°
BIT2:Wave Shape=0:Normal
=1:for Flanger)
ary(ofs+3):Feedback ($9D~$63:−99~+99)
ary(ofs+4):Manual (0~99)
ary(ofs+5):Dummy 0
ary(ofs+6):Dummy 0
ary(ofs+7):Dummy 0
18~19:Stereo Toremolo 1~2
ary(ofs+0):Depth (0~99)
ary(ofs+1):Speed (0~99:0.03~3.00 0.03step
100~199:3.1~13.0 0.1step
200~216:14.0~30.0 1.0step)
ary(ofs+2):LFO Status (BIT0:Waveform=0:Sin, =1:Tri
BIT1:Phase=0:0.0° , =1:180°
BIT2:Wave Shape=0:Normal
=1:for Flanger)
ary(ofs+3):Shape ($9D~$63:−99~+99)
ary(ofs+4):Dummy 0
ary(ofs+5):Dummy 0
ary(ofs+6):EQ High ($F4~$0C:−12~+12)
ary(ofs+7):EQ Low ($F4~$0C:−12~+12)
20:Equalizer
ary(ofs+0):Dummy 0
: :
ary(ofs+3):Dummy 0
ary(ofs+4):Low fc (0~2:0.25k,0.50k,1.00k)
ary(ofs+5):High fc (0~2:1k,2k,4k)
ary(ofs+6):High Gain ($F4~$0C:−12~+12)
ary(ofs+7):Low Gain ($F4~$0C:−12~+12)
21:Overdrive
ary(ofs+0):Dummy 0
ary(ofs+1):Dummy 0
ary(ofs+2):Drive (0~99)
ary(ofs+3):Level (0~99)
ary(ofs+4):Dummy 0
ary(ofs+5):Dummy 0
ary(ofs+6):EQ High ($F4~$0C:−12~+12)
ary(ofs+7):EQ Low ($F4~$0C:−12~+12)
22:Distortion
ary(ofs+0):Dummy 0
ary(ofs+1):Dummy 0
ary(ofs+2):Distortion (0~99)
ary(ofs+3):Level (0~99)
ary(ofs+4):Dummy 0
ary(ofs+5):Dummy 0
ary(ofs+6):Dummy 0
ary(ofs+7):EQ Low Gain ($F4~$0C:−12~+12)
23:Exciter
ary(ofs+0):Blend ($9D~$63:−99~+99)
ary(ofs+1):Emphatic Point (0~9:1~10)
ary(ofs+2):Dummy 0
: :
ary(ofs+5):Dummy 0
ary(ofs+6):EQ High ($F4~$0C:−12~+12)
ary(ofs+7):EQ Low ($F4~$0C:−12~+12)
24:Synphonic Ensamble
ary(ofs+0):Depth (0~99)
ary(ofs+1):Dummy 0
: :
ary(ofs+5):Dummy 0
ary(ofs+6):EQ High ($F4~$0C:−12~+12)
ary(ofs+7):EQ Low ($F4~$0C:−12~+12)
25:Rotary Speaker
ary(ofs+0):Depth (0~99)
ary(ofs+1):Dummy 0
ary(ofs+2):Speed Rate ($F6~$0A:−10~+10)
ary(ofs+3):Dummy 0
: :
ary(ofs+7):Dummy 0
26:Delay/Hall
ary(ofs+0):Delay Time (L) (0~500)
ary(ofs+1):Delay Time (H)
ary(ofs+2):Feedback ($9D~$63:−99~99)
ary(ofs+3):High Damp (0~99)
ary(ofs+4):Reverb Time (0~97:0.2~9.9)
ary(ofs+5):Dummy 0
ary(ofs+6):High Damp (0~99)
ary(ofs+7):Pre Delay (0~150)
27:Delay/Room
ary(ofs+0):Delay Time (L) (0~500)
ary(ofs+1):Delay Time (H)
ary(ofs+2):Feedback ($9D~$63:−99~99)
ary(ofs+3):High Damp (0~99)
ary(ofs+4):Reverb Time (0~97:0.2~9.9)
ary(ofs+5):Dummy 0
ary(ofs+6):High Damp (0~99)
ary(ofs+7):Pre Delay (0~150)
28:Delay/Early Reflection
ary(ofs+0):Delay Time (L) (0~500)
ary(ofs+1):Delay Time (H)
ary(ofs+2):Feedback ($9D~$63:−99~99)
ary(ofs+3):High Damp (0~99)
ary(ofs+4):E/R Time (0~30:100~400)
ary(ofs+5):Pre Delay (0~150)
ary(ofs+6):Dummy 0
ary(ofs+7):Dummy 0
29:Delay/Chorus
ary(ofs+0):Delay Time L (L) (0~500)
ary(ofs+1):Delay Time L (H)
ary(ofs+2):Feedback L ($9D~$63:−99~99)
ary(ofs+3):High Damp L (0~99)
ary(ofs+4):Delay Time R (L) (0~500)
ary(ofs+5):Delay Time R (H)
ary(ofs+6):Feedback R ($9D~$63:−99~99)
ary(ofs+7):High Damp R (0~99)
30:Delay/Chorus (31:Delay Flanger)

```
    ary(ofs+0):Delay Time (L)   (0～500)
    ary(ofs+1):Delay Time (H)
    ary(ofs+2):Feedback         ($9D～$63:－99～99)
    ary(ofs+3):High Damp        (0～99)
    ary(ofs+4):Depth            (0～99)
    ary(ofs+5):Speed            (0～99:0.03～3.00 0.03step
                                100～199:3.1～13.0 0.1step
                                200～216:14.0～30.0 1.0step)
    ary(ofs+6):LFO Status
                        (BIT0:Waveform=0:Sin,=1:Tri
                         BIT1:Phase=0:0.0°  ,=1:180°
                          BIT2:Wave Shape=0:Normal
                                         =1:for Flanger)
    ary(ofs+7):Feedback         0,($9D～$63:－99～99)
32:Delay/Phaser
    ary(ofs+0):Delay Time (L)   (0～500)
    ary(ofs+1):Delay Time (H)
    ary(ofs+2):Feedback         ($9D～$63:－99～99)
    ary(ofs+3):High Damp        (0～99)
    ary(ofs+4):Depth            (0～99)
    ary(ofs+5):Speed            (0～99:0.03～3.00 0.03step
    ary(ofs+6):Feedback         ($9D～$63:－99～99)
    ary(ofs+7):Dummy            0
33:Delay/Tremolo
    ary(ofs+0):Delay Time (L)   (0～500)
    ary(ofs+1):Delay Time (H)
    ary(ofs+2):Feedback         ($9D～$63:－99～99)
    ary(ofs+3):High Damp        (0～99)
    ary(ofs+4):Depth            (0～99)
    ary(ofs+5):Speed            (0～99:0.03～3.00 0.03step
    ary(ofs+6):Dummy            0
    ary(ofs+7):Shape            ($9D～$63:－99～99)
    (M1マニュアルp127参照)
```

m1_print(ms)

機能　M1のシーケンサのSONG0の名前を設定する
引数　ms＝文字列(str:10文字以内)
戻り値　なし

send_to_m1(id)

機能　'm1_midi_ch()'～'m1_print()'までの設定値をM1へ送信する
引数　id＝MIDIのデバイスID値(char:&h30～&h3f)
戻り値　なし
備考　デバイスID＝&H30＋（グローバルチャンネル－1）
グローバルチャンネルとは,GLOBALモードのF5-1で設定できる。
IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。
必ず'm1_midi_ch()'～'m1_print()'を設定してからこの命令を実行すること。
'm1_effect_setup()'のみ省略が可能。このときはZ-MUSICのデフォルトデータが送信される。
'm1_midi_ch()'～'m1_print()'の間にM1関係以外のコマンドがあると正常なデータが送信されないので注意すること。
例

```
10 dim char a(7)={2,1,4,3,6,5,8,7}
20 m1_midi_ch(a)
25 /*音色,音量,キートランスポーズ,ディチューン,パンのフォーマット
30 dim char b(39)={  0,99,0,0,5,   /*TRACK 1
40                   0,99,0,0,5,   /*TRACK 2
50                   0,99,0,0,5,   /*TRACK 3
60                   0,99,0,0,5,   /*TRACK 4
70                   0,99,0,0,5,   /*TRACK 5
80                   0,99,0,0,5,   /*TRACK 6
90                   0,99,0,0,5,   /*TRACK 7
100                  0,99,0,0,5}   /*TRACK 8
110 m1_part_setup(b)
120 dim char c(24)={&H21,&H21}
130 m1_effect_setup(c)
140 m1_print("Zenji.N")
150 send_to_m1(&H30)
```

MEASURE 4
ZMSコマンド

ここではCOMMAND.X上から音楽プログラムを書くための書式，つまりZMSファイル（OPMファイル）の文法について解説します。

4.1. ZMSファイルとは

ZMUSIC.Xではエディタから MMLやコマンドを記述したファイルを演奏することができます。これをZ-MUSICシステムでは「ZMSファイル」と呼びます。X68000本体に付属している「OPMDRV.X」用の「OPMファイル」と概念的にはほとんど同じです。

ZMSファイルはED.Xなどのスクリーンエディタを使い，ここで解説する「ZMSコマンド」を記述することによって作成していきます(ZMSコマンド(Tn)以外のトラックに依存しないZMSコマンドを特に「共通コマンド/コモンコマンド」と呼ぶ)。MUSICZ.FNCに揃っているコマンドは，ほとんどZMSコマンドで記述することができ，BASICのときと同じ感覚で曲を作ることができるでしょう。

X-BASIC上で記述した音楽プログラムをMUSICZ.FNCの'mswitch()'コマンドを使用してから実行すると，ZMSファイルを自動生成することができます(MEASURE3参照)。ZMSファイル形式はCOMMAND.Xレベルから手軽に演奏できたり，コンパイルすることもできて便利なので，X-BASICで作った曲をZMSファイルにコンバートして管理するのもよいでしょう。

4.2. コマンド解説を読むにあたって

省略してもよいパラメータはその旨を記述してありますが，それ以外は省略不可です。コマンドは大文字小文字どちらで記述してもかまいませんが，ここでは説明の都合上コマンド名を大文字，パラメータを小文字で記述しています。

'*'をつけたものはパラメータを複数行にわたって記述できます。それ以外のコマンドではその1行ですべてのパラメータを記述しなければならないことを意味します。

MUSICZ.FNCの外部関数命令と対応するものは，そのコマンド名を挙げてあります。詳しいパラメータ範囲などはそちらの説明を参照してください。

4.3. ZMSファイル用コマンド

●初期化

(I)

各種音源やZMUSIC.X本体の初期化を行う。

通常はZMSの先頭に書く。

MIDI楽器に対して初期化メッセージを送出する(具体的な送信メッセージ内容に関してはMEASURE3のm_init()の項を参照)。

ZMDデータとしては生成されないコマンドである。

(MEASURE3 m_init()参照)

(Zn)

全音符の絶対音長を設定する。設定範囲は $1 \leqq n \leqq 254$。

(I)コマンドでデフォルト値192が自動設定される。通常は設定する必要はない。

ゲームなどのBGMを制作する際，割り込みを軽くする目的で使うとよい。

192，144，128，96といった数値を設定するのが一般的。

(MEASURE3 m_count()参照)

●トラック確保/チャンネルアサイン

(Mtr,sz)

トラックtrにバッファサイズszバイト確保する。

$1 \leqq tr \leqq 80$，$100 \leqq sz \leqq 65535$。

例

(m2,3000)　　トラック2を3000バイト確保する

ZMDデータとしては生成されないコマンドである。

(MEASURE3 m_alloc()参照)

(Ach,tr)

チャンネルchをトラックtrへアサインする。

$1 \leqq ch \leqq 25$，$1 \leqq tr \leqq 80$。

PCM8モード時は26～32が使用可能。

チャンネル番号はベースチャンネル変更コマンドによって，その対象デバイスが変動する（(Bn)コマンド参照)。

chの部分に規定の文字列を指定することによって(Bn)コマンドの影響を受けずに絶対的に指定することも可能である。

FM1～8	FM音源チャンネル1～8
MIDI1～16	MIDIチャンネル1～16
ADPCM	AD PCM音源
ADPCM1～8	PCM8モード時

例

(a1,2)　　チャンネル1をトラック2へ割り当てる

(aFM1,3)　　FM音源チャンネル1をトラック3へ割り当てる

ZMDデータとしては生成されないコマンドである。

(MEASURE3 m_assign()，m_assign2()参照)

(Bn)

ベースチャンネルを設定する。

0＝FM基準　1＝MIDI基準

n＝0を設定するとチャンネル番号1～8がFM音源，9がAD PCM，10～25がMIDIになる。

n＝1を設定するとチャンネル番号1～16がMIDI，17～24がFM音源，25がAD PCMになる。

ただしPCM8モードのときは(Bn)の値によらずチャンネル番号26～32がAD PCMチャンネル2～8に対応する。

ZMDデータとしては生成されないコマンドである。

(MEASURE3 m_ch()参照)

●演奏制御

(Pn1,n2,…,ni)または(P)

演奏を開始する。n_i～はトラック番号で，$1 \leqq n_i \leqq 80$。全パラメータを省略して(P)のみならば，全トラックの演奏を開始する。

例

(p1,2,3)

ZMDデータ作成時にはまったく無視されるコマンド。

(MEASURE3 m_play()参照)

(Sn1,n2,…,ni)または(S)

演奏を停止する。n_i～はトラック番号で，$1 \leqq n_i \leqq 80$。全パラメータを省略して(S)のみならば，全トラックの演奏を停止する。

例

(s1,2,3)

ZMDデータ作成時にはまったく無視されるコマンドである。

(MEASURE3 m_stop()参照)

(Cn1,n2,…,ni)または(C)

演奏を再開する。n_i～はトラック番号で，$1 \leqq n_i \leqq 80$。全パラメータを省略して(C)のみならば，全トラックの演奏を再開する。

例

(c1,2,3)

ZMDデータ作成時にはまったく無視されるコマンドである。

(MEASURE3 m_cont()参照)

(Ech1,ch2,…,chi)

演奏中のチャンネルをリアルタイムにマスク/解除することができる。任意の個数のチャンネル番号を書くことによって指定以外のチャンネルをマスクすることができる。チャンネル番号は $1 \leqq ch_i \leqq 25$，ただしPCM8モード時は $1 \leqq ch_i \leqq 32$ となる。

(Bn),'m_ch()'によってチャンネル番号と対象デバイスは変動する点に注意。

後述の「ZP.R」のオプションスイッチ'-E'とまったく同様の機能。

例

(e1,2)　　チャンネル1，2以外をマスク

必ず(Mtr,size)，(Ach,tr)，'m_alloc()'，'m_assign()'以降に書くこと。トラック確保，チャンネルアサイン前に記述しても意味を持たない。(P)などの直前に書くのがスタンダードな使い方。

ZP -DとしてZP.Rを常駐させたあとに使用できる再演奏機能([SHIFT]+[XF4])を実行したときにも影響を与えるので，ひとつのチャンネルの演奏チェックを何度も行うときには便利。

ZMDデータとしては生成されないコマンドである。

(MEASURE3 m_solo()参照)

(Fn)

正数でフェードアウト，負数でフェードインを行うことができる。この指定を行った時点から，演奏中の全トラックがフェードイン/アウトし始める。ただし，1トラックでもすでにフェードイン/アウトしていた場合は無視される。

-85≦n≦-1　　フェードイン指定，絶対値が大きいほど音量増加スピードが速い

n=0　　フェードイン/アウトの解除

1≦n≦85　　フェードアウト指定，絶対値が大きいほど音量の減衰スピードが速い

ZMDデータ作成時にはまったく無視されるコマンドである。

(MEASURE3 m_fadeout()参照)

●開発補助/その他

(Dn)

[@]，[!]，[end]コマンドの有効/無効化スイッチ。

n=0で無効，n=1で有効となる。

ZMS中のみ有効で，演奏中に指定しても無意味。

ZMDデータとしては生成されないコマンドである。

(MEASURE3 m_debug()参照)

(Q)

各トラックにセットされたMMLの総ステップタイム数を計算し画面に出力する。

ZMDデータ作成時にはまったく無視されるコマンドである。

(MEASURE3 m_total()参照)

.WAVE_FORM wv,lt,lp {$dt_0,dt_1,dt_2,\cdots,dt_{65536}$}

波形メモリの登録。

wv：波形番号(8≦wv≦31)

lt：ループタイプ(0≦lt≦2)

lp：ループポイント(0≦lp≦65535)

$dt_0,dt_1,dt_2,\cdots,dt_{65535}$：波形データ(-32768≦$dt_i$≦32767)

備考　波形番号8～31に登録できる(波形番号0～7はプリセット波形などが設定されておりリザーブ)。

パラメータのループタイプとは波形を最後まで処理を終えたあと，どうループさせるかを設定するもの。

0 →　　ワンショット

(波形を一度実行したら最後の値を継続する)

1 →→→....　リピート

(終点までいったらループポイントに戻る)

2 →←→←....　オルタネイティブ

(ループポイントから終点まで交互に反復する)

ループポイントは0～65535までが有効。これは何番目のデータをループポイントに設定するかを決めるパラメータで省略するとループポイントは0，すなわちデータの先頭がループポイントとみなされる。波形データは符号付き16ビット整数で構成する(−32768～＋32767($8000～$7fff))。データの個数は65535個まで。それ以上は設定できない。

使用例

```
.wave_form 8,0,10 {0,5,-5,10,3,-300,10,6,80,10}
```

(MEASURE3 m_wave_form()参照，詳しい使い方はMEASURE5参照)

.FM_MASTER_VOLUME n

FM音源のマスターボリュームの設定。

設定範囲は0≦n≦255で，255が最大音量。通常は255。

コンパイル時のみ有効で，演奏中に指定しても無意味。

ZMDデータとしては生成されないコマンドである。

(MEASURE3 fm_master()参照)

/ 文字列

'/'より後ろをコメントとして無視する。

例

```
/ ORGAN SOLO
```

ZMDデータとしては生成されないコマンドである。

.COMMENT 文字列

後ろをコメントとして無視する。

単なるコメント行だが，'/'と違うのはコンパイルされたデータ(ZMDデータ)にも残るという点である(ただし演奏には影響なし)。曲のタイトル，制作日時，自分の名前などをZMSファイル先頭に記述するのが一般的。

例

```
.comment STAGE 1 BGM VERSION 1.10(c)1992/11 XVI
```

●テンポ

(On)

テンポnを設定する。

設定後，1分間にn個の4分音符を演奏するようになる。20≦n≦300。ただし，タイマAモードのときは77以下は強制的に77となる。

例

```
(o120)
```

(MEASURE3 m_tempo()参照)

●MMLの書き込み

(T n_1,n_2,・・・n8)

MMLをトラックnへ書き込む。1≦n_i≦80。n_2以降は省略可能。

m_trk(10,"abcde")は(T10)abcdeに相当する。

(MEASURE3 m_trk()参照)

●FM音源の音色設定

(Vn,0,v_1,v_2,・・・,v_{55})　　＊

FM音源の音色設定を行う。

nは定義する音色番号で1≦n≦200。

例

```
(v1,0
/        AF  OM  WF  SY  SP PMD AMD PMS AMS PAN
         60, 15,  2, 210, 40,  0,  2,  0,  3,  0
/        AR  DR  SR  RR  SL  OL  KS  ML DT1 DT2 AME
         31,  5,  0, 12,  2, 30,  1,  2,  7,  0,  0
         31,  5,  0, 12,  8,  6,  1,  2,  5,  0,  0
         31,  5,  0, 12,  8, 28,  1,  2,  3,  0,  0
         31,  5,  0, 12,  8,  6,  1,  2,  5,  0, 0)
```

(MEASURE3 m_vset()参照)

(@n,v_1,v_2,・・・,v_{55})　　＊

AL/FB分離形式によるFM音源の音色設定を行う。

nは定義する音色番号で1≦n≦200。

例

```
        /      AR  DR  SR  RR  SL  OL  KS  ML DT1 DT2 AME
 (@1,          31,  0,  2,  0,  0, 21,  0,  1,  0,  0,  0
               31,  0,  0,  8,  0,  3,  0,  3,  0,  0,  0
               31,  0,  0,  8,  0,  3,  0,  1,  0,  0,  0
               31,  0,  0,  8,  0,  3,  0,  1,  0,  0,  0
        /      AL  FB OM PAN  WF  SY  SP PMD AMD PMS AMS
                5,  7, 15,  3,  0,  0,  0,  0,  0,  0,  0)
```

(MEASURE3 m_fmvset()参照)

.FM_VSET n {v1,v2,…,v5} *

AL/FB分離形式によるFM音源の音色設定。

nは定義する音色番号で1≦n≦200。

機能とデータフォーマットはまったく(@)コマンドと同じである。

例

```
        /      AR  DR  SR  RR  SL  OL  KS  ML DT1 DT2 AME
 FM_VSET1 {31,  0,  2,  0,  0, 21,  0,  1,  0,  0,  0
               31,  0,  0,  8,  0,  3,  0,  3,  0,  0,  0
               31,  0,  0,  8,  0,  3,  0,  1,  0,  0,  0
               31,  0,  0,  8,  0,  3,  0,  1,  0,  0,  0
        /      AL  FB OM PAN  WF  SY  SP PMD AMD PMS AMS
                5,  7, 15,  3,  0,  0,  0,  0,  0,  0, 0}
```

(MEASURE3 m_fmvset()参照)

●AD PCMコンフィギュレーション

n ファイル名,Pp,Vv,Mm,d,Cc,s,R,Ff,l

AD PCM音の登録を行う。

n＝設定音色番号(0～511)
p＝ピッチシフトパラメータ(−12～12)
v＝ボリュームパラメータ(1～100(原音量)～300)
m＝ミキシングノート番号(0～511)
d＝ミキシングディレイパラメータ(0～65535)
c＝カットオフセットパラメータ(0～65535)
s＝カットサイズ(0～65535)
f＝フェードイン/アウトオフセット(0～65535)
l＝フェードイン/アウトレベル(0～127)

ファイル名以降は省略可能。詳しい使い方とパラメータの意味はMEASURE6参照。

(MEASURE3 m_pcmset()，MEASURE6参照)

.ADPCM_BANK n

AD PCM音の登録先のバンクを指定する。

n＝バンク番号(1≦n≦4)

'(I)'命令でデフォルト値1が設定される。

(MEASURE6参照)

.Onk ファイル名,Pp,Vv,Mm,d,Cc,s,R,Ff,l

AD PCM音の登録を行う。

n＝オクターブ値(-1～9)
k＝音階MML(abcdefg，#，+，-)
p＝ピッチシフトパラメータ(-12～12)
v＝ボリュームパラメータ(1～100(原音量)～300)
m＝ミキシングノート番号(0～511)
d＝ミキシングディレイパラメータ(0～65535)
c＝カットオフセットパラメータ(0～65535)
s＝カットサイズ(0～65535)
f＝フェードイン/アウトオフセット(0～65535)
l＝フェードイン/アウトレベル(0～127)

ファイル名以降は省略可能。登録先ノート番号やミキシングノート番号は'.ADPCM_BANK'命令のバンク番号が考慮される。

詳しい使い方とパラメータの意味はMEASURE6参照。

((MEASURE3 m_pcmset()，MEASURE6参照)

.ERASE m .ERASE .Onk(.Oの'.'はあってもなくてもいい)

m＝ノート番号(0～511)
n＝オクターブ値(-1～9)
k＝音階MML(abcdefg，#，+，-)

不要なAD PCMノートを削除する。

ZPCNV.R専用の命令で，ZMS中に実行してもなんの機能も果たさない。

(MEASURE6参照)

.ADPCM_LIST ファイル名

AD PCMデータのコンフィギュレーションファイルを読み込み実行する。

(MEASURE3 m_pcmcnf()，MEASURE6参照)

ADPCM_BLOCK_DATA ファイル名

AD PCMブロックデータ「ZPDファイル」の読み込み，登録を行う。拡張子省略時には'.ZPD'が自動添付される。

((MEASURE3 m_adpcm_block()，MEASURE6参照)

●MIDIデータ出力

(X n1,n2,…,ni) *

MIDIの生データを送信する。MUSICZ.FNCの'm_out()'とほぼ同様のコマンドだが，パラメータをいくつでも書くことができる(複数行にわたっても可)。

データは0≦ni≦255。256以上はその値を7ビットごとに分けて下位から送信される。

例

8192→$00,$40
32700→$3C,$7F,$01

(MEASURE3 m_out()，MEASURE9参照)

.MIDI_DATA {n1,n2,…ni} *

MIDIの生データの送信。

データは0≦ni≦255。256以上はその値を7ビットごとに分けて下位から送信される。

例

8192→$00,$40
32700→$3C,$7F,$01

(MEASURE3 m_dirout()，MEASURE9参照)

.EXCLUSIVE {n1,n2,…,ni} *

エクスクルーシブデータの送信。

データは0≦ni≦127。128以上はその値を7ビットごとに分けて下位から送信される。

例

255→$01,$7F
8192→$00,$40
32700→$3C,$7F,$01

(MEASURE3 m_exc()，MEASURE9参照)

.ROLAND_EXCLUSIVE dev,mdl {n1,n2,…,ni} *

ローランド系の楽器へエクスクルーシブメッセージを送る。詳しい使い方はMEASURE9参照。

dev＝デバイスID，mdl＝モデルID

エクスクルーシブヘッダやチェックサムバイトは自動生成して送信してくれる。データは0≦ni≦127。128以上はその値を7ビットごとに分けて下位から送信される。

例

255→$01,$7F
8192→$00,$40
32700→$3C,$7F,$01

(MEASURE3 m_roland()，MEASURE9参照)

MIDI_DUMP ファイル名

MIDIダンプデータ「MDDデータ」(MEASURE9参照)を楽器へ送信する。拡張子省略時には'.MDD'が自動添付される。
(MEASURE3 m_trans(), MEASURE9参照)

●MIDIデータ入力

(R)

MIDIの生データを取り込み待機状態にする。ミュージックプログラム中に記述しても意味を持たない。詳しい使い方はMEASURE9を参照。
ZMDデータとしては生成されないコマンドである。
(MEASURE3 m_rec(), MEASURE9参照)

●SC-55/SC-155/CM-300専用命令

パラメータの意味などは楽器のマニュアルあるいはMEASURE3を参照してください。

.SC55_INIT id

SC-55の初期化を行う
id:SC-55のデバイスID(省略可能, 初期値$10)
(MEASURE3 sc55_init()参照)

.SC55_V_RESERVE id {n1,n2,…,n16}*

SC-55の各パートのボイスリザーブを行う。
id:SC-55のデバイスID(省略可能, 初期値$10)。
パラメータは必ず16個設定しなくてはならない。また, 10番目のパラメータはリズムパートに相当する。
(SC-55マニュアルP79, またはMEASURE3 sc55_v_reserve()参照)

.SC55_REVERB id {n1,n2,…,n7} *

SC-55のリバーブパラメータの設定を行う。
id:SC-55のデバイスID (省略可能, 初期値$10)。
パラメータは7個まで。
(SC-55マニュアルP79, またはMEASURE3 sc55_reverb()参照)

.SC55_CHORUS id {n1,n2,…,n8} *

SC-55のコーラスパラメータの設定を行う。
id:SC-55のデバイスID(省略可能, 初期値$10)。
パラメータは8個まで。
(SC-55マニュアルP79, またはMEASURE3 sc55_chorus()参照)

.SC55_PART_SETUP pt,id {n1,n2,…,n119}*

SC-55のパートのパラメータを設定する。
pt:パートナンバー(1≦pt≦16)省略不可(パートナンバー=10はリズムパート)。
id:SC-55のデバイスID(省略可能, 初期値$10)。
パラメータは119個まで。
(SC-55マニュアルP79~80, またはMEASURE3 sc55_part_setup()参照)

.SC55_DRUM_SETUP map,key,id {n1,n2,…,n8}*

SC-55のドラムキットの設定を変える。
map:マップナンバー(0, 1)。
key:ノートナンバー(0≦key≦127)。
id:SC-55のデバイスID(省略可能, 初期値$10)。
パラメータは8個まで。
(SC-55マニュアルP82, またはMEASURE3 sc55_drum_setup()参照)

.SC55_PRINT id "文字列"

SC-55のコンソールに文字列を表示する。
id:SC-55のデバイスID(省略可能, 初期値$10)。
文字列は32文字以内。
(MEASURE3 sc55_print()参照)

.SC55_DISPLAY id {n1,n2,…,n16} *

SC-55のグラフィックディスプレイにドットパターンを表示する。
id:SC-55のデバイスID(省略可能, 初期値$10)。
パラメータは必ず16個。

例

```
/画面に'善'を出す
.sc55_display$10 {      %0001000000010000
                        %0000100000100000
                        %0111111111111100
                        %0000000100000000
                        %0011111111111000
                        %0000000100000000
                        %0111111111111100
                        %0001000100010000
                        %0000100100100000
                        %1111111111111110
                        %0000000000000000
                        %0011111111111000
                        %0010000000001000
                        %0010000000001000
                        %0011111111111000
                        %0010000000001000}
```

(MEASURE3 sc55_display()参照)

●MT-32/CM-32L専用命令

パラメータの意味などは, 楽器のマニュアルあるいはMEASURE3を参照してください。

.MT32_INIT id

MT-32の初期化を行う
id:MT-32のデバイスID (省略可能, 初期値$10)
(MEASURE3 mt32_init()参照)

.MT32_P_RESERVE id {n1,n2,…,n9}*

MT-32の各パートのパーシャルリザーブを行う。
id:MT-32のデバイスID (省略可能, 初期値$10)。
パラメータは必ず9個, パラメータ各値の総和は32以内。
9番目はリズムパート。
(MEASURE3 mt32_p_reserve()参照)

.MT32_REVERB id {n1,n2,n3} *

MT-32のリバーブパラメータを設定する。
id:MT-32のデバイスID(省略可能, 初期値$10)。
パラメータは3個まで。
(MT-32マニュアルP35, CM-64マニュアルP30, またはMEASURE3 mt32_reverb()参照)

.MT32_PART_SETUP id {n1,n2,…,n9}*

MT-32の各パートのMIDIチャンネルを設定する。
id:MT-32のデバイスID (省略可能, 初期値$10)。
パラメータは9個まで。9番目はリズムパート。
パラメータはMIDIチャンネルなので1~16まで有効。17以上はパートOFFとみなす。
(MT-32マニュアルP35, CM-64マニュアルP30, またはMEASURE3 mt32_part_setup()参照)

.MT32_DRUM_SETUP n,id {n1,n2,n3,n4}*

MT-32のリズムキットの設定を変更する。
n:変更対象ノートナンバー(24≦n≦87)。
id:MT-32のデバイスID (省略可能, 初期値$10)。

パラメータは4個まで。
(MT-32マニュアルP35, CM6-4マニュアルP30, またはMEASURE3 mt32_drum_setup()参照)

.MT32_COMMON n,id {"名前",n1,n2,n3,n4}*
MT-32の音色のコモンパラメータを設定する。
n：設定対象ティンバー番号(1≦n≦64)。
id：MT-32のデバイスID（省略可能，初期値$10)。
名前は10文字以内(省略不可)。
パラメータは4個まで。
(MT-32マニュアルP34, CM-64マニュアルP29, または MEASURE3 mt32_common()参照)

.MT32_PATCH n,id {n1,n2,…,n7}　*
MT-32のパッチを設定する。
n：設定対象パッチナンバー(1≦n≦128)。
id：MT-32のデバイスID（省略可能，初期値$10)。
パラメータは7個まで。
(MT-32マニュアルP35, CM-64マニュアルP30, またはMEASURE3 mt32_patch()参照)

.MT32_PARTIAL n,p,id {n1,n2,…,n58}*
MT-32の音色のパーシャルパラメータを設定する。
n：設定対象ティンバー番号(1≦n≦64)。
p：設定対象パーシャルナンバー(1≦p≦4)。
id：MT-32のデバイスID（省略可能，初期値$10)。
パラメータは58個まで。
(MT-32マニュアルP34, CM-64マニュアルP29, またはMEASURE3 mt32_partial()参照)

.MT32_PRINT id "文字列"
MT-32のコンソールに文字列を表示する。
id：MT-32のデバイスID（省略可能，初期値$10)。
文字列は20文字以内。
(MEASURE3 mt32_print()参照)

●U220/U20専用命令

パラメータの意味などは，楽器のマニュアルあるいはMEASURE3を参照のこと

.U220_SETUP id {n1,n2,…,n7}　*
U220のセットアップパラメータを設定する。
id：U220のデバイスID（省略可能，初期値$10)。
パラメータは必ず7個。
(U220マニュアルP46, 146, 148, またはMEASURE3 u220_setup()参照)

.U220_PART_SETUP pt,id {n1,n2,…,n13}*
U220のテンポラリパッチのパートパラメータを設定する。
pt：パートナンバー(1≦pt≦6)
id：U220のデバイスID（省略可能，初期値$10)
パラメータは必ず13個。
(U220マニュアルP58, 149, またはMEASURE3 u220_part_setup()参照)

.U220_COMMON id {n1,n2,…,n18}　*
U220のテンポラリパッチのコモンパラメータを設定する。
id：U220のデバイスID（省略可能，初期値$10)
パラメータは必ず18個
(U220マニュアルP54, 149, またはMEASURE3 u220_common()参照)

.U220_TIMBRE n,id {"音色名",n1,n2,…,n12}*
U220に音色パラメータを設定する。
n：セット先音色ナンバー(1≦n≦128)
id：U220のデバイスID（省略可能，初期値$10)
音色名は12文字以内。
パラメータは必ず26個 。
(U220マニュアルP72, 149, またはMEASURE3 u220_timbre()参照)

.U220_DRUM_SETUP id {n1,n2,…,n7}*
U220のテンポラリパッチのドラムパラメータを設定する。
id：U220のデバイスID（省略可能，初期値$10)
パラメータは必ず7個。
(U220マニュアルP63, 149, またはMEASURE3 u220_drum_setup()参照)

.U220_DRUM_INST n,id {n1,n2,…,n20}*
U220のテンポラリドラムキットの各ノートにおけるパラメータを設定する。
n：ノートナンバー(35≦n≦99)
id：U220のデバイスID（省略可能，初期値$10)
パラメータは20個以内。
(U220マニュアルP72, 149, またはMEASURE3 u220_midi_inst()参照)

.U220_PRINT id {"文字列"}
U220のテンポラリパッチの名前を設定する。
id：U220のデバイスID（省略可能，初期値$10)
文字列は12文字以内。
(MEASURE3 u220_print()参照)

●M1専用命令

M1専用命令は，途中でM1関係以外の命令が入ると正常なデータが楽器側へ送信されないので注意。パラメータの意味などは，楽器のマニュアルあるいはMEASURE3を参照してください。

.M1_MIDI_CH {ch1,ch2,…,ch8}
M1のSEQ0の各パートの受信チャンネルを設定する。
パラメータは必ず8個(1≦chi≦16：MIDI CH, 17≦chi：off)
例
.m1_midi_ch {1,2,3,4,5,6,17,10}
(パート1～6をMIDIチャンネル1～6に設定し，パート7はOFF, パート8をMIDIチャンネル10に設定)
(MEASURE3 m1_midi_ch()参照)

.M1_PART_SETUP {n1,n2,…,n40}　*
M1のSEQ0の各パートのパラメータを設定する。
パラメータは必ず5×8パート分＝40個。
(M1マニュアルP126, またはMEASURE3 m1_part_set_up()参照)

.M1_EFFECT_SETUP {n1,n2,…,n25}*
M1のSEQ0のエフェクトパラメータの設定。
パラメータは必ず25個。
(M1マニュアルP127, またはMEASURE3 m1_effect_setup()参照)

.M1_PRINT "文字列"
M1のSEQ0のソング名の設定。
文字列は10文字以内。
(MEASURE3 m1_print()参照)

.SEND_TO_M1 id
'.M1_MIDI_CH', 'M1_PART_SEUP', 'M1_EFFECT_SETUP', '

M1_PRINT'で設定したパラメータをM1へ送信する。
id：M1のデバイスID（省略可能，初期値$30）
備考　デバイスIDは$30＋グローバルチャンネル(0～$f)で求められる。グローバルチャンネルはGLOBALモードのF5-1で設定できる。必ず'.M1_MIDI_CH'～'.M1_PRINT'を設定してからこの命令を実行すること。
'.M1_EFFECT_SETUP'のみ省略が可能。このときはZ-MUSICのデフォルトデータが送信される。
'.M1_MIDI_CH'～'.M1_PRINT'の間にM1関係以外のコマンドがあると正常なデータが送信されないので注意すること。
(MEASURE3 send_to_m1()参照)

MEASURE 5
MML

ここではZMUSICのMMLについて解説します。

5.1. MMLの文法解説を読むにあたって

ZMUSIC.Xは基本的にシャープ/ハドソン製のOPMDRV.Xに上位コンパチです。つまりOPMDRV.Xに存在したMMLはすべて共通に使用することができ，OPMDRV.X用のデータはほとんど変更なしで演奏が可能です。

ZMUSIC.XではMMLの小文字，大文字の区別を一切していません。よって状況，ユーザー各位の趣味に応じて自由に使い分けが可能です。ここでは説明の都合上MMLを大文字でパラメータを小文字で表記することにします。

省略してもよいパラメータはその旨を記述してありますが，それ以外は省略できません。また，数値パラメータは頭に'$'をつければ16進数を，'%'をつければ2進数を設定することができます。なにもつけなければ10進数です。

各コマンドの頭についている印は各音源の種類を表しています。すべてのコマンドがすべての音源に対応しているわけではないので注意してください。対応していないデバイスに対してそのコマンドを用いるとエラーになるケースと無視されるケースがあります。

内蔵FM音源	FM
内蔵AD PCM	PCM
MIDI音源	MIDI

5.2.MML解説

■音階，休符

FM PCM MIDI

An~Gn

音階。ABC~Gはハ長調のラシド~ソに対応する。後ろに#，+をつければ半音上がり，-をつけると半音下がる。nは音長を表し，1が全音符，2が2分音符……となる。nはマスタークロック設定コマンド'm_count()'や(Zn)によって設定範囲が変動する。具体的には，

1≦n≦マスタークロックの値

が設定範囲となる。以下にデフォルトのマスタークロック＝192のときの音楽的音長と絶対音長との対応を示す。

音楽的音長	絶対音長	音楽的音長	絶対音長
1	192	12	16
2	96	16	12
3	64	24	8
4	48	32	6
6	32	64	3
8	24	192	1

絶対音長＝int(マスタークロック/音楽的音長)

nは省略可能で，省略時は後述のL/@Lコマンドで設定したデフォルト音長で演奏される。

nの前に'*'をつけると絶対音長指定になる。絶対音長とはZ-MUSICが処理するカウンタのことで，デフォルトでは全音符が192になっている。このとき，たとえばC4とC*48は同時間発音されるわけである。'*'の後ろには0~65534までの数値を書くことができる。

例

```
a#4
b*386
```

FM PCM MIDI

Rn

休符。nは音長を表す。nは音階と同様の指定が可能。

nの省略時はデフォルト音長が起用される。

例

```
r8
r*96
```

FM PCM MIDI

@Wn

指定時間，前の状態を保存する。nは音長を表す。nは音階と同様の指定が可能。

nの省略時はデフォルト音長が起用される。

例

```
@w*64
@w4..
```

■変化記号/調号

FM PCM MIDI

#
+

シャープ(#，嬰記号)。音階MMLの直後につけると半音高く演奏される。2つ以上記述することもできる。

例

```
c#4 f+8 g#16
```

FM PCM MIDI

-

フラット(b，変記号)。音階MMLの直後につけると半音低く演奏される。2つ以上記述することもできる。

例

```
c-4 e-8 g--16
```

FM PCM MIDI

[K.SIGN ~]

調号を設定する。どの音階にどの変化記号(#，b)がつくのかを設定できる。~の部分は変化記号，音階の順に設定し，1個ずつ「,」で区切る。

例

```
[K.SIGN +c,+d,+f,+g]  (ホ長調)
```

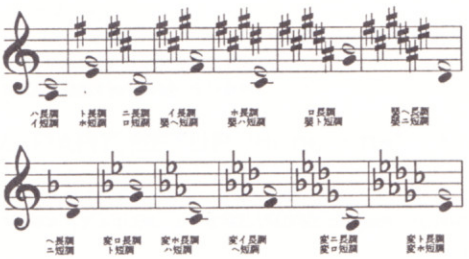
cdfgに自動的に#(シャープ)がつく

```
[K.SIGN -a,-b,-d,-e]  (変イ長調)
```

abdeに自動的に-(フラット)がつく

```
[K.SIGN -a,+b]
```

aに-(フラット)が，bに#(シャープ)が自動的につく



FM PCM MIDI

!

ナチュラル(日本位記号)。音階MMLの直後に設定することで，[K.SIGN ~]で設定した調号を臨時に取り消すことができる。

例

```
[K.SIGN -a] a a!
```

としたとき，1回目のaはa-で，2回目のaはフラットなしのaで演奏される。

■オクターブ

FM PCM MIDI

On

オクターブを設定する。初期値は4。nは-1~9まで設定可能。

実際の発音域は各操作デバイスによって異なる。

- FM音源

オクターブ0のD#からオクターブ8のDまで。

- AD PCM

登録しなければ発音はしないが，セレクトバンク内の(@コマンド，MEASURE6参照)オクターブ−1のCからオクターブ9のGまで対応。

- MIDI

オクターブ-1のCからオクターブ9のGまで。実際は楽器によって違う。各MIDI楽器のマニュアルのインプリメンテーションチャートを参照のこと。

FM PCM MIDI

<
>

オクターブを相対的に変更する。<はオクターブを＋1し，>はオクターブを−1する。設定範囲を超えた場合はエラーとなる。

■音長制御

FM PCM MIDI

Ln
@Ln

デフォルト音長を設定する(シーケンサでいうステップタイムに相当する)。音長をパラメータに持つMMLにおいて音長が省略された場合，このコマンドで設定したものがパラメータとして与えられる。

Lのパラメータnは音楽的音長，絶対音長のいずれかを書くことができる。音楽的音長の場合はn分音符といった意味をなす(設定範囲については音階MMLの説明を参照)。付点も設定可能。絶対音長は音階のときと同じように数値の前に'*'を書いて設定できる。設定範囲は 0 ≦n≦65534。

@Lのパラメータは絶対音長のみで設定範囲は 0 ≦n≦65534。

初期値はL4 (＝@L48) である。

例

```
l16.
@l96
```

FM PCM MIDI

•

付点。音長を示す数値の後ろにつけると音長値の0.5倍されたものが加算される。さらに続けると元の音長の0.25倍，0.125倍……といったものが加算されていく。ただし加算結果が65534以下でなければならない。

絶対音長指定の後ろにつけることもできる。また，加算する音長カウントは整数で処理されるため，算出された音長がアンダーフローするとエラーになる。

音長を省略したときに付点がついた場合は，その時点でのデフォルト音長に対して付点処理された音長がその音符の音長となる。

例

```
a4..
r8.
c*1000.     (c*1500と同じ)
l4 c.       (c4.と同じ)
```

FM PCM MIDI

Qn

1音中で実際に発音している時間を音長のn/8時間単位で設定する(シーケンサでいうゲートタイムに相当する)。

初期値n＝8，設定範囲は1〜8。1が発音時間がもっとも短く(スタッカート)，8がもっとも長い(テヌート)。

FM PCM MIDI

@Qn

キーオフ（発音をやめること）を絶対音長でnカウント早める。発音音長より大きい値の場合，@Qは機能せずQ8の状態で演奏される。たとえば，@Q96　C*16の場合は，発音音長が16-96<0となるので@Qの処理はされず，発音長は16カウントとなる。

初期値は@Q0に相当し，設定範囲は0〜32768。

@Qを設定したときは前のQは機能せず，逆にQを設定したときは前の@Qは機能しない。つまりあとに設定したものが有効となる。

絶対音長0と1

OPMDRV.Xでは絶対音長1の音符はキーオフしないというバグがありましたが，これを逆手に取ってピッチエンベロープなどを実現するテクニックがいつのまにか一般的となってしまいました。Z-MUSICでもこのバグ技を使えるようにしてあります。つまり絶対音長1では，

```
@L1 c&d&e&f&g
```

と，

```
@L1 cedfg
```

は同じということになります。

また、Z-MUSICでは休符、ウェイト、和音とポルタメントを除いて絶対音長0も使用可能になっています(音楽的音長0は使用不可)。音長がゼロの音符は発音処理を行うだけです。操作対象デバイスによって多少の効果の相違がありますので注意してください。

●MIDI

```
c*0e*0g*0
```

ではcegの和音が鳴りっぱなしになります

MIDI楽器の中にはリズムキットを持つものが多くあり、たいていノートオンのメッセージのみで操作可能です(ノートオフの処理を省略してもかまわない)。こういったリズムキットをシーケンスする場合に絶対音長0は有用です。

たとえば02C=バスドラム，02D=スネアドラム，03C+=クラッシュシンバルのような ローランド系リズムキットで，

```
l4 03C+*0 |:4 02 CDCD :|
```

とすると、最初の小節の頭だけシンバルが鳴り、2回目以降は鳴らないといったシーケンスができます。

絶対音長0の音符は楽器の最大同時発声数内で1トラックにつきいくつでも設定可能です。

絶対音長0の音符はそのままでは鳴りっぱなし状態になりますが、これを('&'をつけて)タイで通常音符へつなぐとその通常音符が鳴り終わった時点で絶対音長0の音もノートオフすることができます。

```
c*0&e*0&g4
```

とするとcegの和音が4分音符長で鳴ります。専用和音コマンドでは和音の各構成音に対してベロシティのバラツキを設定できませんが、この絶対音長0の音符を用いて，

```
@u110c*0& @u105e*0& @u88g4
```

のようにして実現できます。

この'&'で結ぶ書式では最大8和音までが記述できます。8和音の時は音長ゼロの音符が7つと通常音長の音符が1つという構成になることになります。音長ゼロの音符を8つ以上記述した場合は古い順にノートオフされない音が出てきます。また、従来の和音コマンドとの混在はもちろん可能で、状況状況に応じて使い分けができます。また、この2つの書式間のタイ/スラーも可能です。

●FM音源

基本的にトラックごとにモノフォニックですので，

```
c*0e*0g*0
```

のような表記で和音を発音させることはできません。しかし、MIDIと同じように絶対音長0の音符を'&'で通常音符へつなぐことによって和音を発音させることはできます。ですから、FM音源でも

```
c*0&e*0&g4
v10c*0& v12e*0& v11g4
```

のような記述で和音の発音が可能です。もちろん、従来の和音コマンドとの混在も可能で、この2つの書式間のタイ/スラーも可能です。

●AD PCM

完全にモノフォニックであるため、絶対音長0による和音発音はできません。

■タイ/スラー

FM PCM MIDI

&

前後の音をつなぐ。同音名の場合は全デバイスにおいてタイとなる。

例

```
a4&a16.
```

異音名の場合は，各デバイスによって反応が異なる。

●FM音源

スラーとなる（キーオフせずに音程だけ変える）。

例

```
a4&a#4
```

●AD PCM

'&'を除いたときのように処理される。

例

```
a4&a#4  →  a4a#4のように演奏される。
```

●MIDI

後述のタイモード(@Jコマンドにより設定)によって対応が異なる。

○通常モード時(@J0)

'&'を除いたときのように処理される。

例

```
a4&a#4  →  a4a#4のように演奏される。
```

○FM音源部互換モード時(@J1)

スラーとなる。

例

```
a4&a#4
```

FM PCM MIDI

^n

音符の直後に書くことによって，nで指定した音長分をその音符の音長に加算する。nの後ろに付点を書けば，付点を考慮した音長が加算される(PC-9801方式のタイ)。nには絶対音長を書くこともできる。

例

```
c4^16
d^*1000
'ceg4^16'
(d2^16<d)
```

MIDI

@Jn

MIDI部におけるタイの処理モードを設定する。

n=0で通常モード(初期値)，n=1でFM音源部互換モードになる。

```
c&d
```

とした場合，通常(@J0時)はMIDIでは&のない，

```
c d
```

として処理される。

しかし，@J1時ではFM音源部と同様に，ある音の発音後，その音の持続音(減衰音)で次の音の音程へ変化してくれるようになる。よって，FM音源部の，

```
(c<d)&(d>c)
```

や，後ろのコラムで動作の相違があると書かれている，

```
(g,e)24&e
@b0,683 c4&c+4
```

のような表記もFM音源部と同様に機能するようになる(ただし，最初にキーオンした音階から1オクターブを超えることはできない)。

■連符

FM PCM MIDI

{〜} n

{}で囲まれたMML群を合計の音長がn分音符音長になるように演奏する。nには付点をつけたり，'*'をつけて絶対音長による指定も行える。{}の中には音長を操作するMMLを書くことはできない(付点,L,@L,Q,@Qなど)。

nの省略時はデフォルト音長が起用される。

例

```
{cdef} 2
{abc} 4..
{cd&de} *192
```

■和音

FM MIDI

'〜'nn,dly(&)

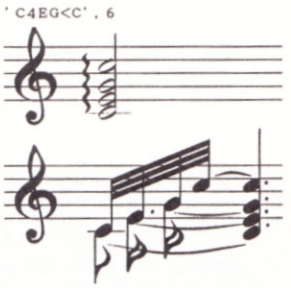
和音を発音する。'〜'間に書けるMMLは以下のとおりである。

音階	c〜b(8個まで)
調号	# + -
音長	1〜(付点も可)
絶対音長	*1〜*65534
オクターブコマンド	O-1〜O9
オクターブスイッチ	< >

一度に8和音まで発音が可能。8音以上音階を書くとエラーとなる。オクターブスイッチは'〜'(シングルクォーテーション)内だけ有効。'〜'内に音長を書かずに外に数値を書くと，これは絶対音長の指定になる。設定範囲は0≦nn≦65534で全音符以上の発音時間も設定できる。

FM音源部では指定したチャンネル以降の連続したチャンネルを使用して和音を実現している。

ディレイパラメータdlyを設定することによって，和音を一度に発音せず，各音を絶対音長dlyずつ時間的にずらして1音ずつ発声する(図のアルペジオ奏法)。ディレイは初期値0で，省略可能だが，省略時は以前設定した値がそのまま起用される。



例

```
'c4eg'
```

ドミソを4分音符で演奏する。

```
o4'gb<d'96
```

ソシレを絶対音長96で演奏する。和音発音後のデフォルトオクターブは4に変わりないことに注意。

```
'c4eg<c',6
```

まずcが発音され，その6カウント後にeが発音され，さらに6カウントごとにg，<cが発音される(図のMML表記に相当する)。ディレイ値は必ず全部の音が発声可能な値でなければエラーとなる。たとえば，

```
'c4eg',30
```

などはgが発音できないのでエラーとなる。

同構成音の和音〜和音間，異構成音の和音〜和音間，単音〜和音間，和音〜単音間を'&'で結んでのタイ/スラーが可能。ただし処理の関係上，タイでつながれた和音のディレイは強制的に0に設定される。つまり，

```
'c4eg',6&'e4gb-',6
```

は，

```
'c4eg',6&'e4gb-',0
```

として処理される。

また，オートベンドやモジュレーション，アフタータッチシーケンス，ARCCなどすべての特殊コマンドが和音においても有効である(当然のことながら，FM音源に対してのARCCは無効だが)。もちろん同時にこれらを和音に対して使用できる。

FM音源部での和音を用いた場合，チャンネルがかちあうと正常に発音されないので注意。たとえば，

```
(t1)'ceg'48
(t2)cde          ←正常に演奏されない
(t3)efg          ←正常に演奏されない
```

は以下のようにしなければならない。

```
(t1)'ceg'48
(t2)@w4
(t3)@w4
```

この例の場合でトラック2や3が未使用の場合は問題はない。

```
(t1)'ceg'48
```

（トラック2，3はアサインも確保もしていない）

■音量/ベロシティ

FM PCM MIDI

Vn

音量の設定を行う。設定範囲は0≦n≦16。初期値は8。AD PCM音源部で使用可能となるのは，PCM8独立チャンネルモードのときのみである。

FM PCM MIDI

@Vn

音量の設定を細かく行う（絶対音量と呼ぶ）。設定範囲は0≦n≦127である。AD PCM音源部で使用可能となるのは，PCM8独立チャンネルモードのときのみである。

VとV

MIDI音源によっては音量をベロシティでのみ決定するものがあります（RX-8など）。そういう楽器に対してのV/@Vコマンド動作はまったく保証されません。

FM音源，AD PCM（PCM8独立チャンネルモード時）/MIDI音源では@VとVの対応値が異なります。これはFM音源の出力音量が指数関数的に増減するのに対してMIDI（&AD PCM）では線形に増減するからです。以下に音源別にに@VとVの対応表を示します。

FM音源

```
Vn    0  1  2  3  4  5  6   7   8   9  10  11  12  13  14  15  16
@Vn 85 87 90 93 95 98 101 103 106 109 111 114 117 119 122 125 127
```

MIDI音源（&AD PCM）

```
Vn    0  1  2  3  4  5  6  7  8  9 10 11 12  13  14  15  16
@Vn 0  7 15 23 31 39 47 55 63 71 79 87 95 103 111 119 127
```

以上のことから相対ボリューム_n ¯nの使用結果においても結果が異なります。たとえば，

```
v8 _1
```

とした場合FM音源では結果が@V105に相当しますがMIDIでは@V62になります。

また，FM音源ではベロシティとボリュームがまったく同じ機能をします。たとえば，

```
V8      @U106      @V106
```

などは同じ機能を果たします。

なお，PCM8独立チャンネルモード時のAD PCM音はV9が元音量です。

FM PCM MIDI

¯n
_n

音量の相対指定を行う。¯は音量増加を表し，_は音量減少を表す。設定範囲は0≦n≦127で，演算の結果，アンダーフローを起こした場合は0に，オーバーフローを起こした場合は127に修正される。また計算は@Vnで表される絶対音量レベルで行われる。

nを省略した場合は以前に設定したものが起用される。nの初期値は1。

なお，AD PCM音源部で使用可能となるのはPCM8独立チャンネルモードのときのみである。

例

```
@v120 _10 c _ d ¯ e
```

（cは@120−10＝@v110で，dは@v110−10＝@v100で，eは@v100+10＝@v110で演奏される）

```
|:10 _10 c :|
```

（だんだんと音量が下がりながらcが演奏される）

FM MIDI

@Un
Un

ベロシティの設定を行う。ベロシティとは鍵盤を叩く速度，すなわち音量のこと。

MIDI楽器によっては，これで音量のみならず音のニュアンスをも変えることができる。設定範囲は0≦n≦127で，127が最強（大）に相当。またFM音源では単純に音量として扱われる。

初期値は127である。

FM MIDI

@U+n
@U−n
U+n
U−n

ベロシティの相対指定を行う。設定範囲は−127≦n≦+127。演算によってオーバーフローを起こした場合は，設定可能範囲内に修正される。

初期値は1。

使用例

```
@u127c @u-10d @u+5e
u127c  u-10d  u+5e
```

c,d,eはそれぞれベロシティ127，117，122でキーオンする。

FM MIDI

@U

以前設定した相対ベロシティの値でもう一度相対ベロシティの指定を行う。

例

```
@u127 @u-10 @u
```

最後の@uもまた@u-10で機能する。

FM MIDI

U

以前設定した@Uの値を再設定する。

例

```
@u127c u-10d ue
```

c,d,eがそれぞれベロシティ127，117，127でキーオンされる。

FM MIDI

Zn1,…,n16

ベロシティのばらつきを設定する。これをZMUSIC.Xではベロシティシーケンスと呼ぶ。各値の設定範囲は0≦n≦127。

これはこの先最大16個の音階MML，和音MML，ポルタメントMMLのベロシティを先に設定するものである。

たとえば，

```
z127,99,80        cdefg
```

とするとcはベロシティ127で，dはベロシティ99で，eはベロシティ80で，ここでfは再びベロシティ127で，gはベロシティ99で発音される。値は最大16個まで記述することができ，1個のみ指定したときは@Uとまったく同様に機能する。数値の頭に±を記述すれば相対指定が可能である。

数値をまったく書かずに'Z'のみ記述するとベロシティシーケンスのOFFと解釈される。

ベロシティシーケンスはループコマンドを無視して処理されるので，

```
z99,100,127 |:100 c :| d e
```

とした場合cが100回ベロシティ99で，ループ外のdがベロシティ100で，eがベロシティ127で演奏される。

このコマンドを効果的に使えばピアノソロなどを極めて人間的にシーケンスすることができる。MIDI楽器によってはベロシティの変化に伴って音色のニュアンスや音色自体を変えることができるので，そういった機能と合わせて使えばかなり興味深い効果を得ることができる。

注意

FM音源にこれを用いた場合は，単純にその値に対応した絶対音量で発音される。

ベロシティコマンド@U/Uを使用すると自動的にベロシティシーケンスは解除される。

Uと@U

機能的にはまったく同じですが，演奏時にリアルタイムに考慮されるか否かの相違があります。

```
@u100|:4 @u+5 c :|
```

としたとき４回とも@u105でcが演奏されますが，

```
@u100|:4 u+5 c :|
```

としたときは@u105，@u110，@u115，@u120でそれぞれ演奏されます。

なお，管理は@UとUはまったく別です。ですから，

```
@u5  |:3 @u+50 c u-5:| d
```

としたとき@u55で3回cが演奏され@u50でdが演奏されます。

Uのみではそれまでに最後に設定した@Uをもう１回設定するので，

```
@u110|:10 u+2 c :|u e
```

のように，ループで変化してしまったベロシティを元に戻すことができます。この例では，直前のベロシティ値によらずeは@u110で演奏されます。

このようにU，@Uを使い分ければ，かなり複雑なシーケンスをすることができます。@Uは，ループの影響を受けないグローバルな存在，Uはループなどの影響を受ける局所的な存在というわけです。

■ピッチシフト/ディチューン/オートベンド

FM MIDI

Kn

音程を半音単位で上下させる(キートランスポーズ)。nはn半音を意味し，設定範囲は−128≦n≦127(±10オクターブ程度に相当)。初期値は0，またニュートラル（中央値）も0である。

FM MIDI

@Kn

キートランスポーズを1/64半音単位で行う(いわゆるディチューン)。設定可能範囲は−768≦n≦768(±1オクターブ程度に相当)。初期値は0，またニュートラルも0である。

Kと@Kはまったく管理が別である。たとえば，

```
@k10 cdefg
```

の頭にk1をつけて，

```
k1 @k10 cdefg
```

としたとすると，１半音と10/64半音上にピッチシフトされてcdefgが演奏される。いちいちMMLを書き直すことなく，各パートのMMLの頭にKコマンドを書くだけでディチューンなども考慮した転調が実現できる。

FM MIDI

@Bn

ベンド値(ディチューン)を設定する。@Kとまったく同じ機能と考えてよいが，１オクターブ≒8192で換算される点が違う。ユーザーがMIDI音源に慣れている場合は@Bを，FM音源に慣れている場合は@Kを使うとよいだろう。設定範囲は-8192≦n≦8191で，初期値は0，ニュートラルも0である。

FM MIDI

@Kn1,n2,dly

オートベンドの設定を行う。@Kの後ろに２つ以上のパラメータを書くとオートベンドの設定となる。この場合，n_1はベンドスタート値，n_2はベンドエンド値，dlyはディレイ値を表す。それぞれ，

$-768 \leqq n_1 \leqq 768$

$-768 \leqq n_2 \leqq +768$

$0 \leqq dly \leqq 32767$

具体的な使用方法は，ベンドレンジが１オクターブになっている場合に，

```
@k0,-64,24      (64=1半音)
```

とすると発音後，24絶対カウント後から音程が下がり始め，キーオフまでに半音下がる。

ディレイは省略可能で，省略時は以前の設定値が起用される。初期値はn1,n2,dlyともに0である。ベンドレンジはFM音源は１オクターブに固定，MIDIは後述の@Gコマンドで変更可能である。

@Kおよび@Bにてディチューンを設定した場合，オートベンドは以降強制的にOFFとなる。

オートベンドとポルタメント

オートベンドとポルタメントの具体例と両コマンドの対象デバイスにおける効果の相違点について少し解説します。

```
(c+4<<c+),8
```

はc+をディレイで指定した8カウント分演奏したあと，４分音符-8カウントの時間内に２オクターブ上のc+まで上昇します。オクターブスイッチの影響で以降２オクターブ上になることに注意してください。

```
(g,e)24&e
```

は絶対カウント24の時間内にgからeへピッチダウンを行いキーオフせずeへつなぐことになります。

ここで注意がひとつあります。以上２つの例はFM音源においてのみ有効です。ひとつ目が実現不可能なのは，MIDIでは楽器側のベンドレンジが１オクターブと想定して処理しているため１オクターブを超えたポルタメントは不可能だからです。内蔵FM音源は音程がリニアな構造をしているため発音可能範囲内であれば任意の音程を出力可能ですが，MIDIにおいてはポルタメントは(オートベンドも)ベンダーホイールの上げ下げでピッチをコントロールしているため２つ目の例をMIDIに対して行うとFMとは違った動作を示してしまいます。２つ目の例をFMと同じ効果をMIDIで実現したい場合は，

```
(g,e)24&g
```

ということになります。つまりMIDIにおいてのタイはキーオフせず，ベンダー値を以前のまま保持することを意味しているのです。同様に，

```
o4 (c<c)&c
```

はFM音源ではo4のcから１オクターブ上のo5のcへポルタメントを実行しo5のcへ音をつなぐことができますがこれと同じことをMIDIでやるためには，

```
o4 (c<c)&>c
```

あるいは，

```
o4 (c<c)&o4c
```

としなければなりません。MIDIの場合は発音後どれくらいベンダーホイールを動かしたかで音程をコントロールするのでタイの場合はポルタメントのスタートキーを書かなければならないのです。このような相違点はオートベンドコマンドにもみられます。

```
@b0,683,0 c4          (683≒8192/12≒1半音)
```

ではFM音源ではc4がc+4へ向かって上昇しますが，c+4を持続したい場合はＦＭの場合は，

```
@b0,683,0 c4&c+4
```

とすればよいのですがMIDIの場合は，

```
@b0,683,0 c4&c4
```

としなければならないわけです。

オートベンド，ポルタメントともにQコマンドの影響を受ける点にも注意してください。たとえば，

```
q4 (b-<b-)
```

とした場合は１オクターブ上のb-に達する前にキーオフしてしまいます。

FM MIDI

@Bn_1,n_2,dly

オートベンドの設定を行う。@Bの後ろに2つ以上のパラメータを書くとオートベンドの設定となる。この場合，n_1はベンドスタート値，n_2はベンドエンド値，dlyはディレイ値を表す。それぞれ，

$-8192 \leqq n_1 \leqq 8191$

$-8192 \leqq n_2 \leqq 8191$

$0 \leqq dly \leqq 32767$

具体的な使用方法は，ベンドレンジが1オクターブになっている場合に，

@b0,-683,24　　(683≒8192/12≒1半音)

とすると発音後，24絶対カウント後から音程が下がり始め，キーオフ時までに半音下がる。

ディレイは省略可能で，省略時は以前の設定値が起用される。初期値はn_1,n_2,dlyともに0である。

ベンドレンジはFM音源は1オクターブに固定，MIDIは後述の@Gコマンドで変更可能である。

@Kおよび@Bにてディチューンを設定した場合，オートベンドは以降強制的にOFFとなる。

FM MIDI

($k_1$$k_2$)n,dly

ポルタメントを行う。k_1はポルタメントスタートキーMML，k_2はエンドキーMML。以下のMMLコマンドを()内に記述できる。

音階	c～b
調号	# ＋ －
音長	1～(付点も可)
絶対音長	＊1～＊32767
オクターブコマンド	O-1～O9
オクターブスイッチ	＜＞

和音コマンドとは違ってオクターブスイッチは( )外へも影響する。音長は和音コマンド同様にキーの後ろに書くことも，( )の外に書くこともできる。外に書いた場合は絶対カウントとみなされる。絶対音長ではnは，$0 \leqq n \leqq 32767$まで設定可能なので全音符以上の発音時間を設定することもできる。

dlyはディレイで絶対カウント単位でポルタメントのかかり方を遅らせることができる。設定可能範囲は$0 \leqq dly \leqq 32767$。dlyは省略すると以前の設定値が起用される。

音長を省略するとデフォルトの音長(Lまたは@Lコマンドで指定したもの)が起用される。

MIDI楽器において，後述の@Gコマンドでベンドレンジを12(1オクターブ)以外に変更したとき，あるいはMIDI楽器のベンドレンジが12に設定されていないときの動作は，表記どおりにならない。

例

```
o4(g2<g)
```

オクターブ4のgでキーオンし，2分音符分鳴り終えるまでにオクターブ5のgまで滑らかに音程をシフトしていく(ちょうど図のMML化)。コマンド終了後はオクターブは5になっていることに注意。

(o3d＊96,e),48

オクターブ3のdでキーオンし，このピッチで48カウント分演奏し，それ以降絶対音長96でキーオフするまでにオクターブ3のeまで音程をシフトしていく。

MIDI

@Gn

ベンドレンジの設定を行う。nはMIDI楽器によって設定範囲が異なる。また，外部からベンドレンジを操作できないものや，プログラムチェンジ(音色切り換え)時に初期化されてしまうものなどがあるので，楽器のマニュアルをよく読んでから使用したほうがいい。また，ベンドレンジを変更するとポルタメントの効果も変化してくるので注意。逆に，ベンドレンジを変更すれば，オートベンドコマンドを用いて1オクターブ以上のベンドを実現することも可能である。初期値はn＝12を設定しているが楽器側がベンドレンジを変更できないものであれば不定である。

FM音源対象トラックにおいては無効(12＝1オクターブに固定)である。

■パンポット

FM PCM MIDI

Pn

パンポットの設定を行う。設定範囲は$0 \leqq n \leqq 3$である。初期値は3。

0は出力停止，1は左，2は右，3は中央。

FM音源ではP0はミュートに相当するがMIDIではこれに対応する機能がないので便宜上@V0 @U0を行う。通常はMIDIではP0は用いないこと。

FM PCM MIDI

@Pn

パンポットを多段階指定する。一応MIDI専用だがFM/AD PCMに設定した場合は値に応じて3段階のL/M/Rに変換される。

設定範囲は$0 \leqq n \leqq 127$であるがMIDI楽器によっては無効となる場合がある(M1,SY-77など)ため各MIDI楽器のマニュアルを参照のこと。

FM PCM MIDI

@P+n
@P−n

パンポット(@P)を相対的に設定する。設定範囲は$-127 \leqq n \leqq +127$で，演算の結果，アンダーフローを起こした場合は0へ，オーバーフローを起こした場合は127へ修正される。

例

```
@P64|:10 @P+5 c16 :|
```

(音がだんだん右から鳴るようになっていく)

パンポット

ローランドMT-32系(CM-32L，CM-32P，CM-64，CM-500)のパンポットは実はMIDI規格とは正反対の値で音場が切り換わります。よってP1,P2,P3はそれぞれMT-32系においてはR,M,Lと音場が切り換わります。ZMUSIC.XはMIDI規格に則って左を@P0，中央を@P64，右を@P127としています。FM音源やその他のMIDI楽器と同じ音場を得るためにMT-32系をご使用の方は背面のアウトプットを左右入れ換えて接続することをおすすめします。

■テンポ

FM PCM MIDI

Tn

テンポを設定する。nはタイマBモード(デフォルト状態)では30～300，タイマAモード(MEASURE2参照)では77～300である。

nは1分間の4分音符の数に相当する。

初期値は120。ひとつのトラックで指定すれば全トラックに影響する。

FM PCM MIDI

T+n
T−n

テンポを相対的に設定する。演算の結果有効範囲を超えてしまった場合は設定可能範囲内に自動修正される。

ひとつのトラックで指定すれば全トラックに影響する。

FM PCM MIDI

@Tn

割り込み周期を割り込みタイマへ直接設定する。

機能的にはTコマンドと同じだがパラメータnが割り込み周期である点が違う。

以下に各タイマとテンポの相関を表す。

タイマA＝1024－(78125/テンポ)

タイマB＝256－(78125/テンポ)/16

ひとつのトラックで指定すれば全トラックに影響する。

FM PCM MIDI

@T+n
@T−n

タイマ値を相対的に設定する。演算の結果有効範囲を超えてしまった場合は設定可能範囲内に自動修正される。

ひとつのトラックで指定すれば全トラックに影響する。

■音色/バンク切り替え

FM PCM MIDI

@n

・FM音源

音色を音色番号nに設定する。1≦n≦200。初期値なし。

・MIDI

音色を音色番号nに設定する。1≦n≦128。初期値なし。

・AD PCM

4つあるAD PCMのバンクのうちどれを使用するかを設定する。1≦n≦4。初期値＝1

MIDI

I n_1,n_2

MIDI楽器の音色のバンク切り換えを行う。コルグのウェーブステーション，ローランドのSC-55，各種GM音源などでは音色がバンクに分かれており，音色切り換えはこの命令でバンクを指定してから行う。同一バンク内で切り換えを行う場合は一度設定すれば以降設定する必要はない。

n_1はバンク番号上位、n_2はバンク番号下位を表し設定範囲はn_1,n_2ともに0～127までである。パラメータは省略すると強制的に0になる。

SC-55ではバンク切り換えパラメータの上位のみ有効となっているので，

I n_1

のみでバンク(SC-55のマニュアルではバリエーションと呼んでいるが)n_1へ切り換えることができる。

ウェーブステーションではバンク切り換えパラメータの下位のみ有効としているので，

I 0,n_2

でバンクn_2へ切り換えることができる。

例

```
i0 @128
i0,12 @11
```

■操作チャンネル変更

FM PCM MIDI

Nn

そのトラックの操作対象デバイスの変更を行う。設定範囲は1≦n≦32で各数値に対応したチャンネル番号はMUSICZ.FNCの'm_ch()'命令やZMSファイルコマンド(Bn)命令によって違ってくる。

m_ch("FM"), (B0)のとき

n= 1～8	内蔵FM音源チャンネル１～８
9	内蔵AD PCM音源
10～25	MIDIチャンネル１～16
26～32	AD PCMチャンネル２～８
	(PCM8独立チャンネルモード時)

m_ch("MIDI"), (B1)のとき

n= 1～16	MIDIチャンネル１～１６
17～24	内蔵FM音源チャンネル１～８
25	内蔵AD PCM音源
26～32	AD PCMチャンネル２～８
	(PCM8独立チャンネルモード時)

FM PCM MIDI

@Nn

Nコマンドと機能的には同じ。しかしMUSICZ.FNCの'm_ch()'命令，ZMSファイルコマンドの(Bn)命令の状態に関わらず絶対的に指定が可能である。

nの設定範囲は，

n= 1～8	内蔵FM音源チャンネル１～８
9	内蔵AD PCM音源
10～25	MIDIチャンネル１～16
26～32	AD PCMチャンネル２～８
	(PCM8独立チャンネルモード時)

でm_ch("FM"), (B0)のときのNコマンドとちょうど同じである。

m_ch("MIDI")とするとN1～16はそのままMIDIチャンネル1～16に対応し，@N1～9は（当然だが）FM1～8そしてAD PCMに対応するので概念的にわかりやすい。

突然のチャンネル切り替え

キーオフしないまま異種のデバイスへチャンネルを切り替えてしまった場合，音が鳴り続けてしまう場合があります。たとえばMIDI音源でタイを使ってキーオフしないうちにFM音源チャンネルに切り替えるとMIDIの音が鳴り止まずに鳴り続けてしまいます。もしそうなってしまった場合は'm_stop()'などで音を止めるしかありません。

例

```
@N10 C&@N1C
↑     ↑FM1チャンネル
MIDI1チャンネル
```

(MIDI楽器で発音されたCが鳴りっぱなしになります)

■アフタータッチシーケンス

FM MIDI

@Zn_1,…,n_8

アフタータッチを音長の1/8時間単位で設定する。これをZ-MUSICシステムではアフタータッチシーケンスと呼ぶ。各値の設定範囲は$0 \leqq n_i \leqq 127$。

発音後の音のニュアンスを細かく変化させることを可能にする。たとえば，

```
@z127,120,110,127,80,60,50,10
```

として絶対音長192の音長でキーオンしたとすると192/8＝24で24カウントごとにアフタータッチを変化させる。順を追って説明すると発音はベロシティ127でキーオンされる。その24カウント後アフタータッチ120の強さで鍵盤を押さえた(鍵盤を押し込む強さを弱めたというべきか?)ことになり，以降も同様である。

MIDI楽器によっては発音後の鍵盤を押す強さ(すなわちアフタータッチ)によって音のニュアンスを変えることができ，サンプラーなどではまったく別の音色を発音させることも可能なので使用方法によってはかなり興味深い効果を得ることができる。

FM音源でこのコマンドを使用した場合は発音後の絶対音量を設定値で変化させることができる。ソフトエンベロープなどを実現したい場合に威力を発揮する(もちろんMIDI楽器の設定でアフタータッチに対応する効果を音量に設定しておけばFM音源と同じようにソフトエンベロープ的な効果を得ることができる)。

パラメータは任意の位置で省略可能で，数値の頭に±をつけることによって相対的な値の指定も可能である。たとえば，

```
@z127,,,10
@z,,,,120,,1
@z,,120,,-10,+5
```

といった記述が可能である。省略した部分は前の値を持続するということになり，その間はアフタータッチの情報は送信されない。先頭パラメータを省略した場合には発音は以前に設定したベロシティ@U/Uの値で行われる。それ以外は先頭のパラメータの値のベロシティでキーオンされる。

相対指定時にオーバーフロー/アンダーフローを起こした値は許容範囲内の最大値最小値に修正される。

値をすべて省略した場合はアフタータッチシーケンス解除と解釈

される。

例

@z

■ピッチモジュレーション

FM MIDI

@Mn$_1$,…,n$_8$ ...(1)

または,

@Mn ...(2)

ピッチモジュレーション（音程を細かく震わせる）の振幅設定を行う。

1) 1/8モードピッチモジュレーション

発音後，音長の1/8時間単位で振幅(モジュレーションデプス)を変化させることができる。設定可能範囲はFM音源は$-32768 \leqq n_i \leqq 32767$，MIDIのノーマルモードは$0 \leqq n_i \leqq 127$，拡張モード時は$-768 \leqq n_i \leqq 768$($-8192 \leqq n_i \leqq 8191$)。値は任意の位置で省略可能，8個まで記述でき，相対指定も可能で，

@m60,,,50,127

@m,,,,127

@m,,120,-10,+5

といった記述ができる。

こちらのモードを指定した場合は後述のディレイコマンドで設定したディレイ値は無視される。

2) ディレイモードピッチモジュレーション

振幅(モジュレーションデプス)をひとつだけ設定する。振幅の設定範囲は$-32768 \leqq n \leqq 32767$。

後述のディレイコマンドで設定したディレイのあとnの深さでピッチモジュレーションを実行する。

初期値はいずれのモードにおいても0。またすべての値を省略した場合はモジュレーション解除のスイッチとなる。

例

@m

■ARCC(Assignable realtime control change)

MIDI

@Cn,r,m

任意のコントロールチェンジナンバーnを選び，これを本命令で登録しておき，後述の@Aコマンドにて，そのコントロールチェンジでMIDI音源をリアルタイムにコントロールすることができる。これをZ-MUSICではアサイナブルリアルタイムコントロールチェンジ(Assignable realtime control change)と呼ぶ(以後ARCC)。

コントロールチェンジ番号の有効値はMIDI楽器によって異なるので各楽器のマニュアルのインプリメンテーションチャートを参考のこと。

ARCCの実行を解除したときに登録したコントロールチェンジnに対してリセットという意味でなんらかの値を設定することができる。その設定する値というのがパラメータrで，省略時は0となる。設定範囲は$0 \leqq r \leqq 127$。

後述の拡張ARCCモード使用時にはプリセット波形やユーザー波形が使用可能となるがその波形の基準値となるのがmである。詳しくは5.3.節で解説する。

MIDI

@An$_1$,…,n$_8$ ...(1)

または

@An ...(2)

1) 1/8モードARCC

発音後音長の1/8時間単位で先述の@Cで登録したコントロールチェンジに対応する値を送信し音源を制御することができる。値の設定範囲は$0 \leqq n_i \leqq 127$(拡張ARCCモード時には設定範囲は$-127 \leqq n_i \leqq 127$)。

値は任意の位置で省略可能，8個まで記述でき，相対指定も可能で，

@a60,,,50,127

@a,,,,127

@a,,120,-10,+5

といった記述ができる。

こちらのモードを指定した場合は後述のディレイコマンドで設定したディレイ値は無視される。

2) ディレイモードARCC

ARCCの設定値をひとつのみ設定する。設定範囲は$0 \leqq n \leqq 127$(拡張ARCCモード時には設定範囲は$-127 \leqq n \leqq 127$)。

後述のディレイコマンドで設定したディレイのあと，nの値で@C命令で登録したコントロールチェンジを実行する。

初期値はいずれのケースにおいても0。またすべての値を省略した場合はARCC解除のスイッチとなる。

例

@a

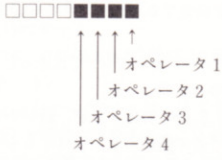
■アンプリチュードモジュレーション

FM

@Cm

X68000のFM音源は4つのオペレータ（発信器）によって構成されているがその1～4の4つのオペレータのうちどのオペレータに対してアンプリチュードモジュレーションの処理を施すかを設定できる。特殊な効果(たとえばワウワウ効果)を狙いたいときなど以外は特に設定する必要はない。

パラメータのmはビット構成を取っており，



ビット＝1に対応するオペレータに対してモジュレーション処理を施す。なお，このコマンドで設定した設定値は次回の音色切り替えまで有効である。音色切り替え時にはその音色のキャリアオペレータに対してモジュレーション処理を施すような設定値が自動的に設定される。

本命令はFM音源に対して多少の知識を必要とする。

例

@c%1001

(オペレータ1と4にモジュレーション処理をする)

FM

@An$_1$,…,n$_8$ ...(1)

または，

@An ...(2)

アンプリチュードモジュレーション（音量を細かく震わせる）の振幅設定を行う。

1) 1/8モードアンプリチュードモジュレーション

発音後音長の1/8時間単位で振幅(モジュレーションデプス)を変化させることができる。値の設定範囲は$-127 \leqq n_i \leqq 127$。値は任意の位置で省略可能，8個まで記述でき，

相対指定も可能で，

@a60,,,50,127

@a,,,,127

@a,,120,−10,+5

といった記述ができる。

こちらのモードを指定した場合は後述のディレイコマンドで設定したディレイ値は無視される。

2) ディレイモードアンプリチュードモジュレーション

振幅をひとつのみ設定する。設定範囲は$-127 \leqq n \leqq 127$。後述のディレイコマンドで設定したディレイのあと，nの振幅でモジュレーションを開始する。

初期値はいずれのケースにおいても0。またすべての値を省略した場合はアンプリチュードモジュレーションの解除スイッチとなる。

例

@a

■モジュレーションサブパラメータ

FM MIDI

@Hn1,n2

FM音源のピッチモジュレーション，アンプリチュードモジュレーション，またはMIDIのピッチモジュレーション，ARCC，拡張ピッチモジュレーション，拡張ARCCのディレイモード用のディレイクロックを設定する。

n_1はピッチモジュレーション用のディレイ。設定範囲は$0 \leqq n_1 \leqq 65534$。

n_2はアンプリチュードモジュレーション/ARCC用のディレイ。設定範囲は$0 \leqq n_2 \leqq 65534$。

値はどちらか一方の省略は可能。初期値は共に0。

例

@h24,10　(両方同時に設定)

@h,10

(アンプリチュードモジュレーション/ARCC/拡張ARCCのディレイのみ設定する)

@h24

(ピッチモジュレーションのディレイのみ設定する)

FM MIDI

@Sn1,n2

FM音源のピッチモジュレーション，アンプリチュードモジュレーション(またはMIDIの拡張ピッチモジュレーション，拡張ARCCモード時)のモジュレーションスピードを設定する。設定範囲は$1 \leqq n_1, n_2 \leqq 16383$。

本命令はMIDIのノーマルモジュレーションモードではまったく考慮されないもので，拡張モード時のみ有効である。

実際の波形にどう影響するかは5.3.節で解説するが，値が小さいほど高速で，値が大きいほど低速で音が振動する。値はどちらか一方の省略は可能。初期値はともに6。

例

@s24,10　(両方同時に設定)

@s,10

(アンプリチュードモジュレーション/拡張ARCCのスピードのみ設定する)

@s24

(ピッチモジュレーションのスピードのみ設定する)

FM MIDI

Sn1,n2

FM音源のピッチモジュレーション，アンプリチュードモジュレーション(またはMIDIの拡張ピッチモジュレーション，拡張ARCCモード時)のモジュレーション波形をセレクトする。

n_1がピッチモジュレーション用の波形，n_2がアンプリチュードモジュレーション用(または拡張ARCC)の波形に対応する。

波形は波形番号0～3がプリセット波形で，8～31がユーザー定義波形となる。スピードや振幅パラメータの波形への影響については5.3.節を参照のこと。

プリセット波形

0：鋸歯波　1：矩形波　2：三角波　3：ワンショット鋸歯波

値はどちらか一方の省略は可能。初期値はともに2の三角波。

例

s1,1　(両方同時に設定)

s,2　(アンプリチュードモジュレーション/拡張ARCCの波形のみ設定する)

s3　(ピッチモジュレーションの波形のみ設定する)

FM MIDI

Hn1,n2

FM音源のピッチモジュレーション，アンプリチュードモジュレーション(またはMIDIの拡張ピッチモジュレーション，拡張ARCCモード時)のモジュレーション波形を，キーオンと同期して初期化させるか(同期モード)，キーオンと同期して初期化せずに波形タイプの切り替えが起こるまで継続させるか(非同期モード)を設定する。

n_1がピッチモジュレーション用，n_2がアンプリチュードモジュレーション用(または拡張ARCC)の同期/非同期設定パラメータとなる。

値はどちらか一方の省略は可能。初期値は共に0の同期モードが設定されている。

例

h1,1　(両方同時に非同期モードに設定)

h,1　(アンプリチュードモジュレーション/拡張ARCCの波形を非同期モードに設定する)

h1　(ピッチモジュレーションの波形を非同期モードに設定する)

■モジュレーションモード

MIDI

Mp,a

MIDIにはモジュレーションのモードが2つある。ひとつは，ある指定されたタイミングにのみMIDI楽器へ制御メッセージを送信するモードで，これをノーマルモードと呼ぶ。もうひとつはリアルタイムに絶えずMIDI楽器へ制御メッセージを送るモードで，こちらを拡張モードと呼ぶ。

ノーマルモードは楽器側のハードウェア的な処理によって特殊効果を実現するのでZ-MUSICの負荷は少ない。一方，拡張モードは絶えず制御メッセージを送るため多少の負荷はあるものの，FM音源部のようにSコマンドでセレクトされる波形に従って制御できるためノーマルモードより多彩な表現ができる。

本命令はピッチモジュレーションとARCCをどのモードで実行するかを決定するものである。

●p:0　ノーマルモードの設定

@Mで設定された値を指定のタイミングでコントロールチェンジ1番のピッチモジュレーションに書き込む。

●p:1　拡張モード1の設定

設定された振幅の効用をFM音源の@Mと同じにして(@Mで指定される振幅は半音＝64ということになる)，Sでセレクトした波形でピッチモジュレーションを実現する。@Mで指定される振幅の範囲は$-768 \leqq @M \leqq 768$。

●p:2　拡張モード2の設定

設定された振幅の値はそのまま14ビットのMIDIのピッチベンダー直値に該当する。この振幅で，Sコマンドでセレクトした波形でピッチモジュレーションを実現する(ベンドレンジ＝12のとき，半音≒683)。@Mで指定される振幅の範囲は$-8192 \leqq @M \leqq 8191$。

●a:0　ノーマルモードの設定

@Aで設定した値を指定されたタイミングで書き込む。

●a:1　拡張モードの設定

@Aで−127～127までの振幅を設定すると，Sコマンドでセレクトした波形に従って生成される値を@Cで登録したコントロールチェンジに対して出力できる。

パラメータはどちらか一方の省略は可能。初期値はともに0(ノーマルモード)。

例

m2,1 (両方同時に設定)

m,1　(ARCCのモードのみ設定する)

m1　(ピッチモジュレーションのモードのみ設定する)

※注意

本命分を実行するとそれまで設定されていた振幅(@Mおよび@Aの値)は無効となる。

■特殊機能スイッチング

FM MIDI

=n

ピッチモジュレーション，ARCC，オートベンド，アフタータッチシーケンス，ベロシティシーケンスの各機能のスイッチングを行う。

nは各ビットごとに意味を持ち，対応は以下のとおりである。

```
d4d3d2d1d0
□□□■■■■■
      └ピッチモジュレーション
     ARCC
    オートベンド
   アフタータッチシーケンス
  ベロシティシーケンス
```

各ビットを0にするとオフ，1にするとオンとなる。

初期値は0（全機能OFF）である。

例

=%00011

（ピッチモジュレーション/ARCCをON。それ以外をOFF）

特殊効果機能の併用について

Z-MUSICではさまざまな特殊機能を組み合わせて使うことができますが，オートベンドとポルタメントは同時に使用できません。万一，オートベンド時にポルタメントを実行した場合にはポルタメントのみ機能し，オートベンドは実行されません。

しかし，これ以外の組み合わせは可能です。たとえばポルタメントする音に対してピッチモジュレーション，ARCC/アンプリチュードモジュレーションをかけて，さらにアフタータッチシーケンスを実行する……といった複数の効果をひとつの音に対して同時にかけることもできるわけです。

■発音制御

FM PCM MIDI

@Dn

ダンパーペダルの状態の設定を行う。設定範囲は0がダンパーペダルオフ，1≦n≦127がダンパーペダルオンである。

ダンパーオンの状態では発音はされるがダンパーオフになるまでキーオフされない。各楽器の最大同時発音数を超えたときには各楽器に処理が委ねられる。通常はいちばん過去に発音された音がキーオフされそのボイスを用いて新たな音がキーオンされることになるだろう。

AD PCMでは単に次の音が鳴るまで発音し続けるという動作をするだけである。

FM音源トラックの最大同時発声数

内蔵FM音源では和音とダンパーにおいての同時発声数は各トラックがどのチャンネルにアサインされているかで異なります。たとえば，あるトラックがFM音源の第1チャンネルにアサインされている場合はそのトラックの同時発音数は8だが第2チャンネルにアサインされている場合は7となります。つまりそのチャンネルから第8チャンネルまでいくつチャンネルが残っているかがFM音源にアサインされたトラックの同時発音数になるわけです。

MIDIの場合はもちろん楽器の最大発音数を超えるまで自由に発音させることができます。

FM PCM MIDI

@Rn

そのトラックは以後ノートオフ処理をしない。リズムマシンや一部の楽器のリズムキットは発音すればキーオフを送らずともよいものがある（そういった楽器には「ノートオフモード」とか「ノーサスティン」といったパラメータがあるはずである）。

こういった楽器やチャンネルに対してこのコマンドを使えばドライバのノートオフ処理を省略でき，割り込み演奏処理がいくぶん軽くなる。

このモードにはいるとQコマンドの影響を受けず，エンベロープを最後まで実行するため，音を細かく区切ったりなどというニュアンスは再現不可能になる。

n=0でノートオフ処理をする，1でノートオフ処理をしない。省略時，および初期状態はは@R0，つまり「ノートオフをする」モードに設定されている。

FM PCM MIDI

強制的にキーオフを行う。そのトラック内において発音中の音をすべてミュートする。

■フェードイン/フェードアウト

FM PCM MIDI

¥n

フェードインまたはフェードアウトを行う。nは音量増減スピードを表し絶対値が大きいほど増減スピードが速くなる。

−85≦n≦−1　フェードイン

n=0　フェードイン/モード解除

1≦n≦85　フェードアウト

各トラックごとに設定可能なので，各トラック違った速度でフェードインやフェードアウトが実行可能。フェードアウトでは出力音量が0になった時点でそのトラックの演奏を終了する。

各音源の特性から均等な音量バランスではフェードアウトされないケースが出てくる。

n=0のフェードイン/アウト解除は処理そのものを中止するだけなので元の演奏状態に戻したい場合はボリュームコマンドなどを再設定する必要がある。

■繰り返し/反復記号

FM PCM MIDI

|:n … |r1 … |r2 … :| ...(1)

または，

|:n … | … :| ...(2)

●(1)のケース

|:n～:|で囲まれたMMLをn回演奏する（n省略時はn=2が自動設定される）。

|r1を|:n～:|の中に挿入すると，|r1から:|までの演奏データを繰り返しr1回目に演奏させることができる。|:～:|は8重まで括ることが可能で複雑なシーケンスも可能である。

各値の設定可能範囲は1≦n,r1≦255である。

例

|:3 a |:2 b :| c:|

（abbc abbc abbcのように演奏される）

|:2 a |1 b :| |2 c:|　（ab ac のように演奏される）

●(2)のケース

|riのriを省略した|を|:n～:|の中に挿入すると|以降に書かれた演奏データは最後の繰り返し時には演奏はしないといった特殊例になる。

例

|:5 a | b :|　（ab ab ab ab aのように演奏される）

多重ループ構造について

Z-MUSICでは多重の繰り返し構造が設定可能ですが，必ずネスト構造になっていなければ希望通りの繰り返しを得ることができません。たとえば，

|: a |: b |1 c :||2 d e :|

という記述では正常に演奏することができません。これは，

|: a |: b |1 c :||2 d e :| :|

としなくてはなりません。

多重ループを使用する際は|:に対応した:|が絶対必要ということなのです。

FM PCM MIDI

[~]コマンド

[do] ～ [loop]	囲まれた間を半永久的にループ演奏する
[d.c.]	始めに飛ぶ（1回のみ有効）
[d.s.]	[segno]または[$]へ飛ぶ（1回のみ有効）
[segno]（または[$]）	
[tocoda]（または[＊]）	[coda]へ飛ぶ（1回以上有効）
[coda]	
[fine]（または[^]）	[d.c.],[d.s.]を実行したあとにここで演奏終了する

このほかに演奏データ作成を支援するコマンドがある(これらのコマンドはm_debug(1)または(D1)のときのみ有効となる。m_debug(0)または(D0)のときは無視される。MEASURE3 m_debug()コマンド，MEASURE4 (Dn)コマンド参照)。

[!]

次の[!]までジャンプする。ジャンプ後の音源やZ-MUSICの内部パラメータはそこまで通常に演奏を行ったときと同じになっている。各トラックごとに設定が行える。[!]が偶数個設定されなかった場合は正常に動作しない。

[@]

次の[@]までジャンプする。ジャンプ後の音源やZ-MUSICの内部パラメータはそこまで通常に演奏を行ったときと同じになっている。ほぼ[!]と同機能だが，あるひとつトラックで設定するとほかのトラックも同時にジャンプを開始する点が違う。[@]が偶数個設定されなかった場合は正常に動作しない。

[!]/[@]の動作

[!]/[@]の動作フローを以下に示します。

1) 音階/和音/ポルタメント以外の「発音しない」コマンドを実行
2) [!]/[@]を見つけるまで1)を繰り返す
3) [!]/[@]を発見
4) 通常演奏へ

[end]

いかなる状態のときにも，これを発見するとそのトラックは演奏を終える。

■MIDI楽器制御命令

MIDI

@In_1,n_2,n_3

Z-MUSICにそのトラックの楽器のメーカーID，デバイスID，モデルIDを設定する。各値はMIDI楽器のマニュアルを参照のこと。これは特に設定しなくても通常のコマンドの動作にはまったく影響しない。ただし，

@E命令
X命令

に関しては楽器個別の動作をするため，これらのコマンドを使用する前には必ず設定しなければならない。

n_1＝メーカーID
(例:ROLAND＝$41,KORG＝$42,YAMAHA＝$43)
n_2＝デバイスID
(各ユーザーが楽器に設定した値。特に設定した覚えのない場合はその楽器のデフォルトのデバイスIDを設定する)
n_3＝モデルID
(例:MT-32＝$16,SC-55＝$42 or $45)

初期値は不定。

MIDI

Xn_1…,n_i

ローランド系の楽器にエクスクルーシブメッセージ送信を行う。@Iコマンドでメーカー IDにローランドを設定しておくことが前提。現在，ワンウェイコミニュケーションのデータセット(コマンドID＝$12)のみ対応。

データはいくつでも記述可能。各値の設定範囲は0≦n_i≦127であるが，これ以上の値を設定した場合は7ビットずつに分割されて送信される(例:8192→$00,40のように下位，上位の順に送信される)。もちろんチェックサムは自動的に送信してくれる。

具体的な用途としてはリバーブパラメータの書き換えや音色のパラメータ単位の書き換えを行うのに便利であるが，楽器のメモリなどを直接操作するものなので，各自の責任のもとで使用すること。

MIDI

@Xn_1,…,x_i

MIDIの生データの送信を行う。もちろんメーカーや楽器の種類には無関係でチェックサムなども送信しない。楽器のメモリなどを書き換える場合もあるので各自の責任の下で使用すること。

MIDI

@Er,c

MIDI楽器のエフェクトをコントロールする。コントロールチェンジのエフェクト1へrを，エフェクト3へcを出力する。ただし，楽器側が対応していない場合はコントロールできない。

パラメータは片方のみ省略できる。パラメータの有効範囲は0≦r，c≦127。初期値は不定。

MIDI(MT-32系専用命令)

@En_1,n_2

@Iコマンドで IDをMT-32に設定してあることが前提。パートn_1のエフェクトスイッチを設定する。n_1はパートナンバーを表し，設定範囲は1≦n_1≦8。n_2はスイッチを表しn_2＝0がオフ，1がオンである。

MIDI

@Ya_1,a_2,d_1,d_2

MIDI楽器のNRPNを設定する。

MIDI楽器にはMIDI規格には定められていないレジスタを持つことが許されており，このコマンドでそのレジスタへ書き込みを行う。

a_1,a_2はそれぞれアドレスの上位下位を表し，d_1,d_2はそれぞれデータの上位下位を表す。設定範囲はすべて0～127でd_2のみ省略可能。

SC-55では音色のコントロールが可能(SC-55マニュアルp75参照)。

■AD PCM制御命令

PCM

@Fn

AD PCM音源の再生周波数を変更する。設定範囲は0≦n≦4で，

n＝ 0: 3.9kHz
1: 5.2kHz
2: 7.8kHz
3:10.4kHz
4:15.6kHz

に対応する。初期値は4で値省略時は4に自動設定される。

PCM8独立チャンネルモード時には，以下の設定が可能になる。

n＝ 5:16ビットPCMデータ方式
6: 8ビットPCMデータ方式

■ノイズ発生

FM

@On

FM音源のチャンネル8のオペレータ4(スロット32)からノイズを発生する。nとノイズ周波数の関係は，

n＝4000/(32×ノイズ周波数[kHz])

で，設定範囲は0≦n≦31。値省略時はノイズモード解除とみなす。

■FM音源レジスタへの書き込み命令

FM PCM

Ya,d

FM音源のレジスタaへ値dの書き込みを行う。AD PCM音源部で使用した場合もFM音源のレジスタへの書き込みが行われる。使用にはFM音源に関しての知識と十分な理解が必要。

またOPMD.Xで有効だった特殊Yコマンドが使用可能である。

y2,n　ノート番号n番のAD PCM音を鳴らす。0≦n≦511
y3,n　AD PCMのパンを設定する。
n=0:OFF，1:LEFT，2:RIGHT，3:MIDDLE
y13,n　AD PCMの再生周波数を変更する(n:@Fに準拠)
y14,n　AD PCM発音モードの設定。
n=0のとき，あとから発音要求があったものを優先

して発音していく。
n=1のとき，現在発音中のAD PCM音が鳴り終えるまであとからくるAD PCMの発音要求を無視する。

■コントロールチェンジ

MIDI

Ya,d

コントロールチェンジa番に値dを書き込む。パラメータa,dの設定範囲は共に0≦a,d≦127。コントロールチェンジについての詳細はMIDI楽器のマニュアルのインプリメンテーションチャートを参照。

■演奏制御

FM PCM MIDI

W

ほかのトラックから同期信号がくるまで演奏処理を中断する。

FM PCM MIDI

Wn

同期待ちしているトラックに対して同期信号を送る。値の設定範囲は1≦n≦80。

例

```
(t1)   @1 v15 o4 q8 cdef w2  ←トラック2へ同期送信
(t2) w @1 v15 o4 q8 gab<c
       ↑どこからか同期信号が来るまで演奏を中断
```

この例では結果的にはcdefgab<cが演奏される。

同期コマンドと強制演奏コマンド使用における注意

コンパイルによってZMDを得たとき，トラック番号が最適化されるため，ZMS時で同期が取れていたものがZMDだと希望どおりに鳴らないことがあります。たとえば，

```
(m1,1000)(a1,1)
(m2,1000)(a2,2)
(m3,1000)(a3,3)
(t1)   @1 v15 o4 q8 cdef w3
(t2)
(t3) w @1 v15 o4 q8 gab<c
```

の場合，コンパイル時の最適化によってトラック2は消去されトラック3がトラック2になってしまうためトラック1の同期信号は意味をなさなくなります。また，

```
(m1,1000)(a1,1)
(m2,1000)(a2,2)
(m3,1000)(a3,3)
(t1)
(t2)   @1 v15 o4 q8 cdef w3
(t3) w @1 v15 o4 q8 gab<c
```

の場合も同様にトラック1が消去されトラック2が1，トラック3が2へと詰められるので同期コマンドは意味を持たなくなります。ですから，同期コマンドを使用した曲をコンパイルしてZMDを得る場合は無意味なトラックを確保してはいけません。

同様に強制演奏コマンドの場合も，参照したトラックとの間にダミーのトラックが存在すると，同様の症状が起きます。

FM PCM MIDI

Jn

トラックnを強制的に初めから演奏開始にする。演奏が終了していても，演奏中であっても演奏開始を命じられたトラックは最初から演奏を始める。値の設定範囲は1≦n≦80。

例

```
(t1)   @1 v15 o4 q8 j2r2j2
(t2)   @1 v15 o4 q8 l8cdef
```

この例では結果的にはcdef cdefが演奏される。

■特権命令

FM PCM MIDI

;n

直接ZMDコードnを埋め込む。使用にはZMDの十分な理解を必要とする。不当な与え方をした場合は最悪，暴走する。値の設定範囲は0≦n≦255。

FM PCM MIDI

?a,d

そのトラックの演奏ワークのa番に値dを書き込む。使用にはZ-MUSICのワークエリアについての十分な理解を必要とする。不当な与え方をした場合は最悪，暴走する。値の設定範囲は0≦a,d≦255。

5.3 モジュレーション波形について

●プリセット波形

Z-MUSICにはピッチモジュレーション用(以下PM)，アンプリチュードモジュレーション用(以下AM)両方に4つのプリセット波形が登録されています。この波形が振幅(@M/@Aコマンド)，波長(@Sコマンド)，ディレイ(@Hコマンド)などの各パラメータによってどう変化するかを図に示しておきましょう。

●波形メモリについて

Z-MUSICでは，

●ZMSコマンド

.wave_form <wave number>,<loop type>[,loop point]={data.....}

●MUSICZ.FNC関数

m_wave_form(<wave number>,<loop type>[,loop point],<ary1:data>)

を用いて作成したユーザー波形を波形番号8～31に登録し使用することができます。この波形が振幅(@M/@Aコマンド)，波長(@Sコマンド)，ディレイ(@Hコマンド)などの各パラメータによってどう影響するかを図に示します。

また以下にユーザー波形使用時の留意点を挙げておきます。

・波形データはピッチモジュレーション/音量モジュレーション/ARCC共用となる。登録した波形データはMMLの'S'コマンドでプリセットの鋸歯波/矩形波/三角波同様に呼び出して使用する。
・MML'@S'で指定されたクロックの間，ひとつの波形データを使用する(下図)。
・実際の波形実行のON/OFFは通常の特殊機能と同様。
・振幅は各ポイントにおける波形データが直値で使用されるが，各特殊機能コマンドの振幅設定コマンドの正負に応じて各値の正負を逆転して使用させることも可能。ただし振幅の絶対値は無関係。
・1/8モードは波形メモリに振幅という観念がないため無効。
・ZMSで登録された波形データはZ-MUSICの組み込みスイッチ'-W'で指定される汎用ワークへ格納される。ZMDの場合は汎用ワークエリアは無関係。

スイッチングオフについて

各特殊機能の制御MMLにパラメータなしや0を設定するとその機能は停止することになります。たとえば@Mまたは@M0でピッチモジュレーション機能を解除することができます。またはスイッチングコマンド'='を使って制御することもできます。この2つには微妙な違いがあるので注意してください。

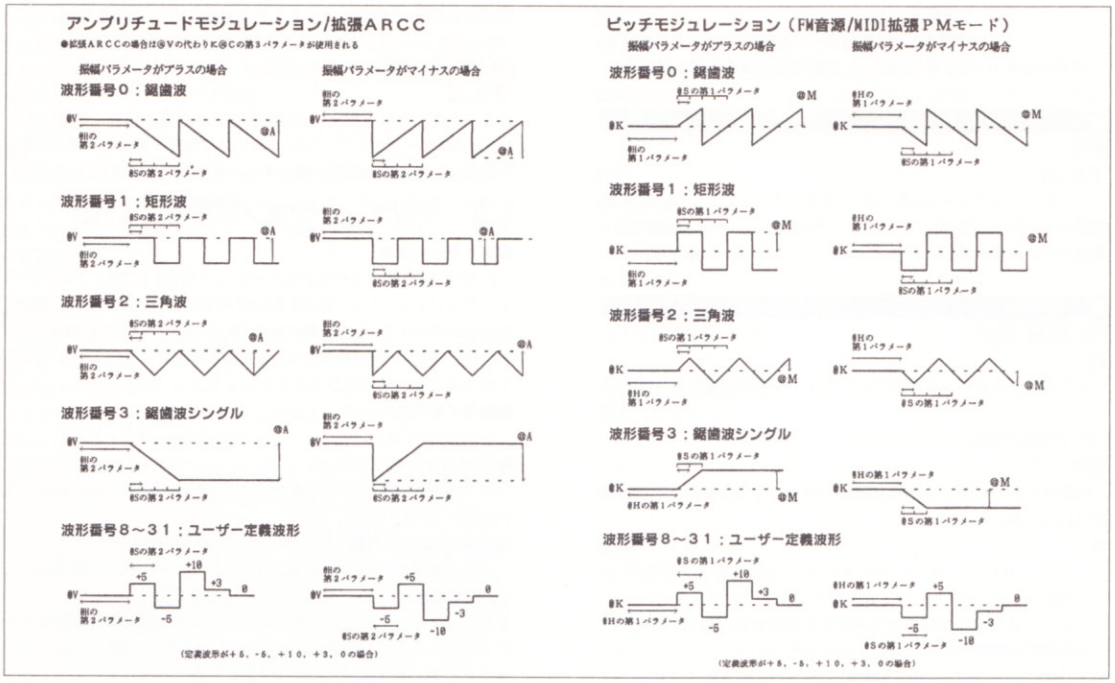
まず，前者の場合では機能停止時になんらかの初期化措置を音源/楽器側に対して実行します。たとえばMIDIで@M0とした場合はモジュレーション処理を終了するとともに楽器側へモジュレーションデプス0を送信しています。

一方後者の場合は純粋にZ-MUSICがモジュレーション処理を中断するだけで楽器に対しては，なんら特殊な処置をしません。ですからたとえばMIDIで，

```
@h40 @m127 c1 @m d1
@h40 @m127 c1 =0 d1
```

とした場合，両者ともcがディレイ40クロックのあとモジュレーションを開始します。そして，次のdが前者と後者では食い違いが出てきます。前者ではdはモジュレーションしないで普通に鳴ります。しかし，後者のほうはZ-MUSICが「ディレイ40クロックのあとモジュレーションを開始」という処理を止めただけで前の音符のcのモジュレーションがそのまま残ることになりdはモジュレーションがかかったまま鳴ります。

こうしたことから'='によるスイッチングは前の状態を保存しつつ，特殊機能を停止したいというときに用いると便利です。



OPMDRV.Xとの互換性について

ZMUSIC.Xは初代のOPMDRV.Xと互換性がありますが以下の点に注意してください（OPMDRV3.Xとは互換性はありません）。

レジスタ番号48～55を操作するYコマンドは都合上@Kコマンドに変換されるため，アサインされたチャンネル以外のチャンネルのキーフラクションを操作した場合は無効となります。

また，ひとつの命令における中間言語やそのサイズが（当然ながら）OPMDRV.Xとは違うのでM_ALLOC命令や(Mtr,sz)コマンドでトラックバッファをギリギリに確保してあるような曲は，トラックバッファが不足してしまうことがあります。そういう場合はトラックバッファを大き目にとり直してください。

m_alloc(1,1000) → m_alloc(1,5000)
(m1,1000) → (m1,5000)

OPMDRV.Xでは一度トラックヘッダを書けば以降同じトラックへMMLをセットする場合はトラックヘッダの省略ができました。もちろんZ-MUSICでも可能は可能なのですが一部制約があります。それはポルタメント命令です。

ポルタメントは"()"という記号を使うためトラックヘッダを省略してこの命令を書いた場合，Z-MUSICのインタプリタはMMLとは認識できずZMSコマンドと勘違いしてしまうのです。それ以外の命令ならばトラックヘッダを省略しても大丈夫です。

さらにOPMDRV.Xでは'm_play()'や'm_cont()', 'm_stop()'などのパラメータにチャンネル番号を指定するような仕様でしたがZ-MUSICではトラック番号で指定します。こちらのほうが汎用性が高いと判断して採用しました。

波形メモリの応用例

・FM音源のピッチモジュレーションで使う場合

波形データは半音を64とした値で処理されます。各値がそのチャンネルのディチューンと加算されて，それがその瞬間のピッチとして決定されます。

たとえば，ディチューンが10のときに波形メモリのデータが0,5,-5のときは10+0=10, 10+5=15, 10−5=5がそれぞれその瞬間のピッチになります。

例

(t1) @m1 @s6 s8 @h40
順位相で波形メモリ8番をFM PMで実行
(t1) @m-1 @s6 s8 @h40
逆位相で波形メモリ8番をFM PMで実行

●FM音源のアンプリチュードモジュレーションで使う場合

波形データはそのチャンネルのボリューム値に加算され，その和をその瞬間の音量として決定されます。音量の範囲は0～127ですがこの範囲を越えた場合はこの範囲内に修正されます。たとえば，音量が@V125のとき波形メモリのデータが0,5,-5のときは125+0=125，125+5=127(130)，125−5=120がそれぞれその瞬間の音量になります。

例

(t1) @a1 @s,6 s,8 @h,40
順位相で波形メモリ8番をFM・AMで実行
(t1) @a-1 @s,6 s,8 @h,40
逆位相で波形メモリ8番をFM・AMで実行

●MIDI音源のピッチモジュレーションで使う場合

拡張モジュレーションモード時のみ有効です。効果はFM音源と同じです。拡張モジュレーションモードには振幅を半音＝64にするか半音≒683にするかの2つのモードがありますが波形データはこの影響を受けます。たとえば波形データが683であったとすると半音＝64モード時には683/64≒10半音分の変化量になりますが，半音≒683モード時には1半音分の変化量となります。用途や趣味にあわせて使い分けてください。

例

(t1) m1 @m1 @s6 s8 @h40
順位相で波形メモリ8番をMIDI・拡張PMで実行
(t1) m1 @m-1 @s6 s8 @h40
逆位相で波形メモリ8番をMIDI・拡張PMで実行

●MIDI音源の拡張ARCCで使う場合

波形メモリは拡張ARCCモード時にのみ有効です。各波形データはARCCニュートラル値(@Cの第3パラメータ)と加算して，その和がコントロールチェンジへ出力されます。応用次第で疑似ランダムオートパンポットやFM音源のような自由波形による音量モジュレーションなども実現できます。

例

(t1) m,1 @a1 @s,6 s,8 @h,40
順位相で波形メモリ8番をMIDI・拡張ARCCで実行
(t1) m,1 @a-1 @s,6 s,8 @h,40
逆位相で波形メモリ8番をMIDI・拡張ARCCで実行

MEASURE6
AD PCMを扱う

ここではZ-MUSICシステムでのAD PCMデータの取り扱いと，支援ツール「ZPCNV.R」について解説します。

6.1. はじめに

ZMUSIC.X起動時にはAD PCM音はひとつもプリセットされていません。よってAD PCM音を鳴らすためにはZ-MUSICに鳴らしたいAD PCM音ファイルを登録することが必要です。

Z-MUSICにAD PCMデータを渡す方法は3通りあります。ひとつはAD PCM登録コマンドをミュージックプログラムに直接記述していく方法です。

2つめはAD PCM登録ZMSコマンドを記述したファイルをエディタ（ED.Xなど）で作成し，これを渡す方法です。

3つ目はそのファイルをZPCNV.RでZPDファイルにコンバートし，これをZ-MUSICに渡す方法です。

6.2. AD PCMコンフィギュレーションファイル

AD PCM登録ZMSコマンドを箇条書きにしたファイルを特に「AD PCMコンフィギュレーションファイル」と呼び，これをZ-MUSICへ受け渡すことでAD PCMデータを登録することができます（奨励ファイル拡張子は'.CNF'）。

AD PCM登録ZMSコマンドは，

〈音階〉=〈AD PCMファイル名〉[,AD PCM加工オプションスイッチ]

のスタイルをとりますが（下図，MEASURE4参照），この各パラメータについて詳しく解説します。

●AD PCMコンフィギュレーション

n ファイルネーム,Pp,Vv,Mm,d,Cc,s,R,Ff,l

AD PCM音の登録

n＝設定音色番号（0～511）
p＝ピッチシフトパラメータ（－12～12）
v＝ボリュームパラメータ（1～100（原音量）～300）
m＝ミキシングノート番号（0～511）
d＝ミキシングディレイパラメータ（0～65535）
c＝カットオフセットパラメータ（0～65535）
s＝カットサイズ（0～65535）
f＝フェードイン/アウトオフセット（0～65535）
l＝フェードイン/アウトレベル（0～127）

ファイルネーム以降は省略可能。
（MEASURE3 m_pcmset()参照）

.ADPCM_BANK n

AD PCM音の登録先のバンクを指定
n＝バンク番号（1≦n≦4）

.Onk ファイル名,Pp,Vv,Mm,d,Cc,s,R,Ff,l

AD PCM音の登録

n＝オクターブ値（－1～9）
k＝音階MML（abcdefg,#,＋,－）
p＝ピッチシフトパラメータ（－12～12）
v＝ボリュームパラメータ（1～100（原音量）～300）
m＝ミキシングノート番号（0～511）
d＝ミキシングディレイパラメータ（0～65535）
c＝カットオフセットパラメータ（0～65535）
s＝カットサイズ（0～65535）
f＝フェードイン/アウトオフセット（0～65535）
l＝フェードイン/アウトレベル（0～127）

ファイルネーム以降は省略可能。登録先ノート番号やミキシングノート番号は'.ADPCM_BANK'命令のバンク番号が考慮されます。
（MEASURE3 m_pcmset()参照）

●音階

AD PCM音を鍵盤上のどこに割り当てるのかを指定するパラメータ。先頭にピリオド'.'またはシャープ'#'をつけてオクターブMMLと音階MMLを記述することによって指定する方法と，0～511の登録先ノートナンバーを書いて登録する方法があります。

音階MML指定の場合はオクターブ－1のドからオクターブ9のソ（.O-1C～.O9G）までの128から選択して指定できます。

Z-MUSICではAD PCMは4つのグループ（バンク）に分けて管理ができ，音階で登録するときはこのバンク内への登録となります。つまり，音階MML指定の場合には，まず，登録先のバンクを決定し，それからそのバンク内の音階を選択するというプロセスで行います。同一バンク内の登録は一度バンク決定コマンドを実行すれば2回目は不要です。

ノートナンバー指定の場合はこのセレクトしたバンクを超えた登録が可能です。以下にその対応を示します。

AD PCMノート番号＝
（バンク番号－1）×128＋（オクターブ＋1）×12＋音階
AD PCMノート番号：0～511
バンク番号：1～4
オクターブ：－1～9
音階：CDEFGABがそれぞれ0～11に対応

	AD PCMノート番号
バンク1	0～127
バンク2	128～255
バンク3	256～383
バンク4	384～511

●AD PCMファイルネーム

登録したいAD PCMファイルをここに指定します。

AD PCMのファイルは環境変数"zmusic"にパスを記述しておけば（囲み参照），カレントからファイルが見つからない場合はそちらからも検索してくれます。

また，ここに音階MMLやノートナンバーを書くことによって，すでに登録したAD PCM音を参照することができます。同じ音を二重に登録したり，AD PCM音を二重に加工したい場合などに便利です。

●AD PCM加工オプションスイッチ

単に記憶デバイスにあるAD PCMファイルを登録するだけでなくZ-MUSICではそのデータに加工を施して登録することができます。

・音程変更

'P'（小文字も可）の後ろに－12～＋12の数値を書くことによってAD PCM音の音程を半音単位で変えることができます。

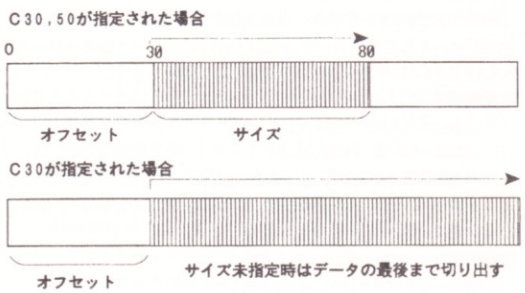
・音量変更

'V'（小文字も可）の後ろに1～300の数値を書くことによってAD PCM音の音量をパーセント単位で変えることができます。

・切り出し

'C'（小文字も可）の後ろにオフセット値と切り出しサイズを記述することによってAD PCMデータ中の任意の一部分を取り出すことができます。

サイズを省略するとオフセットより後ろすべてを切り出すことになります。



・逆転

'R'（小文字も可）を書くことによって逆転再生を指定することができます。

・フェードイン/フェードアウト

'F'(小文字も可)の後ろにオフセット値とスタートレベル(フェードイン時),またはエンドレベル(フェードアウト時)を設定すると波形のエンベロープを変更することができます。オフセットの値を正にすることでフェードアウト,負にすることでフェードインの指定となります(P.51の図参照)。

パラメータは両方とも省略可能ですが,ただしその場合は両パラメータとも0が設定されます(「データ先頭からフェードアウト」に相当)。

オフセットパラメータが正の場合(フェードアウト)

オフセットパラメータが負の場合(フェードイン)

・合成

'M'(小文字も可)の後ろに0～511の数値または音階MMLを書くことによってそのノート番号や音階に以前登録されたAD PCM音と合成することができます。

さらにその後ろに ,で区切ったあとで数値を書くとそのバイト数分遅らせて合成することが可能です。

●注意

AD PCM加工オプションスイッチは,いずれも省略可能で,どれかひとつのみの指定でもかまいません。また加工のプロセスは各スイッチの設定順序には無関係で図のような処理順序で行われます。

音程・音量変更
↓
切り出し
↓
逆転
↓
フェードイン/フェードアウト
↓
合成

また,AD PCM登録コマンドでZ-MUSICにAD PCMを登録するとき,AD PCM加工スイッチを用いる場合にはZ-MUSICの組み込みスイッチの'-W'でワークエリアを十分に取っておかないとエラーとなります(目安としてはコンフィギュレーションファイル内に存在する最大ファイルサイズの4倍くらい。ミックスを用いる場合は更にその2倍くらい)。エラーとなった場合は加工も指定されたノートへの登録も行われません。

後述のZPCNV.RでZPDを作成する場合にはZ-MUSICは無関係なのでAD PCMバッファやワークエリアの確保は不要です。

●使用例1

```
.adpcm_bank 2
.o2c=BASS.PCM
.o2d=SNARE.PCM,v150
```

バンク2のオクターブ2のC(AD PCMノート番号=164)に"BASS.PCM"を,そしてオクターブ2のD(AD PCMノート番号=166)に"SNARE.PCM"をボリューム150%にして登録します。

●使用例2

```
.o2d=SNARE.PCM,v150,p+4
.o2d#=CLAP.PCM,mo2d
```

"SNARE.PCM"をボリューム150%にし,さらに4半音下げたものをオクターブ2のDへ登録。そしてこれと"CLAP.PCM"を合成したものをオクターブ2のD#へ登録します。

●使用例3

```
.o3d=CYMBAL.PCM,c1000,5000,r
```

"CYMBAL.PCM"の1000バイト目から5000バイトを切り出し,これを逆転したものをオクターブ2のDへ登録します。

●使用例4

```
.o2e-=VOICE.PCM,f1000,20,m200,2000
```

"VOICE.PCM"の1000バイト目からデータ終端までを音量20まで滑らかに下げていき,これに対してノート番号200(バンク2のオクターブ5のC)に登録されたデータを2000カウント遅らせて合成してオクターブ2のEbへ登録します。

●AD PCMコンフィギュレーションファイルの登録法

MUSICZ.FNCを用いてX-BASIC上で作られた曲データ中に指定したい場合は,

```
m_pcmcnf("AD PCMコンフィギュレーションファイル名")
```

を使います(MEASURE3参照)。

ZMSファイル中に指定したい場合は,

```
.adpcm_list AD PCMコンフィギュレーションファイル名
```

を用います(MEASURE4参照)。

また,通常の演奏データのように,

```
A>copy AD PCMコンフィギュレーションファイル名 opm
```

としたり,

```
A>zp AD PCMコンフィギュレーションファイル名
```

のようにZ-MUSICへ登録することも可能です(MEASURE7参照)。こちらの場合は曲がたとえ演奏中であってもリアルタイムにAD PCMセットを変更することができます。

また,ZMUSIC.X常駐スイッチの'-S'を用いて,

```
A>zmusic -w50 -p200 -Sdrum.cnf
```

のようにZMUSIC.X常駐時にAD PCMコンフィギュレーションファイルを登録することもできます(拡張子の省略は不可)。ただし,その際には必要分のワークやAD PCMバッファを確保しなければなりません。

6.3. ZPCNV.RとZPD

ZPCNV.Rは,前節で解説したコンフィギュレーションファイルに従って必要なAD PCMデータをひとつのファイルにまとめてしまうアプリケーションです。ひとまとめにする際に音程音量変換,合成作業といった加工処理もしてしまうためAD PCMデータの登録処理にかかる時間は極めて少なくて済みます(AD PCMコンフィギュレーションファイルをZ-MUSICへ渡す方式では1個1個のデータを読み込み,それから加工を行ったりするため,大量のAD PCMデータを登録する場合には,フロッピーディスクなどの中/低速メディアユーザーや10MHzマシンユーザーは処理が終了するまで長時間待たされることがある)。加工処理ルーチンやその際に使用するワークはZPCNV.Rが独自で持っているため,実行にはZMUSIC.Xの常駐は必要ありません。

Z-MUSICではAD PCMコンフィギュレーションファイルをもとに,ひとまとめにしたこのデータをAD PCMブロックデータ,「ZPD」と呼びます。

●ZPCNV.Rの使い方

AD PCMコンフィギュレーションファイルからZPDデータを作成するには,

```
A>zpcnv filename1 filename2
```

とするだけでかまいません。ワークエリアなどの指定は一切不要です。ZMUSIC.Xも常駐させる必要がありません(メモリが足らないときはZMSUIC.Xをはずしてから実行するとよい)。

filename1はコンバートしたいコンフィギュレーションファイル名です(拡張子を省略したときは'.CNF'が自動添付される)。

filename2はできあがったブロックデータのファイルネームでこちらは省略することが可能です。省略時はfilename1の拡張子を'.ZPD'にしたものでセーブされます。

●ZPCNV.R専用ZMS命令

たとえば,

```
.o2d=SNARE.PCM,v150,p+4
.o2d#=CLAP.PCM,mo2d
```

("SNARE.PCM"をボリューム150%にし,さらに4半音下げたものをオクターブ2のDへ登録。そしてこれと"CLAP.PCM"を合成したものをオクターブ2のD#へ登録する)というケースで,得たいのが.o2d#のほうのみで.o2dは単なるテンポラリ的な存在でしかなかったときに,

```
.erase o2d
```

をコンフィギュレーションファイル後尾に設定すると，そのデータを除外してZPDデータを生成してくれます。これにより不本意にZPDが大きくなるのを防ぐことができます。

本命令はZPCNV.R専用のZMS命令でZ-MUSICの曲データ中に指定しても（エラーにはならないが）機能はしないダミー命令として扱われます。

●ZPDデータのZ-MUSICへの登録法

MUSICZ.FNCを用いてX-BASIC上で作られた曲データ中に指定したい場合は，

m_adpcm_block("AD PCMコンフィギュレーションファイル名")

を使います(MEASURE3参照)。

ZMS中に指定したい場合は，

.adpcm_block_data AD PCMコンフィギュレーションファイル名

を用いてください(MEASURE4参照)。

また，通常の演奏データのように，

A>copy AD PCMコンフィギュレーションファイル名 opm

としたり，

A>zp AD PCMコンフィギュレーションファイル名

のようにZ-MUSICへ登録することも可能です(MEASURE7参照)。こちらの場合は曲がたとえ演奏中であってもリアルタイムにAD PCMセットを変更することができます。

また，Z-MUSIC常駐スイッチの'－B'を用いて，

A>zmusic -Bdrum_set.zpd　　(拡張子の省略は可能)

のようにZ-MUSIC常駐時にZPDを登録することもできます。ワークなどの指定は不要で，AD PCMバッファの確保は自動的に行われます。

6.4. AD PCMバッファが0のとき，不足しているとき

AD PCMバッファを確保しなかった場合はAD PCM関連の命令を用いるとエラーになります。ZPDを登録しようとしたときにバッファが必要分より少ない場合はエラーとなりますが，AD PCM登録コマンドでZ-MUSICにAD PCM音を登録しようとしたときにバッファが不足しているときはいちばん古く登録したものが切り捨てられ登録されます。

6.5. PCMファイルリンカ「ZPLK.R」

Z-MUSICシステムver.2.0より，AD PCMデータの加工&リンクツール「ZPLK.R」を正式な支援ツールとして追加しました。

●その機能と特長

・最大32個の複数のファイルをつなげてひとつのファイルにして出力できます。ひとつのファイルにつき最大65535回の反復指定が可能です。反復パターンは4種類あります。

・単一ファイルをいくつもつなげてひとつの大きなファイルにすることも可能です。

・入力ファイルとして16ビットPCMファイル，AD PCMファイルを与えることができます（混在も可能)。

・リンク作業の前に各入力ファイルのサブデータコンバートを行うことができるので，16ビットPCMにAD PCMファイルをリンクさせたり，また，その逆も可能です。

・出力ファイルはその出力直前に16ビットPCMデータコンバート，AD PCMデータコンバート，逆転再生，音量変換，周波数変換，ポルタメント，エンベロープ変更など多彩な加工処理を行うことができます。

・インパルスデータを与えることにより，畳み込み演算を行うことができます。これにより，任意の音声データに対してホールの残響効果やボコーダー処理などの本格的なエフェクト処理をすることができます。

●文法

ZPLK.R 〈[オプション1] 入力ファイル1〉[[オプション2～32] 入力ファイル2～32] 〈出力ファイル名〉

入力ファイルネームは最低1個は必要です。書き込みファイル名は'PCM'に設定するとファイルを書き出さず，音声として出力できます(ZMUSIC.XやPCMDRV.SYS組み込み時)。ファイルネームの指定は拡張子やパスの省略をせずきちんと書いてください。ファイルがカレントより見つからない場合は環境変数'zmusic'に書かれたパスに従ってファイルを検索します。

●オプションスイッチ

-A

出力ファイルを出力する前にそのデータを16ビットPCMデータとみなしてAD PCMデータへ変換します。

-Bf,d,p,s

出力ファイルを出力する前にそのデータを16ビットPCMデータとみなしてポルタメントを行います(連続的に滑らかに周波数変更を行う)。

f　開始時の周波数をfHzとみなす。ただしsは1～65535

d　終了時の周波数をdHzとする。ただしdは1～65535

p　出力ファイルのpバイト目から周波数変換を行う。省略時は0

s　変換するサイズをsバイトとする。省略時はpで指定された位置から後ろ全部を変換対象領域とする。pも省略した場合は出力全域に対して変換処理を施す

-Cp,s

出力ファイルを出力する前にそのデータの任意の一部分を摘出しそれを出力します。

p　出力ファイルのpバイト目から切り出す。省略時は0

s　切り出すサイズをsバイトとする。省略時はpで指定された位置から後ろ全部を切り出す。pも省略した場合はエラーとなる

-Fs,l,m

出力ファイルを出力する前にそのデータを16ビットPCMデータと見なしてエンベロープの形状を変化させます。

s　出力ファイルのsバイト目からエンベロープ形状を変更する。省略時は0

l　エンベロープ形状初期音量(m=0時)/最終音量(m=1時)を0～127の128段階で設定する。省略時はl=0

m　エンベロープの変更パターンを設定する。省略時はm=1

0 フェードイン型

1 フェードアウト型

-Ifilename

インパルスデータのファイルネームを与えると出力ファイルに出力する前にそのデータとの畳み込み演算を行います。ただし，与えられたファイル名の拡張子を'.P16'とした場合はそのインパルスデータを16ビットPCMデータ，'.PCM'とした場合はAD PCMデータとしてみなします。

-P

出力ファイルを出力する前にそのデータをAD PCMデータとみなして16ビットPCMデータへ変換します。

-R

出力ファイルを出力する前にそのデータを16ビットPCMデータとみなして逆転させます。

-Vn

出力ファイルを出力する前にそのデータの音量をパーセント単位で指定します。nの範囲は0～300。スイッチ無指定や値省略時は100とみなします。

-Ti,o,p,s

出力ファイルを出力する前にそのデータを16ビットPCMデータとみなして周波数の変更を行います。

i　元の周波数をiHzとみなす。ただしiは1～65535

o　変換後の周波数をdHzとする。ただしdは1～65535

p　出力ファイルのpバイト目から周波数変換を行う。省略時はp=0

s　変換するサイズをsバイトとする。省略時はpで指定された位置から後ろ全部を変換対象領域とする。pも省略した場合は出力全域に対して変換処理を施す

-Xl,r,t

リンク制御

l　ループタイプの設定。省略時 l=0

0 そのまま
1 正順，逆順の順にループさせる
2 逆順，正順の順にループさせる
3 入力ファイルの逆順をループさせる

r　反復回数の設定。省略時 r=1。範囲1～65535。

t　リンク作業の前に入力ファイルに対して加工を行う。
省略時 t=0
0 なにも行わない
1 入力ファイルを16ビットPCMデータとみなしてAD PCMデータへ変換する
2 入力ファイルをAD PCMデータとみなして16ビットPCMデータへ変換する

注意　複数の変換を指定した場合は以下の手順で各変換を行う。

音量変更周波数変換
↓
ポルタメント
↓
切り出し
↓
逆転
↓
フェードイン/フェードアウト
↓
インパルス演算
↓
16ビットPCM/AD PCM変換

国際基準イ＝a1=440Hzに基づく12平均率音階周波数表
(試験に出るX1より転載)

	O0	O1	O2	O3	O4	O5	O6	O7	O8
C	16.352	32.703	65.406	130.81	261.63	523.25	1046.5	2093.0	4186.0
C#	17.324	34.648	69.296	138.59	277.18	554.37	1108.7	2217.5	4434.9
D	18.354	36.708	73.416	146.83	293.66	587.33	1174.7	2349.3	4698.6
D#	19.445	38.891	77.782	155.56	311.13	622.25	1244.5	2489.0	4978.0
E	20.602	41.203	82.407	164.81	329.63	659.26	1318.5	2637.0	5274.0
F	21.827	43.654	87.307	174.61	349.23	698.46	1396.9	2793.8	5587.7
F#	23.125	46.249	92.499	185.00	369.99	739.99	1480.0	2960.0	5919.9
G	24.500	48.999	97.999	196.00	392.00	783.99	1568.0	3136.0	6271.9
G#	25.957	51.913	103.83	207.65	415.30	830.61	1661.2	3322.4	6644.9
A	27.500	55.000	110.00	220.00	440.00	880.00	1760.0	3520.0	7040.0
A#	29.135	58.270	116.54	233.08	466.16	932.33	1864.7	3729.3	7458.6
B	30.868	61.735	123.47	246.96	493.88	987.77	1975.5	3951.1	7902.1

(単位はHz)

●パラメータの特殊設定

周波数変換(-Tスイッチ)，ポルタメント変換(-Bスイッチ)において，2つの周波数パラメータの設定をする代わりにZ-MUSICシステムで御馴染みの半音ステップの変換パラメータで設定することもできます。

たとえばある音を半音下げたいときには，

```
-Tp-1
```

また，半音上げたいときは，

```
-Tp+1
```

とします。また，ポルタメントのときも同様で，

```
-Bp-10
-Bp+6
```

といった記述ができます。サブスイッチ'p'の値の範囲は-144～＋144です(ただし0は除く)。

●使用例

```
A>zplk a.p16 -x0,10 b.p16 -a c.pcm
```

a.p16というファイルの後ろにb.p16を10回くっつけて，AD PCMへコンバートしc.pcmというファイル名でセーブします。

```
A>zplk -x0,5 a.p16 -x,,2 b.pcm -v80 c.p16
```

a.p16 というファイルを5回繰り返し，これにb.pcmを16ビットPCMへ変換したものをくっつけて，それを音量を80%に設定しc.p16というファイル名でセーブします。

8，16ビットPCMデータ方式とは?

X68000の扱うAD PCMデータとはいわばサンプリングしてできたPCMデータが一種のアルゴリズムで圧縮されていると考えられます。音量の変更や周波数変換，合成処理を行うにはこのAD PCMデータ方式からPCMデータ方式へ変換しなければなりません。PCM8ではこの変換処理を割り込みでリアルタイムで行っていますが，もしこの処理を省けるならばかなり割り込み処理を軽くすることができるはずです。そこでPCM8では無圧縮のPCMデータを演奏できるようになっています。

16ビットPCMデータ方式は符号付き16ビット整数により表現されたデータです(－32768～32767)。同一音長のAD PCMデータの4倍のデータ長になります。ちょうどZVT.Xで'－C'スイッチにより生成されるPCMデータと同じものです。ひとつのデータが2バイト構成であることから必ずデータ長は偶数でなければ正常に鳴りません。

8ビットPCMデータは方式は同様に符号付き8ビット整数により表現されるデータです(－128～127)。同一音長のAD PCMデータの2倍のデータ長になります。

8ビットPCMデータ/16ビットPCMデータともに特別な設定もなしにZPCNVでZPDデータを作成できます。もちろんAD PCMデータと混在させてZPDデータにすることもできます。AD PCMデータは展開すると12ビットくらいの解像度を持っていますので，十分なクオリティで演奏するには16ビットPCMデータをすすめます。

再生周波数はどちらとも15.6kHz固定ですがAD PCM方式との混在が可能でAD PCMデータ側はこのときも5段階の周波数が使用可能です。

MEASURE7
プレイヤーZP.R

ここではZMUSIC.X支援ツールのサンプルとして作成したプレイヤー「ZP.R」について解説します。

7.1. ZP.Rについて

ZMUSIC.Xは，

A>copy ファイルネーム opm

とすることでコマンドラインから演奏データの演奏を行えます。また，

A>copy con opm

としたあと，ZMSコマンドを入力することによって演奏の停止や継続，再演奏などの制御を行うことができます。これらの演奏制御をより快適に，手軽に行えるように開発されたのがZP.Rです。

ZP.Rは演奏ツールとしての機能のほかに「ジュークボックス機能」や「トータルステップカウントの計算」「キーボードによるリアルタイム演奏制御機能(デバッグツール)」などの機能が備わっています。

7.2. コマンドスイッチについて

●演奏制御

−P 演奏データファイル名,セットアップファイル1,……,セットアップファイル4

演奏データをディスクから読み込んで演奏する。演奏データのファイル名の後ろに','で区切って最高4つまでのセットアップファイルを演奏前に読み込み，実行が可能。

セットアップファイルとは，たとえばAD PCMデータのコンフィギュレーションファイル，AD PCMのブロックデータ，MIDIダンプデータ(MDD)などのことを表す（セットアップファイルのMIDIダンプデータは拡張子'.MDD'のときのみ有効)。

セットアップファイルの拡張子は省略できないが演奏データのほうは拡張子を省略でき，その場合は'.ZMD'→'.ZMS'→'.OPM'の順で自動添付してファイルを検索する(見つからなかった場合はエラー)。

例

A>zp -p music.zms,drum.cnf,u220_seup.mdd
A>zp -p etude

スイッチ'−P'は省略することもでき，上の例は，

A>zp music.zms,drum.cnf,u220_seup.mdd
A>zp etude

とすることも可能。

さらに，演奏データを省略してZPDやMDD，AD PCMコンフィギュレーションファイルといったセットアップファイルのみの実行も可能である(拡張子の省略はできない)。ZPDとAD PCMコンフィギュレーションファイルは曲が演奏中であっても実行可能なので，演奏中にAD PCMのセットをリアルタイムに変更することができる。

例

A>zp list.cnf
AD PCMコンフィギュレーションファイルlist.cnfを実行する
A>zp drum_set.zpd
ZPDファイルdrum_set.zpdを登録する
A>zp M1preset.mdd
MDDファイルM1preset.mddを楽器へ転送する

−Pn1,n2,n3,…ni

演奏を開始します。コマンドスイッチの後ろにトラック番号(1≦n_i≦80)を書くとそのトラックのみ演奏を開始する。トラック番号は横につなげていくつでも書くことができる。また，トラック番号を書かずにコマンドスイッチのみを指定した場合は演奏可能なすべてのトラックの演奏を開始する。

例

A>zp -p　　(全トラック演奏開始)
A>zp -s1,2,10　(トラック1,2,10の演奏を開始する)
(MEASURE3 m_play(),MEASURE4 (P)コマンド参照)

−Sn1,n2,n3,…ni
−Cn1,n2,n3,…ni

それぞれ演奏の停止，再開を行う。コマンドスイッチの後ろにトラック番号(1≦n_i≦80)を書くとそのトラックに対して機能する。トラック番号は横につなげていくつでも書くことができる。

また，トラック番号を書かずにコマンドスイッチのみを指定した場合は演奏中の全トラックに対して機能する。

例

A>zp -s　　(全トラック演奏停止)
A>zp -c　　(全トラック演奏再開)
A>zp -s1,2,10　(トラック1,2,10の演奏を停止する)
A>zp -c1,2,10　(トラック1,2,10の演奏を再開する)
(MEASURE3 m_stop(),m_cont(),MEASURE4 (S)コマンド,(C)コマンド参照)

●チャンネルマスク

−En1,n2,n3,…,ni

指定チャンネルのみを演奏しそれ以外のチャンネルの演奏をマスクする。

チャンネル番号はPCM8モードでないときは1≦n_i≦25，PCM8独立チャンネルモードでは1≦n_i≦32となる。チャンネル番号をすべて省略した場合は，マスクをすべて解除(全チャンネル演奏可能に)する。実行はリアルタイムに行うことができ，演奏中の曲の任意のチャンネルをマスクしたり解除したりすることが可能。

例

A>zp -e1,2,3　(チャンネル1～3以外をマスクする)
A>zp -e　　(全チャンネルのマスクを解除する)

チャンネル表現にFM1～8，ADPCM1～8，MIDI1～16といった表記もできる。

例

A>zp -efm1，adpcm2，midi3
(MEASURE3 m_solo()，MEASURE4 (E)コマンド参照)

−Mn1,n2,n3,…,ni

指定チャンネルの演奏をマスクする。それ以外のチャンネルは通常演奏を行う。チャンネル番号はPCM8モードでないときは1≦ni≦25，PCM8独立チャンネルモードでは1≦n_i≦32となる。チャンネル番号をすべて省略した場合は，マスクをすべて解除(全チャンネル演奏可能に)する。実行はリアルタイムに行うことができ，演奏中の曲の任意のチャンネルをマスクしたり解除したりすることが可能。

例

A>zp -m1,2,3　(チャンネル1～3をマスクする)
A>zp -m　　(全チャンネルのマスクを解除する)

チャンネル表現にFM1～8，ADPCM1～8，MIDI1～16といった表記もできる。

例

A>zp -efm1，adpcm2，midi3
(MEASURE3 m_solo()，MEASURE4 (E)コマンド参照)
(MEASURE3 m_mute()参照)

●出力レベルの変更

−Fn

フェードアウト/フェードインを行う。−≦n≦-1のときフェードイン，1≦n≦85のときフェードアウトの指定となり，数値の絶対値が大きいほど音量の増減スピードは速くなる。n=0はフェードイン/フェードアウトの停止。

パラメータを省略した場合はデフォルト値のスピードでフェードアウトを行う。

例

A>zp -f20

フェードアウトをフェードアウトスピード20で行う。
A>zp -f-16
フェードインをフェードインスピード16で行う。
A>zp -f0
フェードインフェードアウトを中断する。
A>zp -f
デフォルトスピードでフェードアウトを行う。
(MEASURE3 m_fadeout(),MEASURE4 (F)コマンド参照)

−Olv,n1,n2,…,ni

指定チャンネルの出力割合を設定する。

lvは出力音量の割合で，設定範囲は0≦lv≦128。128が最大音量かつ基本状態。lvの値が小さいほど音量も小さくなる。lvによって決定づけられる実際の音量は各楽器や音源によって違うので注意(FM音源ではlv=80付近でほとんど無音状態になる)。

n_iは出力割合lvの設定を行うチャンネル番号。チャンネル番号はPCM8モードでないときは$1 \leqq n_i \leqq 25$，PCM8独立チャンネルモードでは$1 \leqq n_i \leqq 32$となる。lvのみを設定し，チャンネル番号をすべて省略した場合は，全チャンネルが出力レベルlvに設定される。

全パラメータを省略した場合は，全チャンネルの出力割合を基本状態(lv=128)へ戻す。

実行はリアルタイムに行うことができ，演奏中の曲の任意のチャンネルの出力割合を変更することが可能。

例

A>zp -o110,1,2,3
(チャンネル1，2，3の出力割合を110/128に設定する)
A>zp -o96
(全チャンネルの出力割合を96/128に設定する)
A>zp -o
(全チャンネルの出力割合を通常状態(128/128)に戻す)

●初期化

−I

Z-MUSICの演奏ワークやFM音源やMIDI楽器などを初期化する。曲が演奏中であれば演奏を停止してから処理を行う。

例

A>zp -i
(MEASURE3 m_init(),MEASURE4 (I)コマンド参照)

●トータルステップタイムの計算

−Q
−Q演奏データファイル名

'−Q'で，演奏中やその時点でZ-MUSICのトラックバッファに格納されている演奏データのステップタイムの合計値を計算し表示する。ファイル名を後ろにつければそれを読み込みバッファに格納してから合計値を計算する。

[do]〜[loop]以外で(たとえば[coda]〜[tocoda]によって)無限ループを構成している曲に対してこれを実行すると無限に計算を繰り返してしまう(インタラプトスイッチを押すしかない)。

なお，

トラック番号:ループ外の総カウント　ループ内の総カウント

というフォーマットで出力される (16進数)。

例

A>zp -q
A>zp -q music.zmd
(MEASURE3 m_total()参照,MEASURE4 (Q)コマンド参照)

●MIDIダンプデータ (MDD) 制御

−A

MIDI楽器からのデータを受信するためにZ-MUSICを待機状態にする。
(MEASURE3 m_rec(),MEASURE4(R)コマンド,MEASURE9参照。

−Aファイルネーム

'−A'として取り込んだ楽器のデータをファイルにセーブする。拡張子省略時は'.MDD'が自動的に添付される。Z-MUSICではこのファイルをMIDIダンプデータ(MDD)と呼ぶ。

(MEASURE3 m_save(),MEASURE9参照)

−Xファイルネーム

'−A'スイッチで作成したMIDIダンプデータを楽器へ転送する。拡張子を省略すると'.MDD'が自動添付される。

例

A>zp -xu220.mdd
(MEASURE3 m_trans(), MEASURE4, .MIDI_DUMP命令, MEASURE9参照)

●ジュークボックス機能

−Jインデックスファイル,セットアップファイル1,…,セットアップファイル4

'-J'を設定してその後ろにインデックスファイル(後述)の名前を書くとこれを読み込みこのファイルに書かれた演奏データを読み込み，これらを次々に演奏する(ジュークボックス機能)。インデックスファイルの拡張子を省略した場合は，拡張子'.JUK'が自動添付される。

演奏開始指定のスイッチ'−P'と同様に4つまでセットアップファイルを指定すれば，ジュークボックス開始前に，これらをまず実行することができる。

例

A>zp -jINDEX,MT32SOUND.MDD

また，ジュークボックス機能実行中には以下のキー操作が可能となる。

[SHIFT]+[OPT.1] 次の曲を演奏
[SHIFT]+[OPT.2] フェードアウト&次の曲を演奏
[SHIFT]+[XF4]　再演奏
[SHIFT]+[XF5]　フェードアウト&再演奏
[SHIFT]+[CTRL] 一時停止/一時停止解除
[CTRL]+[OPT.1] ひとつ前の曲を演奏
[CTRL]+[OPT.2] フェードアウト&ひとつ前の曲を演奏

●インデックスファイル

インデックスファイルは，

繰り返し回数(1〜255)，演奏データファイル名

の書式をとる。ZP.Rのジュークボックス機能では最大64曲までの演奏データの登録ができる。

演奏データはZMDのみ有効。演奏データファイル名の拡張子は省略可能で，省略時は'.ZMD'が自動添付される。

繰り返し回数は省略可能で省略時は繰り返し回数=1が自動設定される。

インデックスファイルに書かれた最後の曲の演奏を完了すると最初の曲へ戻る。

例

2,ETUDE
3,PRELUDE
FUGE

('ETUDE.ZMD'を2ループ演奏し，次に'PRELUDE.ZMD'を3ループ演奏し，最後に'FUGE.ZMD'を1ループ演奏して，これが終了すると初めの'ETUDE.ZMD'へ戻る)

なお，ジュークボックス演奏させるZMDには以下の条件がある。

1) 無限ループは[do]〜[loop]命令を使用している場合のみ有効。
2) ジュークボックスで演奏させる曲データには，ZPD指定('.ADPCM BLOCK_DATA')以外のAD PCMデータ登録命令が存在してはならない(たとえば'.ADPCM_LIST'命令や'.Onk ファイルネーム,Pp,Vv,Mm,d,Cc,s,R,Ff,I'命令)。

●キー定義の変更

キーと機能の対応は上記のようにデフォルトでは定められている

が，これを環境変数'zp_jukectrl'にキービットマップ値を列記することによってユーザーの好みに従って変更可能である。

set zp_jukectrl=〈次の曲へ#1，次の曲へ#2〉
〈フェードアウト&次の曲へ#1,フェードアウト&次の曲へ#2〉
〈停止/再開#1，停止/再開#2〉
〈再生#1，再生#2〉
〈フェードアウト&再生#1,フェードアウト&再生#2〉
〈前の曲へ#1，前の曲へ#2〉
〈フェードアウト&前の曲へ#1,フェードアウト&前の曲へ#2〉

・キー例

[SHIFT]=$E0[CTRL]=$E1[OPT.1]=$E2[OPT.2]=$E3
[XF1]=$A5 [XF2]=$A6 [XF3]=$A7 [XF4]=$B0 [XF5]=$B1

設定例

```
set zp_keyctrl=$e2,$b1  $e3,$b1  $e3,$b1  $e3,$e0  $e2,
$e0  $e3,$a7  $e0,$a6
        ↑
```

「前の曲へ」は[OPT.1]+[XF3]を意味する

注意

インデックスファイルに指定されている曲やその曲が使用するデータはジュークボックス機能開始前に読み込んでしまうため，多くのメモリを消費する。しかし，ジュークボックス機能では，演奏データやその他の必要データを先読みしてしまうため，ZMUSIC.Xが確保した各バッファを一切使用しない。よってZMUSIC.X側の各バッファの大きさはいくらでも構わない。

A>zmusic -t0 -p0 -w0　(トラックバッファ，AD PCMバッファ，ワークエリア=0)

として常駐しても問題なくジュークボックス機能は実行できる。

・処理の都合上，後述のデバッグツールとの併用はできない（一度後述の'-R'スイッチにて終了する必要がある)。

●デバッグツール機能

−D

ZP.Rが常駐し以降，以下のキー操作が可能となる(デバッグツール)。

[SHIFT]+[OPT1]　一時停止
[SHIFT]+[OPT2]　一時停止解除
[SHIFT]+[XF5]　早送り(押している間有効)
[SHIFT]+[XF4]　REPLAY
[SHIFT]+[XF3]　低速演奏(押している間有効)
[SHIFT]+[XF1]　フェードアウト
[SHIFT]+[XF2]　フェードイン

例

A>zp -d

●キー定義の変更

キーと機能の対応は上記のようにデフォルトでは定められているが，これを環境変数'zp_keyctrl'にキービットマップ値を列記することによってユーザーの好みに変更可能である。

set zp_keyctrl=〈再生#1，再生#2〉〈停止#1，停止#2〉〈再開#1，再開#2〉
〈早送り#1，早送り#2〉〈低速#1，低速#2〉
〈フェードアウト#1，フェードアウト#2〉
〈フェードイン#1，フェードイン#2〉

・キー例

[SHIFT]=$E0[CTRL]=$E1[OPT.1]=$E2[OPT.2]=$E3
[XF1]=$A5 [XF2]=$A6 [XF3]=$A7 [XF4]=$B0 [XF5]=$B1

設定例

```
set zp_keyctrl=$e2,$b1  $e3,$b1  $e3,$b1  $e3,$e0  $e2,
$e0  $e3,$a7
        ↑
```

「フェードイン」は[OPT.1]+[XF3]を意味する

●注意

処理の都合上，ジュークボックスとの併用はできない（一度後述の'-R'スイッチにて終了する必要がある)。

●常駐解除

−R

'-J','-D'はZP.Rがシステムに常駐するので，機能を停止するためには常駐解除を行わなければならない。'-R'を設定するとZP.Rがシステムに常駐していた場合常駐解除の処理を行う。ただしZP.Rが常駐していないのにこのスイッチを設定するとエラーとなる。

●同期演奏機能

−W 演奏データファイル名,セットアップファイル1,…,セットアップファイル4

−Pと同じく演奏開始の機能で設定パラメーターPと同様。セットアップファイルの実行後，演奏は開始せず，MIDIデータの$FA(スタートメッセージ)を受信するまで待機状態になる。$FAを受信すると即座に演奏を開始する。

MEASURE8
AD PCM加工ツールZVT.X

ここでは，自分でサンプリングデータを作成したり加工したりするツール「ZVT.X」の使用法について解説します。

8.1. ZVT.Xについて

ZVT.XはX68000のサンプリングデータ(AD PCMデータ)を扱うためのツールです。動作は視覚的に操作が行える「ビジュアルモード」と外部コマンドとして機能する「コマンドモード」とに分かれています。ここでは各モード別々に説明をします。

8.2. ビジュアルモード

●基本操作

基本的に画面下に表示されているメニューにカーソルを合わせリターンキーもしくはスペースキーを押すことでそれぞれの機能が実行されます。メニューには，選択されるとすぐに機能するものと若干のパラメータなどを聞いてくるものに分かれます。このパラメータなどの入力時に[ESC]キーを押すことやリターンキーのみを入力することによってそのモードから抜けることができます。

なお，メニューに表されている機能以外にも機能があり，これらはキーボードによるキー入力により実行されます（もちろんメニューに表示されているコマンドもキー入力によって実行可)。メニューに表示されている機能は実は，ほんの一部にすぎないため，「ZVT.X」を使いこなすためにはキーコマンドを覚える必要があります。

●カーソル操作

コマンド選択カーソルの移動にはカーソルキーを使います。

AD PCMデータを読み込んだり，サンプリングを実行すると，「START POINT」と「END POINT」の行にデータ列，さらに枠内に波形と青と赤の縦線が表示されます。「START POINT」,「END POINT」に表示されている数列は，それぞれ枠中の青と赤の縦線が指し示している付近のAD PCMデータやPCMデータです。この青と赤の縦線を波形カーソルと呼び，波形に対してなんらかの加工処理を行う場合は，この2つの波形カーソルに挟まれた範囲に対して行われます。波形カーソルは，コマンド選択カーソルを「START POINT」や「END POINT」に合わせ，テンキーを操作することによって移動させることができます(図1)。

```
   7  8  9
   4  5  6
   1  2  3
   0  ,  .
[7] 左へ32ステップ移動          [9] 右へ32ステップ高速移動
[4] 左へ1ステップ移動            [6] 右へ1ステップ移動
[1] 左へ波長の1/16ステップ移動   [3] 右へ波長の1/16ステップ移動
        [0] 波形の先頭へ瞬間移動
        [,] 波形の真ん中へ瞬間移動
        [.] 波形の最後尾へ瞬間移動
```

図1 波形カーソルのキー操作

●キーコマンド一覧(アルファベット順)

() 内の文字はコマンドメニュー名に対応する。

A
B

アクセス対象バンクをセレクトする。

ZVT.Xには加工したりサンプリングしたりするデータエリアが2つあり，コマンド機能の対象となるデータをそれぞれバンクA，バンクBと呼んでいる。本コマンドでこの操作対象バンクを選択することができる。

C

スタートポイントとエンドポイント間の波形を切り出す。

D (DISK)

ディスクモードへ入る。

1. LOAD

波形データのロードを行う。ファイルネームを聞いてくるのでパスや拡張子を含めて91文字以内で入力する。ファイル名を省略したり，ワイルドカードを使用するとファイル選択モード(囲み参照)に入る。

2. SAVE

波形データのセーブを行う。ファイルネームを聞いてくるのでパスや拡張子を含めて91文字以内で入力する。ファイル名を省略したり，ワイルドカードを使用するとファイル選択モード(囲み参照)に入る。

3. SAVE BY 4PHASES

原波形を含む4段階の波形データを自動生成しセーブする（ただし自動生成されるデータはAD PCMデータ形式固定)。

このモードを選択すると次に，

1. VOLUME 2. PITCH

のメニューが現れる。1を選べば4段階のボリュームで，2を選べば4段階の音程でセーブされることになる。

これを選択後さらに，

1. MINUS 2. PLUS

のメニューが現れる。これはどの方向に4段階のデータを生成するかを決めるものである。

ボリュームを選んだ場合にマイナス方向を選択すると原波形のボリュームの，

100%,88%,76%,60%

のデータが自動生成されセーブされる。プラス方向では原波形のボリュームの，

100%,114%,126%,140%

のデータがセーブされる。音程を選んだときには各方向に半音単位の変化量でセーブされる。つまり原波形が今オクターブ4のCだったとすると，

マイナス方向では　O4のC,O3のB,O3のA#,O3のA
プラス方向では　O4のC,O4のC#,O4のD,O4のD#

がセーブされることになる。

最後にファイルネームを聞いてくるが拡張子は無視され自動生成される。ファイルの拡張子は強制的に'.4' '.3' '.2' '.1'の4つになる（原波形が'.4')。ファイルネームを省略したり，ワイルドカートを使用するとファイルセレクトモードへと移行する。

4. TEST PLAY

ファイルを再生する。ファイルネームを聞いてくるのでパスや拡張子を含めて91文字以内で入力する。ファイル名を省略したり，ワイルドカードを使用するとファイル選択モードに入る。

ファイル選択モード

ディスク関連のコマンドでファイルネームを省略したり，ワイルドカードを設定するとファイル選択モードになります。ここではカーソルキーによるファイルの選択やディレクトリ間の移動がスムーズに行えるようになっています。

使用キーは以下のとおりです。

カーソルキー	カーソルの移動
RETURN/ENTER	ファイルやディレクトリの選択
ESC	ファイル選択モードを終了する（ファイル選択の拒否）
TAB	カーソルを左最上部へ瞬間移動。またディレクトリの移動もこれで可能。スピーディに親ディレクトリに移動可能
?	ファイルネームの入力。ワイルドカードを指定するとそれに対応したファイルのみが画面に現れる
SPACE	再生(PCM/AD PCMモードの影響を受ける)
数字キー(0～4)	SPACEキーでのPCMデータ再生における再生周波数の切り換え

E

PCMデータおよびAD PCMデータにエフェクト(加工)をかける。

1.↑/↓ 2.←/→ 3.ADD 0 TO TAIL 4.REVERSE 5.CHORUS 6.DELAY 7.SFX 8.SOMOOTHING 9.ENVELOPE

の9種類のエフェクトよりひとつを選択する。

1. ↑/↓

PCMデータをグラフのY軸方向にシフトするもの(マイナス値で下へ)機能選択後,どのくらいシフトするか(オフセット値)を聞いてくる

2. ←/→

PCMデータをグラフのX軸方向にシフトするもの(マイナス値で左へ)機能選択後,どのくらいシフトするか(オフセット値)を聞いてくる

3. ADD 0 TO TAIL

データの最後尾にn個の0を加える。機能選択後,オフセット値を聞いてくる

4. REVERSE

逆転

5. CHORUS

音が透き通った感じになる

6. DELAY

エコーがかかったような音になる

7. SFX

金属音に変化する

8. SMOOTHING

こもったような音に変化する

9. ENVELOPE

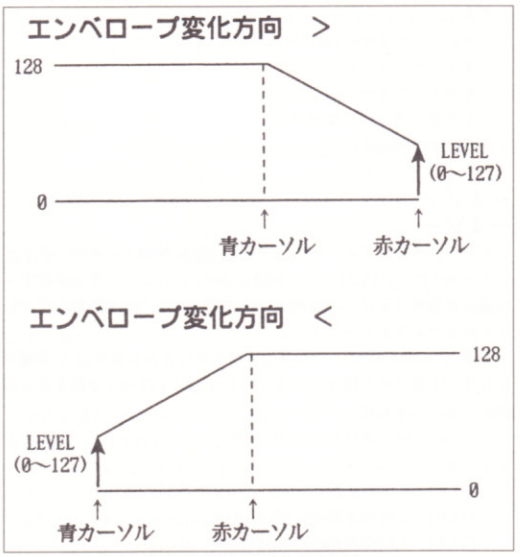
エンベロープを変形する(フェードイン/フェードアウト)機能選択後,イン/アウトレベル(0~127)とエンベロープの変更方向を聞いてくる

1. < … フェードイン

2. > … フェードアウト

設定したパラメータとその効用は図2参照

なお,機能は青赤の2つの波形カーソルに挟まれた区間に対して働くことに注意。



F (SAMPLING RATE)

サンプリング時や再生時の周波数を設定をする。サンプリング周波数は,一般に周波数が高ければ高いほど高音質となる。

I

波形の頭と後尾を見つける。単純なアルゴリズムによる探査のため複雑な波形に対しては希望の結果を得られないことがある。また,データモードによって動作のしかたが微妙に違うため,データモードを変えて実行すると違った結果を得ることができる。

J

音程の変更を1Hz単位で行う。コマンド入力後,変更前の周波数と変更後の周波数を聞いてくる。周波数パラメータ設定範囲は1~65535。変換前と変換後の周波数は厳密に入力する必要はなく,両周波数の整数比を入力するだけでもよい。つまり「1オクターブ上げる(周波数を2倍にする)」場合,

変更前周波数=1Hz　　変更後周波数=2Hz

でも,

変更前周波数=100Hz　　変更後周波数=200Hz

でも得られる結果は同じである。

K (PITCH)

半音単位でピッチをシフトする。パラメータは−12から+12。±1オクターブの音程変換が可能。

M (MONITOR)

モニタ機能のON/OFF。「AUDIO IN」端子からの入力をX68000のA/D,D/Aコンバータを通してヘッドホンなどで聞くことができる。

N

ポルタメント(音程を滑らかに変化させる)を1Hz単位で行う。コマンド入力後,その音の基本周波数と変化希望周波数を聞いてくる。周波数パラメータ設定範囲は1~65535。2つの周波数パラメータは厳密に入力する必要はなく,両周波数の整数比を入力するだけでもよい。つまり,「1オクターブ上へ滑らかに変化させる(周波数を滑らかに2倍にする)」場合,

変更前周波数=1Hz　　変更後周波数=2Hz

でも,

変更前周波数=100Hz　　変更後周波数=200Hz

でも得られる結果は同じである。

O

再生やモニタ機能のパンポットを設定する。

P (PLAY)

バンクセレクトされているバンクのデータを再生する。

Q (EXIT)

[ESC]+[Q]でZVT.Xを終了する。誤操作を避けるためこのコマンドのみ2キーコマンドになっている。

R (RECORD)

録音を行う。オートサンプリング機能のスイッチ(TRIGGER)がON,さらにモニタ機能(MONITOR)ONのときにはオートサンプリングを行うがこれらのスイッチが片方でもOFFのときには通常のサンプリングを行う。

S (DATA SIZE)

サンプリングする音の時間をデータサイズによって設定する。

T (SAMPLING TIME)

サンプリングする音のデータサイズを「秒」で設定する。

V (VOLUME)

音量を変化させる。パラメータはパーセント単位。1~300%の範囲のみ有効。

X

アクセスバンクの切り換え。[X]キーを押すたびにセレクトバンクが,A/Bと交互に切り換わる。

Z

2つのデータ間の畳み込み演算を行う。2つの音声データの特性をひとつにまとめたりすることができる。演算結果をどちらのバン

クに格納するかを聞いてくる。
1. A＊B→B … 演算結果をBへ格納する
2. A＊B→A … 演算結果をAへ格納する
なお，演算にはかなりの時間を必要とする。

@　(TRIGGER)
オートサンプリング機能の有効/無効を設定する。

＊
PCMのデータ表示を10進数または16進数へ交互に切り換える。デフォルトは16進数モード。

＋　(COMPOUND)
バンクAとBのデータをミキシングしてひとつのデータにする。パラメータとしてミックス方向を聞いてくる。
1. A→B … AとBをミックスしてミックスされたデータをBへ格納する
2. A←B … AとBをミックスしてミックスされたデータをAへ格納する

－
バンクAとBのデータ間の算術/論理演算を行う。演算子を，
1. ADD 和
2. SUB 差
3. OR 論理和
4. AND 論理積
5. EOR 排他的論理和

の5つから選択する。演算子タイプ選択後，演算結果をどちらのバンクに格納するのかをさらに聞いてくる。
1. A○B→B … 演算結果をBへ格納する
2. A○B→A … 演算結果をAへ格納する

／
ソースバンクのデータをデスティネーションバンクのデータのスタートポイントへはめ込む。パラメータとしてはめ込む方向を聞いてくる。
1. A→B … AをBのスタートポイントへはめ込む
2. A←B … BをAのスタートポイントへはめ込む

＝
バンク間のデータのコピー。パラメータとしてコピー方向を聞いてくる。
1. A→B … AをBへコピーする
2. A←B … BをAへコピーする

？
各バンクの情報を表示する。
ADPCM DATA ADDRESS:
　AD PCMデータが格納されているアドレス
START POINT:青カーソルのオフセット位置
END POINT:赤カーソルのオフセット位置
ADPCM DATA SIZE:AD PCMデータの大きさ
EFFECTIVE DATA SIZE:
　青カーソルと赤カーソルに挟まれた区間の大きさ
PCM DATA ADDRESS:PCMデータが格納されているアドレス
FREE AREA:フリーエリア

CLR
セレクトされているバンクのデータをクリアする。

TAB
データモード(PCM/AD PCM)の切り換えを行う。

8.3. コマンドモード
ZVT.Xの機能はビジュアルモードからだけでなくHuman68kのコマンドラインからも実行することができます。

A>zvt 〈コマンドスイッチ列1〉〈ファイルネーム1〉〈ファイルネーム2〉

A>zvt -m[n] [コマンドスイッチ列1]〈ファイルネーム1〉 [コマンドスイッチ列2]〈ファイルネーム2〉〈ファイルネーム3〉

という書式でコマンド指定を行います。

ファイルネーム1で指定されたAD PCMデータファイルを読み込み，これをコマンドスイッチで指定された機能で加工処理を行い，これをファイルネーム2としてセーブする，というのがコマンドモードの基本動作となります。

ミキシング(合成)指定のときのみファイルネーム3の指定を必要とし，コマンドスイッチ列2が有効となります。よってこのときはファイルネーム1で指定されたAD PCMデータファイルを読み込み，これをコマンドスイッチ列1で加工し，ファイルネーム2で指定されたAD PCMデータファイルを読み込み，これをコマンドスイッチ列2で加工し，この2つを合成して，ファイルネーム3としてセーブする，ということになります。

●コマンドオプションスイッチの解説

－4P+
－4P－
ビジュアルモードの4段階データセーブに相当するもの。ファイルネーム1のAD PCMデータを読み込み，これに対して半音単位で4段階のピッチシフトを行い拡張子'.4'，'.3'，'.2'，'.1'を添付したファイルネーム2でセーブする。

'－4P'の後ろの＋,－の符号はピッチシフトの方向を表す。＋が音程を4段階に上昇させる指定で，逆に－が音程を4段階に下降させる指定となる。たとえば，FILE1.PCMがオクターブ4のCだったときに，

A>zvt -4p+ FILE1.PCM ABC

とすると，

オクターブ4のC(原波形)が'ABC.4'
オクターブ4のC＃が'ABC.3'
オクターブ4のDが'ABC.2'
オクターブ4のD＃が'ABC.1'

としてセーブされる。

－4V+
－4V－
ビジュアルモードの4段階データセーブに相当するもの。ファイルネーム1のAD PCMデータを読み込み，これに対して%単位で4段階の音量変更を行い，拡張子'.4'，'.3'，'.2'，'.1'を添付したフ」ァイルネーム2でセーブする。

'－4V'の後ろの＋,－の符号は音量変更の方向を表す。＋が音量を4段階に上昇させる指定で，逆に－が音量を4段階に下降させる指定となる。たとえば，

A>zvt -4v+ FILE1.PCM ABC

とすると，

FILE1.PCMの音量の100%(原波形)が'ABC.4'
FILE1.PCMの音量の114%が'ABC.3'
FILE1.PCMの音量の126%が'ABC.2'
FILE1.PCMの音量の140%が'ABC.1'

としてセーブされる。

また，

A>zvt -4v- FILE1.PCM ABC

とすると，

FILE1.PCMの音量の100%(原波形)が'ABC.4'
FILE1.PCMの音量の88%が'ABC.3'
FILE1.PCMの音量の76%が'ABC.2'
FILE1.PCMの音量の60%が'ABC.1'

としてセーブされる。

−A

ファイルネーム1のPCMデータをファイルネーム2のAD PCMデータへ変換する。

−C

−Aの逆の動作。PCMデータは16ビット符号付きで格納される。

−G

基本的にビジュアルモードはG-RAMが'RAMディスクなどで使用されていると起動不可能である。しかしこのスイッチを指定するとG-RAMディスクなどを破壊し強制的にビジュアルモードへ移行させることができる。

−I n

ファイルネーム1のAD PCMデータとファイルネーム2のAD PCMデータへはめ込みファイルネーム3でセーブする。−Iの後ろにオフセット値を書けるがこれはAD PCMのバイト数となる（通常は0または省略で構わない）。

−L filename1, filename2

ビジュアルモード起動時にfilename1をバンクAに，filename2をバンクBに読み込んで起動する。起動時に読み込める音声ファイルはAD PCMに限定される。

−Mn

ファイルネーム1のAD PCMデータとファイルネーム2のAD PCMデータを加工指定があればそれぞれ加工処理し，これらをミックス（合成）しファイルネーム3でセーブする。−Mの後ろにオフセット値を書けるがこれはAD PCMのバイト数となる（通常は0または省略で構わない）。

−Pn

ファイルネーム1のAD PCMデータの音程をn半音分シフトしファイルネーム2でセーブする。パラメータは−12から+12まで設定可能。

−Vn

ファイルネーム1のAD PCMデータの音量を%単位で変更しファイルネーム2でセーブする。パラメータはパーセンテージで表し，1〜300%まで設定可能。

●コマンドスイッチの複数設定について

コマンドスイッチの'P','V'は同時に設定可能です。また，コマンドスイッチ'P','V'はコマンドスイッチの'4','A','C','M'と同時に設定可能です。

以下に使用具体例をいくつか示します。

A>zvt -p3 -v150 ABC.PCM DEF.PCM

とすると'ABC.PCM'のAD PCMデータを+3半音分シフト，音量を150%にしたものを'DEF.PCM'でセーブします。

A>zvt -a -p+5 ABC.PCM DEF.PCM

とすると'ABC.PCM'のPCMデータを5半音分上へシフトしたうえでAD PCMデータへ変換し，これを'DEF.PCM'でセーブします。

A>zvt -c -p-5 ABC.PCM DEF.PCM -v180

のようにファイルネームの後ろにスイッチを設定可能です。この例では'ABC.PCM'のAD PCMデータを−5半音分シフトし，さらに音量を原波形の180%に変更したPCMデータを'DEF.PCM'でセーブします。

A>zvt -p-3 -v50 ABC.PCM -p5 -v120 DEF.PCM GHI.PCM -m

とすると'ABC.PCM'のAD PCMデータを−3半音分シフト，音量を50%にしたものと'DEF.PCM'のAD PCMデータを+5半音分シフト，音量を120%にしたものとをミックスし'GHI.PCM'でセーブします。

A>zvt -4v+ -v80 -p-7 ABC.PCM DEF

では'ABC.PCM'のAD PCMデータを−7半音分シフトし音量を80%×100%=80%に変更したものを'DEF.4'でセーブし，そして80%×114%=91.2%に変更したものを'DEF.3'でセーブ，80%×126%=100.8%に変更したものを'DEF.2'でセーブ，80%×140%=112%に変更したものを'DEF.1'でセーブします。

●ワイルドカードについて

ファイルネームにワイルドカードが使用可能です。たとえば，

A>zvt -v150 -p5 *.PCM *1.PCM

A>zvt -m *.PCM A*.PCM *B.PCM

といった使い方ができます。ただし，使えない場合もあるので注意してください。

●ファイルネーム'PCM'について

ZMUSIC.XまたはPCMDRV.SYSが組み込まれているとき，出力側のファイルネームに'PCM'を設定した場合は，生成されたデータをファイルとして生成せずに，直接再生して聞くことができます。

たとえば，

A>zvt -p-3 -v50 ABC.PCM -p5 -v120 DEF.PCM PCM -m

とすると'ABC.PCM'のAD PCMデータを−3半音分シフト，音量を50%にしたものと'DEF.PCM'のAD PCMデータを+5半音分シフト，音量を120%にしたものとをミックスし，これをセーブせず音声として聞くことができます。

ファイルの拡張子とデータ形式

拡張子'.PCM'のファイルはZVT.XではAD PCMファイルと自動的にみなしてロード/セーブします。また拡張子'.P16'のファイルは16ビットPCMファイルと見なしてロード/セーブします。それ以外の拡張子を持ったファイルをアクセスする場合は，その時点でのデータモードがAD PCMならAD PCMとして，データモードが16ビットPCMデータなら16ビットPCMデータとしてアクセスします。なるべく'.PCM'，'.P16'のどちらかを用いるようにしましょう。

オフセット値について

「←/→」と「ADD 0 TO TAIL」では，機能メニューを選択後，オフセット値を聞いてきますがこれはデータモードによって効用が異なります。

たとえばPCMモードのとき1000を入力するとオフセットはPCMデータ1000個分（つまり2000バイト）ということになります。またAD PCMモードで1000と入力した場合にはAD PCMデータ1000個分（つまり1000バイト）ということになります。

オフセット値の例

・データモードがPCMのとき

3.9kHz 1秒=3900
5.2kHz 1秒=5200
7.8kHz 1秒=7800
10.4kHz 1秒=10400
15.6kHz 1秒=15600

・データモードがAD PCMのとき

3.9kHz 1秒=1950
5.2kHz 1秒=2600
7.8kHz 1秒=3900
10.4kHz 1秒=5200
15.6kHz 1秒=7800

「↑/↓」のオフセット値はPCMデータへ直接加算される数値を意味しているのでデータモードにはまったく無関係です。

MEASURE9
MIDIデータを扱う

ここではZMUSIC.Xの機能のひとつであるMIDI楽器の設定をファイルとしてセーブする機能について解説します。また，楽器のメモリを操作するエクスクルーシブメッセージについて具体的な用例を挙げて解説します。

9.1. MIDI楽器の内部データをファイルにセーブする

ZMUSIC.Xには常駐後に使用可能となるシステムファイル名'MIDI'を用いて楽器の状態（音色や設定など）をファイルに落とす機能があります。これによって楽器の設定状態や音色のデータをX68000で管理することができるようになります。ここではこの機能の具体的な使用方法について述べます。

●MIDIダンプデータとはなにか

楽器内部のメモリ内容などをファイルへ落としたものをZ-MUSICシステムではMIDIダンプデータと呼びます。さて，このデータはZ-MUSICシステムでは16進数のテキスト形式のデータで扱われます。ですから，作成されたファイルは，

A>ed ファイルネーム

でテキストエディタからエディットが可能です（ただしMIDIダンプデータの先頭につく改行コードは，実は重要な意味を果たしていますので消したりしてはいけません）。テキストファイルにした理由はデータ中に$1Aを発見するとCOOKEDモードのデバイスドライバはファイルの最後と判断して動作を終了してしまうためです。

このほうが簡単にエディタから読み込めて書き換えができますし，問題はないでしょう。もちろん機械語レベルの操作のためにZ-MUSICファンクションコールでは文字型でない生データのやり取りもサポートしています（MEASURE10参照）。なにかアプリケーションを作成する場合はそちらを使うべきです。

●MIDIデータをファイルに落とすには

まず，楽器側にMIDI OUT端子がなくてはなりません。さらに楽器側に自分のメモリ内のデータを外部へ送信する機能が必要です。MIDI OUT端子があるような楽器ならばたいていは備わっている機能ですので楽器のマニュアルを参照してみましょう。おそらく「MIDIデータダンプ」，「バルクダンプ」といった項目のところに載っているはずです。

次にX68000のMIDI IN端子と楽器のMIDI OUT端子をMIDIケーブルで接続します。

以上のような準備をしているという前提で話を進めていきます。MIDIデータの受信にはさまざまな方法があります。以下にそれらの手順を列挙していきます。いずれかの方法でデータを保存してください。

●コマンドラインから行う場合（その1）

0) ドライバの常駐を確認する
1) 楽器側のバルクダンプのメニューを出し確認する
2) A>copy con opmを実行する
3) (R)と入力しリターン
4) 楽器のバルクダンプを実行する
5) [SHIFT]+[BREAK]でコマンドに戻る
6) 楽器のコンソールに"COMPLETED"などのメッセージを確認するまでひたすら待つ
7) A>copy midi filenameを実行する

●コマンドラインから行う場合（その2）

0) ドライバの常駐を確認する
1) 楽器側のバルクダンプのメニューを出し確認する
2) A>zp-aを実行
3) 楽器のバルクダンプを実行する
4) 楽器のコンソールに"COMPLETED"などのメッセージを確認するまでひたすら待つ
5) A>zp-a filename を実行する

●BASICから行う場合

0) ドライバの常駐を確認しmusicz.fncを組み込んだBASICを立ち上げる
1) 楽器側のバルクダンプのメニューを出し確認する
2) m_rec()を実行する
3) 楽器のバルクダンプを実行する
4) 楽器のコンソールに"COMPLETED"などのメッセージを確認するまでひたすら待つ
5) m_save("filename")を実行しセーブする

●注意点

データの取り込み用バッファとしてトラックバッファ，文字型データへの変換の際にワークエリアを使用します。データ取り込み中ビープ音が鳴った場合はトラックバッファやワークが不足していることを表します。データの取り込みを行うときはトラックバッファ，ワークエリアともに十分に確保してZMUSIC.Xを常駐させるべきです。具体的には，

A>zmusic -t200 -w200

とします（この例ではトラックバッファ，ワークエリアともに200Kバイト確保して常駐している）。また，ビープ音が鳴った場合にはデータをセーブしてもそのデータには信頼性がありません。

9.2. MIDIダンプデータの楽器への転送

以上のようにして作成されたファイルを，逆に楽器側へ転送する方法について解説します。これもいくつかの方法がありますので，用途などにあわせて適当なものを選択してください。具体的には以下のような手順に従って操作することになります。

●コマンドラインから行う場合（その1）

0) ドライバの常駐を確認する
1) A>copy filename midiを実行する
2) 待つ

●コマンドラインから行う場合（その2）

0) ドライバの常駐を確認する
1) A>zp-x filenameを実行する
2) 待つ

●BASICから行う場合

0) ドライバの常駐を確認しmusicz.fncを組み込んだBASICを立ち上げる
1) m_trans("ファイルネーム")を実行する
2) 待つ

●曲中に設定するとき

BASICプログラム … m_trans("ファイルネーム")命令を用いる
ソースファイル（ZMSファイル）… .midi_dump=ファイルネーム命令を用いる

エクスクルーシブ送信後のウェイト

ZMUSIC.XではMIDIダンプデータの送信時や，共通コマンドの実行時，EOX（END OF EXCLUSIVE MESSAGE）$F7を送信した場合，その後，ウェイトを入れるようにプログラムされています。これは楽器側がエクスクルーシブを処理するのに時間を必要とするためです。このウェイトの値はZMUSIC.X常駐時のオプションスイッチ'-X'で設定することができます。

MMLの@XやXコマンドを用いてエクスクルーシブを送信した場合は（テンポのずれを避けるために）EOXの送信後にウェイトを入れていません。この場合，ユーザーが休符やウェイトコマンドを挿入するなどして適度なウェイトを与えてください。

9.3. ローランドエクスクルーシブについて

すでに多くのユーザーがローランドの楽器を使用していると思われます。ここではそのローランド製のMIDI楽器のエクスクルーシブについて特に解説を行います。

ZMUSIC.Xでは'roland_exclusive'や'm_roland()'をはじめ，MMLレベルでも'X'コマンドなどが用意されており，かなり容易に楽器のメモリを書き換えることが可能です。ここでは'roland_exclusive'命令を使ったCM-32P（あるいはCM-64のPCM音源パート）のパーシャルリザーブを例として解説を行います。

＊6－3 System area

パーシャル・リザーブは6パート分を一度に送らなければ無効です。
また、6パートのパーシャル・リザーブ合計が31以下でなければなりません。

Offset address	Description	
00 00	0aaa aaaa	MASTER TUNE 0 - 127 (432.1Hz - 457.6Hz)
00 01	0000 00aa	REVERB MODE 0 - 3 (ROOM, HALL, Plate, Tap delay)
00 02	0000 0aaa	REVERB TIME 0 - 7 (1 - 8)
00 03	0000 0aaa	REVERB LEVEL 0 - 7
00 04	00aa aaaa	PARTIAL RESERVE (PART 1) 0 - 31
00 05	00aa aaaa	PARTIAL RESERVE (PART 2) 0 - 31
00 06	00aa aaaa	PARTIAL RESERVE (PART 3) 0 - 31
00 07	00aa aaaa	PARTIAL RESERVE (PART 4) 0 - 31
00 08	00aa aaaa	PARTIAL RESERVE (PART 5) 0 - 31
00 09	00aa aaaa	PARTIAL RESERVE (PART 6) 0 - 31
00 0A	000a aaaa	MIDI CHANNEL (PART 1) 0 - 16 (1 - 16 ,OFF)
00 0B	000a aaaa	MIDI CHANNEL (PART 2) 0 - 16 (1 - 16 ,OFF)
00 0C	000a aaaa	MIDI CHANNEL (PART 3) 0 - 16 (1 - 16 ,OFF)
00 0D	000a aaaa	MIDI CHANNEL (PART 4) 0 - 16 (1 - 16 ,OFF)
00 0E	000a aaaa	MIDI CHANNEL (PART 5) 0 - 16 (1 - 16 ,OFF)
00 0F	000a aaaa	MIDI CHANNEL (PART 6) 0 - 16 (1 - 16 ,OFF)
00 10	0aaa aaaa	MASTER VOLUME 0 - 100
Total size	00 00 11	

Address Map

Address	Block	Sub Block	Reference
50 00 00	Patch Temp.	Part 1	6-1
		Part 2	
		Part 5	
		Part 8	
51 00 00	Patch Memory	# 1	6-2
		# 2	
		#127	
		#128	
52 00 00	System Area		6-3
7F xx xx	All Parameters Reset		6-4

9.3.1. マニュアルの後ろを見てみる

マニュアルの後ろを見てみましょう。CM-64ならば33ページを開いてください。お持ちでない方は表を参照してください。以下はCM-64を例にとって話を進めていきます。まず'＊6-3'という表に注目します。この表の中段に，

00 04	00aa aaa	PARTIAL RESERVE (PART 1) 0-31
00 05	00aa aaa	PARTIAL RESERVE (PART 2) 0-31
00 06	00aa aaa	PARTIAL RESERVE (PART 3) 0-31
:
| 00 09 | 00aa aaa | PARTIAL RESERVE (PART 6) 0-31 |

というものが見つかるはずです。これを確認したら次にこの表'＊6-3'のタイトルを見てください。「SYSTEM AREA」とありますね。ここで，この表の右にあるアドレスマップからこの'SYSTEM AREA'という文字を探します。すると，

52 00 00

が'SYSTEM AREA'のベースアドレスということがわかります。つまり，表'＊6-3'の中のパラメータを設定したい場合は，表の左に書いてあるオフセットアドレスと，このベースアドレスを足したアドレスに値を送ればよいことになります。この例でいくと，

52 00 00 ＋ 00 04 ＝ 52 00 04

がパーシャルリザーブのパラメータのアドレスということになります(表'＊6-3'のタイトルの下を見ると「パーシャルリザーブは6パート分を一度に送らなければ無効です」とあるのでもちろんこれに従うこと)。

結局，以下のようにコマンドを書けばよいことになります。

.roland_exclusive dev,mdl {$52,$00,$04, 2,3,5,4,3,6}

この例ではパート1～6，それぞれ2,3,5,4,3,6声をリザーブしています。もちろん楽器の最大同時発声数を超えてはいけません。この例では2+3+5+4+3+6=23で，まだ31－23＝8声分の余裕がありますね。

さて，いまの例でdev,mdlというのが出てきましたが，これは操作対象の楽器のID番号のことを表しています。

mdlはモデルIDと呼ばれるものでメーカーがその楽器に与えた識別番号のことです。MT-32/CM-32L/CM-32P/CM-64はすべて$16と決められています (CM-64ならばマニュアルの28ページの上部に記載されている)。

devはデバイスIDと呼ばれるもので，同じ機種(あるいは同一のデバイスIDを持った)楽器でもこれを変えておけば個別に操作が可能となります。本来はユーザーが任意に設定可能な識別番号です。しかしCM-32L/CM-32P/CM-64は$10に固定となっています。結局，上の例は，

.roland_exclusive $10,$16 {$52,$00,$04 ,2,3,4,5,6}

ということになります。

以上の手順をまとめると，

1) 設定したいパラメータの表を探す
2) 表のタイトルをアドレスマップより探す
3) 設定したいパラメータのオフセット値と2)で見つけたベースアドレスを加算する
4) 2種類のIDを確認してコマンドを書く

となります。

9.3.2. その他の例

CM-32P(あるいはCM-64のPCMパート)の各パートの受信MIDIチャンネルを設定するときは，

roland_exclusive $10,$16 {$52,$00,$0a,0,1,2,3,4,16}

のようになります。この例ではパート1～5をそれぞれチャンネル1～5に，パート6はOFFにしています。MIDIチャンネルは通常1～16ということになっていますが内部処理的には0～15なので注意してください。そういうわけでパラメータの0はMIDIチャンネル1，パラメータの1はMIDIチャンネル2……以下同様ということになります。最後のパラメータの16はMIDIチャンネル17ということではなくそのパートをミュートする（使用しない）ということを意味しています。多くのローランド系の楽器は，

MIDIチャンネル	実際のパラメータ値
1～16	0～15
OFF	16

ということになっていますので覚えておいてください。

同様にしてリバーブの設定や各パートの細かい設定(リバーブのON/OFF，キートランスポーズの設定など)も行うことができます。もちろんBASICコマンドの'm_roland()'やMMLの'X'コマンドもいままで述べてきた同様な方法で行えます。

注意点としては通常１～128で使用していたものが内部処理的には０～127となっていたり，－64～＋63で使用していたものが０～127へシフトされていたりするのでパラメータの設定にはマニュアルを確認することが必要です。

複数のX68000による同期演奏

0) ZMUSIC.Xを/Eオプションを設定して常駐する
1) X68000-AのMIDI OUTからX68000-BのMIDI INへつなぐ
2) X68000-BのMIDI OUTを楽器１へつなぐ
3) X68000-BのMIDI THRUを楽器２へつなぐ
4) X68000-Bを，
A>ZP /W filename
として演奏待機状態にする
5) X68000-Aを，
A>ZP filename
で演奏開始する。

２台以上の場合はX68000-xのMIDI OUTをX68000-(x＋1)のMIDI INへつないでいくか，X68000-xのMIDI THRUをX68000-(x＋1)のMIDI INへつないでく２通りが考えられますが実用には４台までが限界と思われます。

MEASURE10
Z-MUSICのファンクションコール

ここではZMUSIC.Xが常駐したあとに使用できるファンクションコールについて説明します。使用にはMPU68000の機械語の知識が必要です。一般的な使用をする限りは読み飛ばして構いません。

10.1. ファンクションコールのコール法

●コール方法その1(OPMDRV.Xコンパチモード)

```
moveq.l #n,d1      *n=ファンクションナンバー
lea     ~,a1       *パラメータが必要であれば設定
moveq.l #m,d2      *パラメータが必要であれば設定
IOCS    _OPMDRV
```

●コール方法その2(高速応答モード)

```
moveq.l #n,d1      *n=ファンクションナンバー
lea     ~,a1       *パラメータが必要であれば設定
moveq.l #m,d2      *パラメータが必要であれば設定
trap    #3
```

戻り値はd0.lとa0.lのみで,その他のレジスタは全部保存されます。ファンクション内でエラーが発生すればd0にそのエラーコード(MEASURE12参照)が代入されています。d0.l=0は正常終了を表します(戻り値にd0を使用している場合を除く)。

ファンクション$00~$0FまではOPMDRV.Xとコンパチでほぼ同等の機能を果たします。「その1」はあくまで「OPMDRV.X」との互換性を保つために設けたコール方法で「その2」の方法のほうが応答が高速です。

ドライバの常駐チェックは具体的にはリスト1のようにします。原理はTRAP #3のベクタが指し示すアドレス-8から"ZmuSiC"があるかどうかを調べているだけです。さらに"ZmuSiC"の後ろにはバージョンコード2バイトが格納されています。

上位バイト　バージョン整数部, バージョン小数第1桁
下位バイト　対応バージョンコード, バージョン小数第2桁

対応バージョンコード

UNIVERSAL VERSION	0
16bit VERSION	1
RS-MIDI VERSION	2
POLYPHON VERSION	3

例

POLYPHON VERSION 1.52 → "ZmuSiC",$15(.b),$32(.b)

サンプルリスト1

```
chk_drv:
        *>eq=常駐確認
        *>mi=常駐していない
        move.l  $8c.w,a0
        subq.w  #8,a0
        cmpi.l  #'ZmuS',(a0)+
        bne     chk_drv_err
        cmpi.w  #'iC',(a0)+
        bne     chk_drv_err
        rts
chk_drv_err:
        moveq.l#-1,d0
        rts
```

ファンクション$00　　m_init

機能: 音源, ドライバの初期化
引数: なし
戻り値: なし

ファンクション$01　　m_alloc

機能: トラックバッファの確保
引数: d2の上位16ビット=トラックナンバー(1~80)
d2の下位16ビット=バッファサイズ($00A0~$ffff)
戻り値: なし
d0.l=エラーコード

ファンクション$02　　m_assign

機能: 確保したトラックバッファをどの音源に割り当てるかを設定する
引数: d2の上位16ビット=チャンネル番号(1~25, ただしPCM8独立チャンネルモード時は1~32となる)
d2の下位16ビット=トラック番号 (1~80)
戻り値: なし
d0.l=エラーコード
備考: チャンネル番号とその対応デバイスはベースチャンネルモードによって変動する。

m_ch("FM"), (B0)のとき

チャンネル番号	1~8	内蔵FM音源チャンネル1~8
	9	内蔵AD PCM音源
	10~25	MIDIチャンネル1~16
	26~32	AD PCMチャンネル2~8 (PCM8独立チャンネルモード時)

m_ch("MIDI"), (B1)のとき

チャンネル番号	1~16	MIDIチャンネル1~16
	17~24	内蔵FM音源チャンネル1~8
	25	内蔵AD PCM音源
	26~32	AD PCMチャンネル2~8 (PCM8独立チャンネルモード時)

ファンクション$03　　m_vget

機能: FM音源音色データの取り出し
引数: d2.l=音色番号(1~200)
a1.l=データ格納領域のアドレス(55バイト分)
戻り値: 0(a1)~54(a1):音色データ
d0.l=エラーコード
備考 0(a1)~54(a1)とパラメータの対応

0:AF	(0~63)
1:OM	(0~15)
2:WF	(0~3)
3:SYC	(0,1)
4:SPD	(0~255)
5:PMD	(0~127)
6:AMD	(0~127)
7:PMS	(0~7)
8:AMS	(0~3)
9:8PAN	(0~3)

10:DUMMY
11(OP1),22(OP2),33(OP3),44(OP4):AR (0~31)
12(OP1),23(OP2),34(OP3),45(OP4):1DR (0~31)
13(OP1),24(OP2),35(OP3),46(OP4):2DR (0~31)
14(OP1),25(OP2),36(OP3),47(OP4):RR (0~15)
15(OP1),26(OP2),37(OP3),48(OP4):1DL (0~15)
16(OP1),27(OP2),38(OP3),49(OP4):TL (0~127)
17(OP1),28(OP2),39(OP3),50(OP4):KS (0~3)
18(OP1),29(OP2),40(OP3),51(OP4):MUL(0~15)
19(OP1),30(OP2),41(OP3),52(OP4):DT1 (0~7)
20(OP1),31(OP2),42(OP3),53(OP4):DT2 (0~3)
21(OP1),32(OP2),43(OP3),54(OP4):AME(0,1)

パラメータの機能や詳しい解説は専門書や取扱説明書のOPMDRV3.X,「BASICマニュアル」のM_VSET()やM_VGET()の項を参照のこと。

ファンクション$04　　m_vset

機能: FM音源音色の設定
引数: d2.l=音色番号(1~200)
a1.l=データ格納領域のアドレス(55バイト分)

戻り値: なし
d0.l=エラーコード
備考: パラメータフォーマットについてはファンクション$03の備考を参照。

ファンクション$05　m_tempo

機能: テンポの設定/取り出し
引数: 20≦d2.l≦300:音楽的テンポ値(1分間に4分音符を何個演奏するか)
d2.l=−1:d0.lに現在のテンポ値,a0.lに現在のタイマ値を返す
戻り値: なし
d0.l=エラーコード

ファンクション$06　m_trk

機能: MMLデータのバッファへの書き込み
引数: d2.l=トラック番号(1〜80)
a1.l=MML文字列データ格納アドレス
戻り値: なし
d0.l=エラーコード

ファンクション$07　m_free

機能: トラックバッファの空き容量
引数: d2.l=トラック番号(1〜80)
戻り値: d0.l=空き容量値
備考: ZMD演奏中の場合は戻り値の内容は保証されない。

ファンクション$08　m_play

機能: 演奏開始
引数: トラック1〜32が'd2.lのビット0〜31
トラック33〜64が'd3.lのビット0〜31
トラック65〜80が'd4.wのビット0〜15
に対応し,ビット=1で指定されたトラックの演奏を開始する。ただし,d2.l=d3.l=d4.w=0の場合は全トラック演奏開始する。
戻り値: なし
d0.l=エラーコード

ファンクション$09　m_stat

機能: 演奏状態の検査
引数: チャンネル1〜32が'd2.lのビット0〜31に対応し,
ビット=1のチャンネルに対して検査を行う。
d2.l=0のときは全チャンネル検査を行う
戻り値: 検査査定されたチャンネルのうちどれかひとつでも演奏中なら,d0.l=1を返す。
d2.l=0で全チャンネルを検査対象にした場合は,演奏中のチャンネルに対応したビット=1をd0.lに返す。
備考: チャンネル番号とその対応デバイスはベースチャンネルモードによって変動する。

m_ch("FM"),(B0)のとき

チャンネル番号		
	1〜8	内蔵FM音源チャンネル1〜8
	9	内蔵AD PCM音源
	10〜25	MIDIチャンネル1〜16
	26〜32	AD PCMチャンネル2〜8(PCM8独立チャンネルモード時)

m_ch("MIDI"),(B1)のとき

チャンネル番号		
	1〜16	MIDIチャンネル1〜16
	17〜24	内蔵FM音源チャンネル1〜8
	25	内蔵AD PCM音源
	26〜32	AD PCMチャンネル2〜8(PCM8独立チャンネルモード時)

ファンクション$0A　m_stop

機能: 演奏停止
引数: トラック1〜32が'd2.lのビット0〜31
トラック33〜64が'd3.lのビット0〜31
トラック65〜80が'd4.wのビット0〜15
に対応し,ビット=1で指定されたトラックの演奏を停止する。ただし,d2.l=d3.l=d4.w=0の場合は全トラック演奏停止する。
戻り値: なし
d0.l=エラーコード

ファンクション$0B　m_cont

機能: 演奏再開
引数: トラック1〜32が'd2.lのビット0〜31
トラック33〜64が'd3.lのビット0〜31
トラック65〜80が'd4.wのビット0〜15
に対応し,ビット=1で指定されたトラックの演奏を再開する。ただし,d2.l=d3.l=d4.w=0の場合は全トラック演奏再開する。
戻り値: なし
d0.l=エラーコード

ファンクション$0C　m_atoi

機能: トラックバッファの先頭アドレスを得る
引数: d2.l=トラック番号(1〜80)
戻り値: d0.l=トラックバッファの先頭アドレス値

ファンクション$0D　init_all

機能: 音源ドライバの初期設定
引数: なし
戻り値: なし

ファンクション$0E　int_stop

機能: 演奏割り込みの中断
引数: なし
戻り値: なし

ファンクション$0F　m_play2

機能: 前回に実行されたm_playをもう一度実行する
引数: なし
戻り値: なし
備考: 前回のm_play実行時のパラメータでもう一度m_playを呼び出す。ファンクション$11で演奏を開始したあと,このファンクションを実行しても無意味。

ファンクション$10　adpcm_read

機能　AD PCMデータの登録と加工
引数　a1.l=ファイルネーム格納アドレス
または
(a1).l=ソースデータナンバー(0〜511)
d2.l=設定ノートナンバー(0〜511)
d3の上位16ビット=ピッチシフトパラメータ
(0-11:ピッチダウン
12:ニュートラル
13-24:ピッチアップ)
d3の下位16ビット=ボリュームパラメータ
(1%〜300%)
d3.l=0でデータにノンタッチ
d4の上位16ビット=ディレイカウント(0〜65535)
d4の下位16ビット=ミキシング元ノートナンバー(0〜511)
=−1でミキシングは行わない
d5の上位16ビット=オフセットカウント(0〜65535)
d5の下位16ビット=カットサイズ(0〜65535)
d5.l=0でデータの切り出しを行わない
d6.l≠0で逆転
d6.l=0で逆転なし

d7の上位16ビット＝オフセットカウント(0～65535)
d7のビット8～15＝モード
(－1:フェードイン,＋1:フェードアウト)
d7のビット0～7＝フェードイン/アウトレベル(0～127)

戻り値: なし
d0.l＝エラーコード

ファンクション$11　play_cnv_data

機能: ZMDデータの演奏
引数: d2.l＝データ総サイズ
d2.l＝0の場合はドライバ内にデータを転送せず即演奏(高速応答)
a1.l＝演奏データ格納アドレス(備考参照)
戻り値: なし
d0.l＝エラーコード
備考: ZMDデータの構造

Offset	＋0	:$10(.b)	偶数アドレスから
	＋1～＋6	:'ZmuSiC'	メモリ上になくても構わない
	＋7	:Version Number(.b)	
		a1.lで指し示すべきアドレス(奇数アドレス)	

なお,内部フォーマットについての詳しい解説はMEASURE12参照。

ファンクション$12　se_play

機能: 効果音演奏
引数: d2.l＝何トラックから割り込ませるか(1～32)
a1.l＝演奏データ格納アドレス(備考参照)
戻り値: なし
備考: 効果音の構造

Offset	＋0	:$10(.b)	偶数アドレス
	＋1～＋6	:'ZmuSiC'	メモリ上になくても構わない
	＋7	:Version Number(.b)	
	＋8～	:共通コマンド	
	＋?	:Number of tracks(.w)	
		←a1.lで指し示すべきアドレス(偶数アドレス)	

なお,内部フォーマットについての詳しい解説はMEASURE12参照。

ファンクション$13　se_adpcm1

機能: AD PCMの効果音演奏
引数: a1.l＝AD PCMデータ格納アドレス
d2.l＝AD PCMデータサイズ
d3(bit00～07)＝PAN(0～3)
d3(bit08～15)＝FRQ(0～4)
d3(bit16～23)＝優先レベル(低0～255高)
戻り値: なし
備考: 優先レベルの例

現在鳴っている音の優先レベル	新しく鳴らす音の優先レベル	結果
0	0	新しいのが鳴る
0	1	新しいのが鳴る
1	0	新しいのが拒否される

通常は0(従来通り)で構わない。ゲームなどの面と面のつなぎのメッセージやイベントのメッセージなどの爆発音などの効果音に邪魔されたくないときに使用するとよい。
優先レベルはPCM8独立チャンネルモード時は意味をなさない。

ファンクション$14　se_adpcm2

機能: Z-MUSIC内に登録済みのAD PCM音の効果音演奏
引数: d2.l＝ノート番号(0～511)
d3(bit00～07)＝PAN(0～3)
d3(bit08～15)＝FRQ(0～4)
d3(bit16～23)＝優先レベル(低0～255高)
戻り値: なし
備考: 優先レベルについてはファンクション$13と同様。

ファンクション$15　set_ch_mode

機能: ベースチャンネルモード指定
引数: d2.l≠0 MIDIチャンネル基準モード
d2.l＝0 FMチャンネル基準モード
戻り値: なし
備考: このファンクションの設定内容によってチャンネル番号と対応デバイスが変化する。

●d2.l＝0のとき(m_ch("FM"),(B0)のとき)

チャンネル番号	1～8	内蔵FM音源チャンネル1～8
	9	内蔵AD PCM音源
	10～25	MIDIチャンネル1～16
	26～32	AD PCMチャンネル2～8(PCM8時)

●d2.l≠0のとき(m_ch("MIDI"),(B1)のとき)

チャンネル番号	1～16	MIDIチャンネル1～16
	17～24	内蔵FM音源チャンネル1～8
	25	内蔵AD PCM音源
	26～32	AD PCMチャンネル2～8(PCM8時)

ファンクション$16　midi_rec

機能: MIDIデータの記録
引数: なし
戻り値: なし
備考: ファンクション実行後,MIDI IN端子より受信したデータをトラックバッファへ格納していく。バッファオーバーフローを起こすとビープ音が鳴る。この場合データ内容は意味をなさない。

ファンクション$17　midi_rec_end

機能: MIDIデータ記録モード終了
引数: d2.l＝0 文字列データへ変換
d2.l≠0 データを加工しない
戻り値: d0.l＝データサイズ
a0.l＝data address
エラーの場合はd0.l＝エラーコード/a0.l＝0となる
備考: ファンクション$16を終了する。文字列データ(d2.l＝0)へ変換する際にワークエリアを使用する。文字列データ作成時にワークが不足しているとビープ音が鳴る。この場合データ内容は意味をなさない。

ファンクション$18　midi_trns

機能: MIDIデータの転送
引数: d2.l＝0 ASCII文字列データ転送モード
d2.l≠0 バイナリデータ転送モード(データサイズ)
a1.l＝データ先頭アドレス
戻り値: なし
d0.l＝エラーコード

ファンクション$19　calc_total

機能: 各トラックの演奏データのステップタイムの合計値を求め演奏トラックワークのp_totalに格納する
引数: d2≠0:計算してp_totalに格納する
d2＝0:計算してp_totalに格納して結果を画面に表示する
戻り値: d0.l＝－1:演奏データにエラーが残存しているため,計算を行えなかった
d0.l＝－2:どのトラックにも演奏データは存在しなかった
d0.l＝0:正常終了
備考: トラックバッファに格納されている演奏データについて計算処理を行う。演奏中の場合は,演奏を停止して計算処理をする。

ファンクション$1a　fade_out

機能: 演奏中の全トラックに対してフェードイン/アウトを行う
引数: d2.l=−85〜−1:フェードイン
d2.l=0:フェードアウトモード解除
d2.l=1〜85:フェードアウト
戻り値: d0.l=0:設定完了
d0.l=−1:設定失敗
(すでにフェードイン/フェードアウト実行中)
備考: d2.l=0以外ではd2.lの絶対値は音量の増減スピードに相当し，絶対値が大きいほど増減スピードは速くなる。d2.l=0では単に各トラックの音量が下がるのを停止するのみ。通常音量には戻すことはしない。演奏中のトラックのうち1トラックでもフェードイン/アウトを実行中なら本ファンクションは拒絶される(戻り値:d0.l=−1になる)。

ファンクション$1b m_vset2

機能: FM音源音色の設定(AL/FB分離フォーマット形式)
引数: d2.l=音色番号(1〜200)
a1.l=パラメータデータ格納アドレス(55バイト分)
戻り値: なし
d0.l=エラーコード
備考 0(a1)〜54(a1)とパラメータの対応

00(OP1),11(OP2),22(OP3),33(OP4):AR	(0〜31)
01(OP1),12(OP2),23(OP3),34(OP4):1DR	(0〜31)
02(OP1),13(OP2),24(OP3),35(OP4):2DR	(0〜31)
03(OP1),14(OP2),25(OP3),36(OP4):RR	(0〜15)
04(OP1),15(OP2),26(OP3),37(OP4):1DL	(0〜15)
05(OP1),16(OP2),27(OP3),38(OP4):TL	(0〜127)
06(OP1),17(OP2),28(OP3),39(OP4):KS	(0〜3)
07(OP1),18(OP2),29(OP3),40(OP4):MUL	(0〜15)
08(OP1),19(OP2),30(OP3),41(OP4):DT1	(0〜7)
09(OP1),20(OP2),31(OP3),42(OP4):DT2	(0〜3)
10(OP1),21(OP2),32(OP3),43(OP4):AME	(0,1)
44:AL	(0〜7)
45:FB	(0〜7)
46:OM	(0〜15)
47:PAN	(0〜3)
48:WF	(0〜3)
49:SYC	(0,1)
50:SPD	(0〜255)
51:PMD	(0〜127)
52:AMD	(0〜127)
53:PMS	(0〜7)
54:AMS	(0〜3)

パラメータの機能や詳しい解説は専門書や取扱説明書のOPM DRV3.Xまたは「X-BASICマニュアル」のM_VSET()やM_VGET()の項を参照のこと。

ファンクション$1c send_rd_exc

機能: ローランドエクスクルーシブデータの転送
引数: d2.l=データサイズ
d3(bit16〜31)=デバイスID
d3(bit0〜15)=モデルID
a1.l=データ格納アドレス
戻り値: なし

ファンクション$1d send_exc

機能: エクスクルーシブデータの転送
引数: d2.l=データサイズ
a1.l=データ格納アドレス
戻り値: なし

ファンクション$1e sc55_p_rsv

機能: ローランドSC-55のパーシャル(ボイス)リザーブ
引数: d2.l=データサイズ
(d2.l=16でないと楽器の状態は保証されず)
d3.b=デバイスID(d3.b=−1で前回のを使用する)
a1.l=パラメータデータ格納アドレス
戻り値: なし
備考: パラメータフォーマットはMEASURE3 sc55_v_reserve()参照。

ファンクション$1f sc55_reverb

機能: ローランドSC-55のリバーブパラメータセット
引数: d2.l=データサイズ
(1≦d2.l≦7でないと楽器の状態は保証されず)
d3.b=デバイスID(d3.b=−1で前回のを使用する)
a1.l=パラメータデータ格納アドレス
戻り値: なし
備考: パラメータフォーマットはMEASURE3 sc55_reverb()参照。

ファンクション$20 sc55_chorus

機能: ローランドSC-55のコーラスパラメータセット
引数: d2.l=データサイズ
(1≦d2.l≦8でないと楽器の状態は保証されず)
d3.b=デバイスID(d3.b=−1で前回のを使用する)
a1.l=パラメータデータ格納アドレス
戻り値: なし
備考: パラメータフォーマットはMEASURE3 sc55_chorus()参照。

ファンクション$21 sc55_part_parameter

機能: ローランドSC-55のパートパラメータ
引数: d2.l=データサイズ
(1≦d2.l≦119でないと楽器の状態は保証されず)
d3(bit16〜31)=PART NUMBER(1〜16)
d3.b=デバイスID(d3.b=−1で前回のを使用する)
a1.l=パラメータデータ格納アドレス
戻り値: なし
備考: パラメータフォーマットはMEASURE3 sc55_part_setup()参照。

ファンクション$22 sc55_drum_parameter

機能: ローランドSC-55のドラムパラメータ
引数: d2.l=データサイズ
(1≦d2.l≦8でないと楽器の状態は保証されず)
d3(bit24〜31)=MAP NUMBER(0〜1)
d3(bit16〜23)=NOTE NUMBER(0〜127)
d3.b=デバイスID(d3.b=−1で前回のを使用する)
a1.l=データ格納アドレス
戻り値: なし
備考: パラメータフォーマットはMEASURE3 sc55_drum_setup()参照。

ファンクション$23 sc55_print

機能: ローランドSC-55の画面に文字列を表示する
引数: d2.l=文字列サイズ
(1≦d2.l≦32でないと楽器の状態は保証されず)
d3.b=デバイスID(d3.b=−1で前回のを使用する)
a1.l=文字列格納アドレス
戻り値: なし

ファンクション$24 sc55_display

機能: ローランドSC-55の画面にグラフィックを表示する
引数: d3.b=デバイスID(d3.b=-1で前回のを使用する)
a1.l=データ格納アドレス(パラメータの数は16個に固定)
(必ず偶数。さもなくばアドレスエラーが発生する)
戻り値: なし
備考: パラメータフォーマットはMEASURE3 sc55_display()参

照。

ファンクション$25　　mt32_p_rsv

機能: ローランドMT-32のパーシャルリザーブを行う
引数: d2.l=データサイズ
(d2.l=9でないと楽器の状態は保証されず)
d3.b=デバイスID(d3.b=−1で前回のを使用する)
a1.l=データ格納アドレス
戻り値: なし
備考: パラメータフォーマットはMEASURE3 mt32_p_reserve()参照。

ファンクション$26　　mt32_reverb

機能: ローランドMT-32のリバーブパラメータの設定
引数: d2.l=データサイズ
(1≦d2.l≦3でないと楽器の状態は保証されず)
d3.b=デバイスＩＤ(d3.b=−1で前回のを使用する)
a1.l=データ格納アドレス
戻り値: なし
備考: パラメータフォーマットはMEASURE3 mt32_reverb()参照。

ファンクション$27　　mt32_setup

機能: ローランドMT-32のパートパラメータの設定
引数: d2.l=データサイズ
(1≦d2.l≦9でないと楽器の状態は保証されず)
d3.b=デバイスID(d3.b=−1で前回のを使用する)
a1.l=データ格納アドレス
戻り値: なし
備考: パラメータフォーマットはMEASURE3 mt32_part_setup()参照。

ファンクション$28　　mt32_drum

機能: ローランドMT-32のドラムパラメータの設定
引数: d2.l=データサイズ
(1≦d2.l≦4でないと楽器の状態は保証されず)
d3(bit16〜31)=NOTE NUMBER(0〜127)
(ただし受信可能範囲(24〜87)以外は無視される)
d3.b=デバイスID(d3.b=−1で前回のを使用する)
a1.l=データ格納アドレス
戻り値: なし
備考: パラメータフォーマットはMEASURE3 mt32_drum_setup()参照。

ファンクション$29　　mt32_common

機能: ローランドMT-32の音色のコモンパラメータの設定
引数: d2.l=データサイズ
(1≦d2.l≦15でないと楽器の状態は保証されず)
d3(bit16〜31)=PROGRAM NUMBER(1〜64)
d3.b=デバイスID(d3.b=−1で前回のを使用する)
a1.l=データ格納アドレス
戻り値: なし
備考: a1.lから格納されているパラメータデータは以下の書式に従う。
PROGRAM NAME(最大10文字),$00(.b),Param1(.b),…,Param4(.b)
パラメータフォーマットはMEASURE3 mt32_common()参照。

ファンクション$2a　　mt32_partial

機能: ローランドMT-32の音色のパーシャルパラメータの設定
引数: d2.l=データサイズ
(1≦d2.l≦58でないと楽器の状態は保証されず)
d3(bit24〜31)=PROGRAM NUMBER(1〜64)
d3(bit16〜23)=PARTIAL NUMBER(1〜4)
d3.b=デバイスID(d3.b=−1で前回のを使用する)
a1.l=データ格納アドレス
戻り値: なし
備考: パラメータフォーマットはMEASURE3 mt32_partial()参照。

ファンクション$2b　　mt32_patch

機能: ローランドMT-32のパッチパラメータの設定
引数: d2.l=データサイズ
(1≦d2.l≦7でないと楽器の状態は保証されず)
d3(bit16〜31)=PATCH NUMBER(1〜128)
d3.b=デバイスID(d3.b=−1で前回のを使用する)
a1.l=データ格納アドレス
戻り値: なし
備考: パラメータフォーマットはMEASURE3 mt32_patch()参照。

ファンクション$2c　　mt32_print

機能: ローランドMT-32の画面に文字列を表示する
引数: d2.l=文字列サイズ
(1≦d2.l≦20でないと楽器の状態は保証されず)
d3.b=デバイスID(d3.b=−1で前回のを使用する)
a1.l=文字列格納アドレス
戻り値: なし

ファンクション$2d　　u220_setup

機能: ローランドU220のセットアップパラメータの設定
引数: d2.l=データサイズ
(d2.l=7でないと楽器の状態は保証されず)
d3.b=デバイスID(d3.b=−1で前回のを使用する)
a1.l=データ格納アドレス
戻り値: なし
備考: パラメータフォーマットはMEASURE3 u220_setup()参照。

ファンクション$2e　　u220_common

機能: ローランドU220テンポラリパッチコモンパラメータ設定
引数: d2.l=データサイズ
(d2.l=18でないと楽器の状態は保証されず)
d3.b=デバイスID(d3.b=−1で前回のを使用する)
a1.l=データ格納アドレス
戻り値: なし
備考: パラメータフォーマットはMEASURE3 u220_setup()参照。

ファンクション$2f　　u220_d_setup

機能: ローランドU220テンポラリパッチドラムパラメータ設定
引数: d2.l=データサイズ
(d2.l=7でないと楽器の状態は保証されず)
d3.b=デバイスID(d3.b=−1で前回のを使用する)
a1.l=データ格納アドレス
戻り値: なし
備考: パラメータフォーマットはMEASURE3 u220_drum_setup()参照。

ファンクション$30　　u220_p_setup

機能: ローランドU220テンポラリパッチパートパラメータ設定
引数: d2.l=データサイズ
(d2.l=13でないと楽器の状態は保証されず)
d3(bit16〜31)=PART NUMBER(1〜6)
d3.b=デバイスID(d3.b=−1で前回のを使用する)
a1.l=データ格納アドレス
戻り値: なし
備考: パラメータフォーマットはMEASURE3 u220_part_setup()参照。

ファンクション$31　　u220_print

機能:　ローランドU220テンポラリパッチエリアにメッセージ表示
引数:　d2.l＝文字列サイズ
　　(1≦d2.l≦12でないと楽器の状態は保証されず)
　　d3.b＝デバイスID(d3.b＝−1で前回のを使用する)
　　a1.l＝文字列格納アドレス
戻り値: なし

ファンクション$32　　u220_timbre

機能:　ローランドU220ティンバーパラメータの設定
引数:　d2.l＝データサイズ
　　(27≦d2.l≦39でないと楽器の状態は保証されず)
　　d3(bit16〜31)＝PROGRAM NUMBER(1〜128)
　　d3.b＝デバイスID(d3.b＝−1で前回のを使用する)
　　a1.l＝データ格納アドレス
戻り値: なし
備考:　a1.lから格納されているデータは以下の書式に従う。
　Timbre　Name(12文字以内),$00(.b),Param1(.b),…,param26(.b)
　　パラメータフォーマットはMEASURE3 u220_timbre()参照。

ファンクション$33　　u220_drum

機能:　ローランドU220テンポラリエリアのリズムキット書き換え
引数:　d2.l＝データサイズ
　　(1≦d2.l≦20でないと楽器の状態は保証されず)
　　d3(bit16〜31)＝NOTE NUMBER(35〜99)
　　d3.b＝デバイスID(d3.b＝−1で前回のを使用する)
　　a1.l＝データ格納アドレス
戻り値: なし
備考:　パラメータフォーマットはMEASURE3 u220_drum_inst()参照。

ファンクション$34　　m1_midi_ch

機能:　コルグM1のシーケンサSONG0の受信MIDIチャンネルの設定
引数:　a1.l＝データ格納アドレス
戻り値: なし
備考:　0(a1)＝パート1の受信MIDIチャンネル(1〜16)
　　1(a1)＝パート2の受信MIDIチャンネル(1〜16)
　　:
　　7(a1)＝パート8の受信MIDIチャンネル(1〜16)

ファンクション$35　　send_to_m1

機能:　ファンクション$34,$36,$37で設定したパラメータを
　　コルグM1のシーケンサSONG0へ設定
引数:　d3.b＝デバイスID(d3.b＝−1で前回のを使用する)
戻り値: なし

ファンクション$36　　m1_p_setup

機能:　コルグM1のシーケンサSONG0のトラックパラメータ設定
引数:　a1.l＝データ格納アドレス
　　(パラメータデータは必ず40個なければならない)
戻り値: なし
備考:　パラメータフォーマットはMEASURE3 m1_part_setup()参照。

ファンクション$37　　m1_e_setup

機能:　コルグM1のシーケンサSONG0エフェクトパラメータ設定
引数:　a1.l＝データ格納アドレス
　　(パラメータは必ず25個なければならない)
戻り値: なし
備考:　パラメータフォーマットはMEASURE3 m1_effect_setup()参照。

ファンクション$38　　m1_print

機能:　コルグM1のシーケンサSONG0のソングネームの設定
引数:　d2.l＝文字列サイズ
　　(1≦d2.l≦10でないと楽器の状態は保証されず)
　　a1.l＝文字列格納アドレス
戻り値: なし

ファンクション$39　　block_adpcm_data

機能:　ZPDデータの登録
引数:　a1.l＝ZPDファイルのファイル名格納アドレス
戻り値: なし
　　d0.l＝エラーコード
備考:　ファイルネームは文字列,$00(.b)の書式に従う。

ファンクション$3a　　get_trk_tbl

機能:　演奏トラックテーブル/絶対チャンネルテーブルアドレス
引数:　なし
戻り値: d0.l＝絶対チャンネルテーブルのアドレス
　　a0.l＝演奏トラックテーブルのアドレス
備考:　m_ch("FM"),(B0)のときはd0.lの指し示すアドレスから,
　　0,1,2,3,4,5,6,7,8,9,10,11,12,13,
　　14,15,16,17,18,19,20,21,22,23,24
　m_ch("MIDI"),(B1)のときはd0.lの指し示すアドレスから,
　　9,10,11,12,13,14,15,16,17,18,19,
　　20,21,22,23,24,0,1,2,3,4,5,6,7,8
　a0.lの指し示すアドレスからは演奏すべきトラックナンバーが最大32個格納されている。終端コードは$ff(.b)。
　たとえばトラック1〜9を演奏する(している)ときは,
　　0,1,2,3,4,5,6,7,8,$ff　　(値はトラック番号−1)
のようになっている。

ファンクション$3b　set_loop_time

機能:　全トラックがd2.b回以上ループしたらa1.lへBSRする
引数:　d2.b＝0　loop time(s)　(0〜255)
　　a1.l＝飛び先アドレス(＝0で解除)
戻り値: なし

ファンクション$3c　　get_play_work

機能:　演奏トラックワークのアドレスを返す
引数:　d2.b＝TRK NUMBER(1〜80)
戻り値: a0.l＝演奏ワークアドレス
　　d0.l＝コンパイルワークアドレス
備考:　具体的なワーク内容についてはMEASURE13を参照。

ファンクション$3d　　get_timer_mode

機能:　どのタイマで割り込みを実現しているかを返す
引数:　なし
戻り値: d0.l＝0　timer A
　　d0.l＝1　timer B

ファンクション$3e　　set_fm_master_vol

機能:　FM音源のマスターボリュームを設定する
引数:　d2.l＝マスターボリューム値(0〜255)
戻り値: なし
備考:　演奏中は無効。ZMD作成前にのみ有効。

ファンクション$3f　　set_timer_value

機能:　タイマの値を設定する
引数:　d2.l＝タイマ値
戻り値: なし
備考:　設定した値は,現在テンポ源となっている割り込みタイマに対して直接書き込まれる。

ファンクション$40　　release_support

機能：　ZMUSIC.X用(またはZMSC.X用)のサブプログラムの名前を登録する(あるいは登録したものを取り消す)

●登録時

引数：　a1.l＝filename address　(解除プログラムの名前を登録)

戻り値：d0.l＝0≦result code　正常登録された(ok)

d0.l＝−1　すでに4つ登録されており，これ以上登録できない(error)

備考：　名前を登録しておくとZMUSIC.X解除時に一緒に解除する。filename(100文字以内)は，

'a:¥bin¥zp -r',0

のようなフォーマットとする最大4つまで登録可能。

●キャンセル時

引数：　a1.l＝0

d2.l＝登録終了時に返ってきたresult code

戻り値：d0.l＝0　キャンセル完了(ok)

d0.l＝−1 エラー

備考：　result codeにより任意のプログラムの登録キャンセルが可能。

ファンクション$41　jump_active

機能：　[!]/[@]/[end]コマンドの有効化/無効化スイッチ

引数：　d2.l＝0　無効

d2.l≠0　有効

戻り値：なし

備考：　演奏中は無効。ZMD作成前にのみ有効。

ファンクション$42　set_mclk

機能：　全音符の絶対音長を設定する

引数：　d2.l＝1〜254

戻り値：なし

備考：　演奏中は無効。ZMD作成前にのみ有効。

ファンクション$43　picture_sync

機能：　映像同期モードのオン/オフ

引数：　d2.l＝0　映像同期モードオフ

≠0　映像同期モードオン

戻り値：

状態	設定状態	結果
ON	OFF	OFF(正常終了:d0.l=0)
ON	ON	エラー(d0.l=−1)
OFF	OFF	エラー(d0.l=−1)
OFF	ON	ON(正常終了:d0.l=0)

備考：　具体的な使用方法についてはMEASURE14を参照のこと。

ファンクション$44　mask_channels

機能：　演奏チャンネルのマスクキングを行う

引数：　d2.l＝マスクしたいチャンネルのビットパターン

bit＝1 mask ch

bit＝0 enable ch

備考：　m_ch("fm")または(B0)のときはビット0〜24が'FM1〜8，ADPCM1，MIDI1〜16に対応し，m_ch("midi")または(B1)のときはビット0〜24が'MIDI1〜16，FM1〜8，ADPCM1に対応する。

m_ch(),(Bn)によらずPCM8独立チャンネルモード時はビット25〜31が'ADPCM2〜8に対応する。

ファンクション$45　buffer_info

機能：　各バッファのアドレス，サイズ，終了アドレス情報を得る

引数：　なし

戻り値：

0(a0)＝トラックバッファの先頭アドレス　trk_top

4(a0)＝トラックバッファのサイズ　trk_buf_size

8(a0)＝トラックバッファ終了アドレス　trk_buf_end

＝ZMUSIC.Xの最終アドレス　(dev_end_adr)

12(a0)＝AD PCMバッファ先頭アドレス　adpcm_buffer_top

16(a0)＝AD PCMバッファサイズ　adpcm_buffer_size

20(a0)＝AD PCMバッファ終了アドレス　adpcm_buffer_end

24(a0)＝汎用ワークエリア先頭アドレス　adpcm_work_top

28(a0)＝汎用ワークエリアサイズ　adpcm_work_size

32(a0)＝汎用ワークエリア終了アドレス　adpcm_work_end

36(a0)＝新規に登録されるAD PCMデータが格納されるアドレス　adpcm_buffer_next

adpcm_buffer_top≦adpcm_buffer_next≦adpcm_buffer_end

40(a0)＝次にワークが使用される場合の汎用ワーク使用可能開始アドレス　adpcm_work_now

adpcm_work_top≦adpcm_work_now≦adpcm_work_end

44(a0)＝汎用ワークエリアの本当のサイズ　adpcm_work_true_size

48(a0)＝演奏トラックワーク格納アドレス　seq_wk_tbl

52(a0)＝効果音トラックワーク格納アドレス　seq_wk_tbl2

56(a0)＝AD PCMデータの管理テーブル格納アドレス　adpcm_tbl

60(a0)＝波形メモリの管理テーブル格納アドレス　wave_tbl

64(a0)＝FM音源音色バッファのアドレス　neiro

68(a0)＝各トラックの先頭アドレス格納テーブルアドレス(1,2,3,...80)　trk_po_tbl

72(a0)＝m_allocで確保した各トラックのサイズ格納アドレス(1,2,3,...80)　trk_len_tbl

備考：　各バッファのアドレスを書き換えることによって，バッファを外部のものに切り替えたりすることも可能となる。たとえばAD PCMバッファをZMUSIC.X常駐時に確保しなかったとしても，あとで，別プログラムが'用意した領域を指し示すようにadpcm_buffer_topなどを書き換え，ポインタであるadpcm_buffer_nextをadpcm_buffer next＝adpcm_buffer_topと初期化してやれば，その領域をAD PCMバッファとして使用できるようになる。

汎用ワークエリアはAD PCMコンフィギュレーションファイルの一時退避場所としてや，波形メモリのバッファ(ZMS時)，AD PCMデータ加工時のワークとして使用されておりデータ内容保守の関係で状況に応じてワークサイズ(adpcm_work_size)が'伸縮する。よって汎用ワークエリアを外部プログラムで用意したものと差し換えるにはadpcm_work_top,adpcm_work_size,adpcm_work_endを設定しさらにadpcm_work_now＝adpcm_work_top，adpcm_work true_size＝adpcm_work_sizeとする必要がある。

ファンクション$46　set_zpd_tbl

機能：　ZPDデータの情報を登録する

引数：　a1.l＝ZPD情報テーブルのアドレス

(a1.l＝ZPDファイルの先頭アドレス＋8に相当)

戻り値：なし

備考：　AD PCMデータの管理テーブルであるadpcm_tblの先頭からノート番号0から順番に，

AD PCMデータ格納アドレス(.l),AD PCMデータサイズ(.l)

.....

のように格納されている。adpcm_tblのアドレスはファンクション$45で求めることができるので，まったく外部のプログラムからAD PCMデータの差し換えをすることも可能。

ファンクション$47　set_output_level

機能：　演奏チャンネルの出力割合を設定する

引数：　d2.l＝変更したいチャンネルのビットパターン

bit＝1 変更希望チャンネル

bit＝0 変更しないチャンネル

d3.b＝0〜127　(出力レベル。0は消音)

d3.b＝128以上　(通常に戻す)

備考：

m_ch("fm")または(B0)のとき
ビット0～24が「FM1～8,ADPCM1,MIDI1～16に対応
m_ch("midi")または(B1)のとき
ビット0～24が「MIDI1～16,FM1～8,ADPCM1に対応
m_ch(),(Bn)によらずPCM独立チャンネルモード時はビット25～31が「ADPCM2～8に対応する。
FM音源の場合は変化が「指数関数的であるので注意(d3.b=90付近でほとんど消音する)。

ファンクション$48　eox_wait

機能: EOX($F7)送信後のウェイト時間を設定する
引数: d2.l=ウェイト値(0～65535)
備考: d2.l=0　ウェイトなし
d2.l=1　ウェイト最小(約0.017秒)
d2.l=65535　ウェイト最大(約1092秒)

ファンクション$49　set_wave_form1

機能: 波形メモリの登録(その1)
引数: a1.l=波形データ先頭アドレス
d2.l=データサイズ(データ数ではなく物理サイズ)
d3(bit16～23)=波形番号(8～31)
d3(bit24～31)=ループタイプ(0～2)
d3.w=ループポイント(0～65535)
戻り値: なし
d0.l=エラーコード
備考: 波形メモリの具体的な仕様についてはMEASURE5を参照。

ファンクション$4a　set_wave_form2

機能 波形メモリの登録(その2)
引数 a1.l=波形データ先頭アドレス
d2.l=データ数(データの物理サイズではなく個数)
d3(bit16～23)=波形番号(8～31)
d3(bit24～31)=ループタイプ(0～2)
d3.w=ループポイント(0～65535)
戻り値: なし
d0.l=エラーコード
備考: 波形メモリの具体的な仕様についてはMEASURE5を参照。

ファンクション$4b　mask_tracks

機能: トラック単位のマスク
引数: d2.l=そのトラックのマスクを解除する(1～80)
d2.l=0(全トラックマスク解除)
d2.l=そのトラックをマスクする(－80～－1)
備考: すでにファンクション$44のチャンネルマスクでマスクしてあるとそれらは以後トラックマスクとして処理される。つまり以後本ファンクションで制御可能となる。

ファンクション$4c　set_output_level2

機能: トラック単位の出力割合を設定する
引数: d2.l=変更したいトラック番号(1～80)
d2.l=0　(全トラックを操作対象とする)
d3.b=0～127　(出力レベル。0は消音)
d3.b=128以上　(通常に戻す)
備考: すでにファンクション$44のチャンネルマスクでマスクしてあるとそれらは以後トラックマスクとして処理される。つまり以後本ファンクションで制御可能となる。

ファンクション$4d　get_loop_time

機能: 演奏中の現在の繰り返し回数を得る。
引数: なし
戻り値: d0.l=1～255(現在の繰り返し回数)
d0.l<0(演奏終了,または停止中)

ファンクション$4e　get_1st_comment

機能: 演奏中の演奏データに登場した最初のコメント文を得る
引数: なし
戻り値: a0.l=文字列格納アドレス。文字列の終端は$00。

ファンクション$4f　int_start

機能: Z-MUSICの割り込みを再開する。
引数: なし
戻り値: なし
備考: 通常は用いる必要はない。ファンクション$0eで割り込みを停止/禁止したあとで,割り込みを復帰する目的などで使う。

ファンクション$50　zm_status

機能: Z-MUSICの常駐状況の情報テーブルのアドレスを返す
引数: なし
戻り値: a0.l=情報テーブルの格納アドレス
備考: 情報テーブルの内容は以下のとおり。

```
0(a0).b=MIDIチャンネル有効か (ne=有効,0=無効)
1(a0).b=$FF     RS-232C MIDI
        $00     MIDI I/F(CZ-6BM1相当品)
        $01     POLYPHON MIDI
2(a0).b=強制MIDIチャンネル有効モードか (ne=YES,0=NO)
3(a0).b=AD PCMの発音にPCM8.Xを使用中か (ne=YES,0=NO)
4(a0).w=PCM8モード<0   独立チャンネルモード
        >0   ポリモード
6(a0).bの第0ビット=1 ジュークボックス類が常駐している
                =0 常駐していない
6(a0).bの第1ビット=1 演奏制御を割り込みで行うプログラムが常駐している(ZP /Dなど)
                =0 常駐していない
7(a0).b=$01     演奏待機状態(ファンクション$54参照)
        =$FF    演奏開始
  (このエリアを外部アプリケーションで$00にしておけば演奏開始や演奏待機状態になったことを知ることができる)
8(a0).b
   :            直前に読んだファイルのファイル名
98(a0).b        (ZPD,MDD,CNFなど)
-$1(a0)
 :              エラーコード格納エリア
-$8(a0)
-$c(a0).l       システムカウンタ
```

●エラーコード格納エリア
エラーが発生した場合,そのエラーコードが,このエリアに格納される。-$8(a0)がいちばん新しく発生したエラーで,-$1(a0)がいちばん昔に発生したエラー。古いものはどんどん切り捨てられていく。外部アプリケーション側でこの8バイトを初期化($00に)しておけば,その時点から発生したエラーを知ることができる。

●システムカウンタ
Z-MUSICの割り込みが発生されるたびにインクリメントされる。ここをアプリケーション側でクリアしておけばその時点からのステップカウントが求められる(Z-MUSICの割り込みは1ステップカウントごとに発生する)。
注意　ワークの書き換えはZ-MUSICプログラム本体を十分理解してから行うこと。

ファンクション$51　sc55_init

機能: SC-55の初期化を行う
引数: d3.b=デバイスID(d3.b=－1で前回のを使用する)
戻り値: なし

ファンクション$52　mt32_init

機能: MT-32の初期化を行う
引数: d3.b=デバイスID(d3.b=－1で前回のを使用する)

戻り値：なし

ファンクション$54　　intercept_play

機能：　演奏開始制御
引数：　d2.l＝-1 演奏させないモードに設定
　　d2.l＝0　演奏させないモードを解除，さらにトラック
　　バッファ内の演奏データを演奏開始
　　d2.l＝1　演奏させないモードを解除
戻り値：なし
備考：　演奏させないモードに設定したあとは，あとで必ずモード解除(d2.l=0かd2.l=1)を行うこと。

ファンクション$55　　m_inp1

機能：　1バイトMIDI入力
引数：　d2.l＝0 入力がない場合エラーコードを持って帰還
　　d2.l≠0 入力がない場合1/60×d2.l秒間受信を待つ
戻り値：d0.b=入力データ
　　ただし，d0.l=負の場合はエラー

ファンクション$56　　m_out1

機能：　1バイトMIDI出力
引数：　d2.b＝出力データ
戻り値：なし

ファンクション$57　　occupied_size

機能：　各バッファの実際の使用容量を返す
引数：　a1.l=戻り値の格納バッファアドレス(12バイト必要)
戻り値：なし
備考：　0(a1).l トラックバッファの使用容量
　　4(a1).l AD PCMバッファの使用容量
　　8(a1).l 汎用ワークエリアの使用容量

MEASURE11 エラーについて

ここではエラー発生時のエラーメッセージ，エラーの原因と対処方法について解説します。

11.1. MUSICZ.FNCのエラー

●エラー番号:1
パラメータの省略はできません
原因と処置:省略できないパラメータが省略されている。

●エラー番号:2
トラック番号が規定外です
原因と処置:トラック番号の有効範囲は1～80。

●エラー番号:3
ZMUSIC.Xが未登録です
原因と処置:ZMUSIC.Xを組み込んでいなければ演奏はできない。

●エラー番号:4
トラックバッファのサイズの指定が規定外です
原因と処置:トラックバッファサイズの有効範囲は100～65535。

●エラー番号:5
メッセージ:トラックバッファの確保ができません
原因と処置:トラックバッファが不足していると考えられるのでZMUSIC.Xを一度，解除してトラックバッファを大き目に設定して再常駐してみる。

●エラー番号:6
チャンネル番号が規定外です
原因と処置:チャンネル番号の有効範囲は1～25。ただしPCM8独立チャンネルモード時は1～32。

●エラー番号:7
配列の型が規定外です
原因と処置:配列変数の型を命令の仕様どおりにする。

●エラー番号:8
ZMUSIC.Xは現在動作を停止中です
原因と処置:外部のアプリケーションによりZMUSIC.Xが動作を停止している。そのアプリケーションでZMUSIC.Xを再始動させる。

●エラー番号:9 リザーブ
｜コマンドの使用法に誤りがあります
原因と処置:バンク番号は上位バイトと下位バイトを分けて書かなければならない。有効範囲は両方とも0～127である。

●エラー番号:10
この命令はコンパイルできません
原因と処置:その命令はステートメントではなくコマンドである。

●エラー番号:11
@Xコマンドの使用法に誤りがあります
原因と処置:値は最低1個は書かなければならない。

●エラー番号:12
@｜コマンドの使用法に誤りがあります
原因と処置:各パラメータの有効範囲は0～127。値の省略はできない。

●エラー番号:13
相対ボリュームの指定に誤りがあります
原因と処置:相対ボリュームのパラメータの有効範囲は1～127。

●エラー番号:14
@Cコマンドの使用法に誤りがあります
原因と処置:FM音源部とMIDI音源部ではその機能が違う点に注意。

●エラー番号:15
@Aコマンドの使用法に誤りがあります
原因と処置:振幅値の有効範囲は-127～127。また，パラメータが多すぎたりしていないか確認してみる。

●エラー番号:16
内蔵音源には無関係なコマンドです
原因と処置:MIDI専用の命令をFM音源やAD PCMに対して使用している。

●エラー番号:17
@Hコマンドの使用法に誤りがあります
原因と処置:モジュレーションディレイの有効範囲は0～65534。

●エラー番号:18
@Sコマンドの使用法に誤りがあります
原因と処置:モジュレーションスピードの有効範囲は1～16383。

●エラー番号:19
未定義のMMLです
原因と処置:ZMUSIC.Xには存在しないコマンドを実行しようとした。

●エラー番号:20
未定義の［～］コマンドです
原因と処置:ZMUSIC.Xには存在しないコマンドを実行しようとした。

●エラー番号:21
］がありません
原因と処置:［～］系のコマンドで終端の］がない。

●エラー番号:22
繰り返し回数が規定外です
原因と処置:繰り返し回数の有効範囲は1～255。

●エラー番号:23
繰り返し構造が異常です
原因と処置:多重ループを組んだ場合はネスト構造になっていなければならない。詳しくはMEASURE5 ｜:～:｜コマンドの項を参照。

●エラー番号:24
繰り返し番号が規定外です
原因と処置:繰り返し番号の有効範囲は1～255。

●エラー番号:25
オクターブ値がありません
原因と処置:オクターブの値を書く。

●エラー番号:26
オクターブ値が規定外です
原因と処置:オクターブ値の有効範囲は-1～9。

●エラー番号:27
音長が規定外です

原因と処置：音楽的音長の有効範囲は1～64。絶対音長の有効範囲は1～65534。

●エラー番号：28
トラックバッファが不足しています
原因と処置：'m_alloc()'命令または(M)命令で，そのトラックのトラックバッファを大き目に確保し直す。

●エラー番号：29
@Bコマンドの使用法に誤りがあります
原因と処置：ベンド値の有効範囲は-8192～8191。ディレイ値の有効範囲は0～32767。

●エラー番号：30
AD PCMには無関係なコマンドです
原因と処置：FM音源やMIDI音源用のコマンドはAD PCMに対して使用できない。

●エラー番号：31
タイの指定方法に誤りがあります
原因と処置：'&'コマンドや'^'コマンドの使用方法に誤りがあるか，'^'コマンドにおいて合計音長が65534を超えている可能性がある。

●エラー番号：32
Tコマンドの使用法に誤りがあります
原因と処置：テンポの値の有効範囲は以下のとおり。
タイマAモード：77～300
タイマBモード：20～300

●エラー番号：33
@Tコマンドの使用法に誤りがあります
原因と処置：タイマ値の有効範囲は，
タイマAモード：0～1023
タイマBモード：0～255

●エラー番号：34
音長値がありません
原因と処置：音長を記述する。

●エラー番号：35
音量値がありません
原因と処置：音量値を記述する。

●エラー番号：36
音量値が規定外です
原因と処置：音量値の有効範囲は0～16。

●エラー番号：37
@Kコマンドの使用法に誤りがあります
原因と処置：ベンド値の有効範囲は-8192～8191。ディレイ値の有効範囲は0～32767。

●エラー番号：38
キーコードの値が規定外です
原因と処置：キーコードの有効範囲はオクターブ－1のCからオクターブ9のGまで。

●エラー番号：39
音色番号が規定外です
原因と処置：音色番号の有効範囲はFM音源では1～200，MIDIでは1～128，AD PCMのバンクは1～4。

●エラー番号：40
}がありません
原因と処置：連符コマンドの終端記号を書く。

●エラー番号：41
{～}内に音符が多すぎます
原因と処置：音符の数が多すぎるためそれぞれの音符の音長が，計算結果でアンダーフローを起こしていると考えられる。音符の数を減らすか絶対音長指定を用いる。

●エラー番号：42
{～}内に音符がありません
原因と処置：音符を書く。

●エラー番号：43
Q/@Qコマンドの使用法に誤りがあります
原因と処置：Qコマンドのパラメータの有効範囲は1～8。@Qコマンドの有効範囲は1～32768。

●エラー番号：44
{～}の中に規定外のものが存在します
原因と処置：連符内には音長を操作するコマンド(L，@L，Q，@Q，^)を使用できない。音長を操作するコマンドがないか確認してみる。

●エラー番号：45
Yコマンドの使用法に誤りがあります
原因と処置：レジスタ番号，コントロール番号，データが不適当でないか確認してみる。

●エラー番号：46
Jコマンドの使用法に誤りがあります
原因と処置：パラメータのトラック番号の有効範囲は1～80である。

●エラー番号：47
P/@Pコマンドの使用法に誤りがあります
原因と処置：Pコマンドの有効範囲は0～3，@Pコマンドの有効範囲は0～127。

●エラー番号：48
Kコマンドの使用法に誤りがあります
原因と処置：Kコマンドの有効範囲は-768～＋768。

●エラー番号：49
和音コマンドの使用法に誤りがあります
原因と処置：和音コマンドの'～'中に不適当なコマンドが記述されていないか確認してみる。

●エラー番号：50
和音の数が多すぎるか，あるいは後ろの'がありません
原因と処置：和音コマンドでは8和音以上の和音を発音できない。単に終端記号をつけ忘れている場合はそれを記述する。

●エラー番号：51
@Vコマンドの使用法に誤りがあります
原因と処置：@Vコマンドの有効範囲は0～127。

●エラー番号：52
ポルタメントコマンドの使用法に誤りがあります
原因と処置：ポルタメントコマンド(～)中に不適当なコマンドが記述されていないか確認してみる。

●エラー番号:53
ベロシティの値が規定外です
原因と処置:ベロシティの有効範囲は0～127。

●エラー番号:54
N/@Nコマンドの使用法に誤りがあります
原因と処置:チャンネル番号の有効範囲は1～25。ただしPCM8独立チャンネルモード時は1～32。

●エラー番号:55
@Mコマンドの使用法に誤りがあります
原因と処置:振幅値の有効範囲は-32768～32767。また，パラメータが多すぎたりしていないか確認してみる。

●エラー番号:56
Hコマンドの使用法に誤りがあります
原因と処置:2つのパラメータ両方の省略はできない。

●エラー番号:57
@Zコマンドの使用法に誤りがあります
原因と処置:アフタータッチの値は0～127でなければならない。

●エラー番号:58
パラメータの指定方法が規定外です
原因と処置:パラメータの個数や設定有効範囲を確認してみる。

●エラー番号:59
ディレイの値が長すぎます
原因と処置:ポルタメントのディレイの有効範囲は0～32767，和音のディレイの有効範囲は0～255。

●エラー番号:60
AD PCM用のバッファが不足しています
原因と処置:ZMUSIC.Xを一度，常駐解除してAD PCMバッファを大きめに設定して再常駐してみる。

●エラー番号:61
AD PCMの使用はできません
原因と処置:AD PCMバッファを確保しないで常駐させたためAD PCMの使用ができなくなっている。ZMUSIC.Xを一度，常駐解除してからAD PCMバッファを確保して再常駐してみる。

●エラー番号:62
ディスクの読み込みに失敗しました
原因と処置:ファイルが見当たらないか，ファイルが壊れているか，ファイルサイズが異常。読み込み先デバイスの状態をもう一度，確認してみる。

●エラー番号:63
Wコマンドの使用法に誤りがあります
原因と処置:パラメータのトラック番号の有効範囲は1～80である。

●エラー番号:64
ワークエリアが不足しています
原因と処置:ZMUSIC.Xを一度，常駐解除してワークを大き目に設定して再常駐してみる。

●エラー番号:65
正体不明のエラーが発生しました
原因と処置:他のプログラムなどによってZMUSIC.Xが破壊されている可能性があるので必要ファイルをセーブして，すみやかにリセットすべきである。

●エラー番号:66
バージョンナンバーが違うか，あるいはZMUSIC用のデータではありません
原因と処置:X68000システムディスク付属のDUMP.Xなどで演奏データを確認してみる。

●エラー番号:67
AD PCM音色定義コマンドの使用法に誤りがあります
原因と処置:文法が違うかパラメータの与え方がおかしいと考えられる。MEASURE6を参照。

●エラー番号:68
MIDIボードが未装着です
原因と処置:MIDIボードが装着されていないときはMIDI楽器を操作できない。

●エラー番号:69
MIDI楽器に対するパラメータが規定外です
原因と処置:楽器個別コマンドの仕様を確認してみる。

●エラー番号:70
パラメータ値が規定外です
原因と処置:その命令のパラメータの有効範囲をもう一度確認してみる。

●エラー番号:71
ファイルの書き出しに失敗しました
原因と処置:ファイル名が異常か，ディスクが書き込み禁止であったり，フリーエリアが不足している可能性がある。書き込み先のデバイスをもう一度確認してみる。

●エラー番号:72
Xコマンドの使用法に誤りがあります
原因と処置:値は最低1個は書かなければならない。

●エラー番号:73
データは受信されていません
原因と処置:MIDIケーブルが断線しているか，送信側のMIDI楽器の操作を誤っている可能性がある。楽器のマニュアルとMIDIケーブルが正しく接続されているか確認してみる。

●エラー番号:74
波形番号が規定外です
原因と処置:波形番号の有効範囲は0～31。ただし，波形番号0～3はプリセット波形，波形番号8～31はユーザー波形(波形メモリ)。

●エラー番号:75
Mコマンドの使用法に誤りがあります
原因と処置:ピッチモジュレーションのモードは0～2，ARCCモードは0～1。Mコマンドの値を両方省略していないか確認してみる。

●エラー番号:76
波形メモリ登録コマンドの使用法に誤りがあります
原因と処置:ループパターンが0～2になっているか，ループポイントの値が正しいかを確認してみる。

●エラー番号:77
；コマンドの使用法に誤りがあります
原因と処置:パラメータは最低1個は必要。パラメータの有効範囲は

0～255。

●エラー番号:78
¥コマンドの使用法に誤りがあります
原因と処置:スピードパラメータの絶対値は85以下。

●エラー番号:79
?コマンドの使用法に誤りがあります
原因と処置:パラメータの有効範囲は0～255。

●エラー番号:80
@Fコマンドの使用法に誤りがあります
原因と処置:周波数パラメータの有効範囲は0～6。

●エラー番号:81
@Gコマンドの使用法に誤りがあります
原因と処置:パラメータの有効範囲は0～127。

●エラー番号:82
@Yコマンドの使用法に誤りがあります
原因と処置:パラメータの個数は3つ，または4つ。各パラメータの有効範囲は0～127。

●エラー番号:83
@Eコマンドの使用法に誤りがあります
原因と処置:パラメータは最低1個は必要。パラメータの有効範囲は0～127。

エラーファイルの活用

コンパイル時のエラーメッセージをうまく活用すると開発効率が上がります。たとえば，以下のようにして'MUSIC.ZMS'というZMSファイルをコンパイルした場合，

```
A>zmusic -c MUSIC > ER
```

エラーが発生すると'ER'というファイルネームのファイルができます(これをエラーファイルと呼びます)。さて，ここで，

```
A>ed ER
```

としてエディタでエラーファイルを読み込んでください。エラーファイルの読み込みを確認したら[ESC]を押してから[V]を押してください。するとMUSIC.ZMSファイルを読み込み，'MUSIC.ZMS'のエラーの発生した行へとカーソルがジャンプします(これはエディタED.Xのタグジャンプという機能です)。あとはエディタで誤った箇所を訂正するだけです。エラーが複数あった場合は[SHIFT]+[F6]で，もう一度エラーファイルの画面にして次のエラーメッセージの行にカーソルを動かして再び[ESC]+[V]を実行してみましょう。すると'MUSIC.ZMS'ファイルのそのエラーのある行へジャンプしているはずです。こうしてエラーを修正していきます。エラーがまったくない場合はエラーファイルは「空」になります。

ちなみにエラーファイルは，

```
MUSIC.ZMS     18     SYNTAX ERROR
MUSIC.ZMS     35     M_TRK ERROR
              :
```

のようにファイル名，行番号，エラー内容の順番になっています(タグファイルフォーマット)。

11.2. ZMSファイルコンパイル時のエラー

ZMSファイルを，

```
A>zmusic -c ファイルネーム
```

とするとZMSをコンパイルしたZMDを得ることができますが，このときZMSに誤りがあれば当然エラーを発生します。このときのエラーメッセージは英文ですがエラーメッセージの末尾についた番号はMUSICZ.FNCのエラーと対応したものになっています。たとえば，

```
MUSIC.OPM     22     ILLEGAL CHANNEL NUMBER AT @N (No.54)
```

と出た場合，MUSICZ.FNCのエラーのエラー番号54と同内容のエラーが発生していることを表します。

つまりZMSコンパイル時に発生するエラーの多くはMUSICZ.FNCのエラーと1対1に対応するものなので説明は省略し，ここではZMSのコンパイル時に発生するエラーでMUSICZ.FNCのエラーと対応しないもののみを示します。

●エラー番号:84
SYNTAX ERROR.
原因と処置:あるZMSコマンドのパラメータを最後まで書かないで，次のZMSコマンドを実行しようとしていたり，ファイルが終わっていたりして正常にZMSコマンドを実行できない。

●エラー番号:85
(A) COMMAND ERROR.
原因と処置:チャンネルアサインのZMSコマンドの使用法に誤りがある。チャンネル番号の有効範囲は1～25。ただしPCM8独立チャンネルモード時は1～32。また，トラック番号の有効範囲は1～80。

●エラー番号:86
(M) COMMAND ERROR.
原因と処置:トラックバッファ確保のZMSコマンドの使用法に誤りがある。トラック番号の有効範囲は1～80で，バッファサイズの有効範囲は0～65535。

●エラー番号:87
(T) COMMAND ERROR.
原因と処置:MML書き込みコマンドのZMS命令の使用法に誤りがある。

●エラー番号:88
FM VOICE SET COMMAND ERROR.
原因と処置:FM音源の音色設定のZMSコマンドの使用法に誤りがある。音色のパラメータの書式や有効範囲はMEASURE4を参照。

●エラー番号:89
ADPCM CONFIGURATION COMMAND ERROR.
原因と処置:AD PCMの登録ZMSコマンドの使用法に誤りがある。書式やパラメータの意味や有効範囲はMEASURE6を参照。

●エラー番号:90
ILLEGAL PARAMETER FORMAT IN COMMON COMMAND.
原因と処置:ZMSの共通コマンドにおいて不当にパラメータなどを省略している。

●エラー番号:64
WORK AREA IS TOO SMALL.
原因と処置:ワークエリアが不足している。一度，ZMUSIC.Xを常駐解除してワークエリアを大きめに設定して再常駐させる。

●エラー番号:91
(O) COMMAND ERROR.
原因と処置:テンポ設定ZMSコマンドの使用法に誤りがある。設定できるテンポはタイマAモードのときは77～300，タイマBモードのときは20～300。

●エラー番号:92
UNDEFINED COMMAND ERROR.
原因と処置:未定義のZMS共通コマンドを使用しようとした。

●エラー番号:93

PARAMETER OUT OF RANGE.

原因と処置:有効範囲を超えた値をZMSの共通コマンドのパラメータとして与えている。

ビープ音について

COMMAND.Xから,

A>COPY　ファイルネーム　OPM

として演奏させた場合に，そのファイル中にエラーがあった場合はエラーメッセージなどは出力せずビープ音を発声します(ビープ音の登録がなされていないシステムではもちろん鳴りませんが)。これは手抜きではなく，なにか別のアプリケーションから同様の操作をされたときにエラーメッセージなどで画面などを壊さないためにこうなりました。ビープ音が鳴ったときはその演奏データを一度コンパイルしたりしてエラーの内容を確認すべきです。

また,MIDIデータの取り込み中にビープ音が鳴った場合はデータの取り込み用バッファ，具体的にはトラックバッファ，ワークエリアが不足しています。一度ZMUSIC.Xを解除しそれらを大き目にとって再常駐させてください。

11.3. ZPCNV.Rのエラー

File open error. Does the FILENAME1 surely exist in your device?

原因と処置:AD PCMコンフィギュレーションファイルが読み込めない。記憶デバイスにファイルがあるか確認してみる。

File read error. Check FILENAME1.

原因と処置:AD PCMコンフィギュレーションファイルの読み込みに失敗した。ファイルが破壊されていないか確認してみる。

???? is unable to read.

原因と処置:AD PCMデータの読み込みに失敗した。そのファイルが本当に存在しているかどうか確認してみる。また，環境変数'zmusic'のパスが正しく設定されているか確認してみる。

Illegal File length.

原因と処置:読み込もうとしたAD PCMデータのファイルサイズが異常。大きすぎるか，ファイルサイズが0の可能性がある。

File open error. Check FILENAME2.

原因と処置:書き込み用ファイルがオープンできない。異常なファイルネームを指定しているか，記憶デバイスが書き込み禁止になってる可能性がある。

File write error. Check FILENAME2.

原因と処置:ZPDの書き込みに失敗した。記憶デバイスの容量が十分かどうか確認してみる。また，ファイルネームが異常でないか，書き込み禁止のファイルに対して上書きしようとしていないかなどを確認してみる。

Out of memory.

原因と処置:作業メモリが不足している。メモリを増設するか，必要のないプログラムの常駐を解除しフリーエリアを確保する。

Syntax error.

原因と処置:AD PCMコンフィギュレーションの文法に誤りがある。文法や用法の詳しい解説はMEASURE6を参照。

Refered to empty note number.

原因と処置:参照しようとしたノート番号にはAD PCMが登録されていない。AD PCMコンフィギュレーションファイルの内容をもう一度確認してみる。

Illegal parameter format.

原因と処置:数値を不当に省略している。

Illegal note number.

原因と処置:ノート番号の有効範囲は0～511。

Illegal pitch shift parameter.

原因と処置:音程変更の有効範囲は-12(1オクターブ下)～+12(1オクターブ上)まで。

Illegal volume value.

原因と処置:音量変更の有効範囲は1%～300%まで。ただし原音量を100とする。

Illegal delay value.

原因と処置:ミキシングディレイの有効範囲は0～65535。

Cut size is too big.

原因と処置:AD PCMの切り出しサイズの有効範囲は0～65535。

Illegal offset value.

原因と処置:フェードイン/アウトオフセット，切り出しオフセットの有効範囲は0～65535。

Illegal output level.

原因と処置:フェードイン/アウトレベルの有効範囲は0～127。

Illegal Bank number.

原因と処置:AD PCMのバンクの有効範囲は1～4。

11.4. ZP.Rのエラー

ZP.RがZMUSIC.Xを制御してその際に発生したエラーは,

ERROR n

として表示されます。nはエラーコードでMUSICZ.FNCのエラーのエラーコードと同一です。ここではZP.R特有のエラーについてだけ解説します。

V-DISP interrupt has already used by other applications.

原因と処置:ZP.Rのジュークボックス機能やデバッグツールではV-DISP割り込みを使用する。なにか別のプログラムがこの割り込みをすでに使用しているため，ZP.Rの常駐が行えなかった。

ZMUSIC.X is not included.

原因と処置:ZMUSIC.Xが組み込まれていないため,ZP.Rは使用できない。

Illegal file size.

原因と処置:読み込もうとしたファイルサイズが0か大きすぎる。

Out of memory.

原因と処置:作業メモリが不足している。メモリを増設するか，必要のないプログラムの常駐を解除しフリーエリアを確保する。

Read error.

原因と処置:データの読み込みに失敗した。ファイルが破壊されていないか確認してみる。

File not found.

原因と処置:ファイルが見つからなかった。記憶デバイスにそのファイルが本当に存在しているか,環境変数'zmusic'が正しく設定されて

いるか確認してみる。

Too many filenames are written in index-file.
原因と処置:ジュークボックスのインデックスファイルに64曲以上の曲名を書いている。

ZP.R is not kept in your system.
原因と処置:ZP.Rが常駐していないのに常駐解除を実行した。

ZP.R has already kept.
原因と処置:ZP.Rはすでに常駐しているのに，さらに常駐しようとした。

Fail in making of string MIDI data.
原因と処置:MIDIデータの文字型データ変換に失敗した。ZMUSIC.Xを一度常駐解除し，トラックバッファやワークを増やして再常駐して，一連の作業をもう一度やり直してみる。

Write error.
原因と処置:ファイルの書き出しに失敗した。記憶デバイスの容量が不足していないか確認してみる。また，ファイルネームが異常でないか，書き込み禁止のファイルに対して上書きしようとしていないかなどを確認してみる。

Unidentified file.
原因と処置:Z-MUSICシステムとは無関係のファイルをアクセスしようとした。

Illegal command has existed.
原因と処置:AD PCM定義コマンド(n ファイルネーム,Pp,Vv,Mm,d,Cc,s,R,Ff,l)やAD PCMコンフィギュレーション実行コマンド(.adpcm_list)を用いたZMDはジュークボックス機能で演奏することはできない。ZPDを用いて演奏するように改造する必要がある。

ADPCM buffer is too small.
原因と処置:AD PCMバッファが小さすぎるためZPDを登録することができなかった。ZMUSIC.Xを一度常駐解除し，AD PCMバッファを大き目に設定して再常駐させてもう一度試してみる。

Track buffer is too small.
原因と処置:トラックバッファが小さすぎるため演奏データを演奏することができなかった。ZMUSIC.Xを一度常駐解除し，トラックバッファを大き目に設定して再常駐させて，もう一度試してみる。

Version number mismatch.
原因と処置:バージョンの違うZMDを演奏しようとした。

MIDI is Unable to use.
原因と処置:MIDIボードがないのにMIDIの曲を演奏しようとした。

An error is still remained in ZMD/ZMS.
原因と処置:演奏データにエラーが残っているためにステップタイムの計算(/Q)が行えなかった。

Illegal parameter.
原因と処置:既定外のパラメータを設定した。パラメータを使用範囲に訂正する。

11.5. ZMUSIC.Xのエラー

MUSIC BIOS has already included in your system.
原因と処置:ZMUSIC.Xあるいはその他の音楽ドライバがすでに常駐しているため常駐できなかった。または，割り込みベクタなどが不当に書き換えられている可能性がある。

Didn't you include special ADPCM driver?
原因と処置:PCM8.X以外のなにか特殊なAD PCMドライバが組み込まれているためZMUSIC.Xは常駐できなかった。

Out of memory.
原因と処置:作業メモリが不足している。メモリを増設するか，必要のないプログラムの常駐を解除しフリーエリアを確保する。

ZmuSiC is not kept in your system.
原因と処置:ZMUSIC.Xが常駐していないのに常駐解除しようとした。

ZmuSiC is unable to release.
原因と処置:ZMUSIC.Xが不当に書き換えられているため常駐解除できなかった。また，特殊な環境から常駐をした場合，常駐解除できないことがある。このときは一度その環境に戻ってから常駐解除を行わなくてはならない。

Illegal version number.Unable to release.
原因と処置:バージョンが違うために解除できなかった。

Unknown error! Please reset your system.
原因と処置:正体不明のエラーが発生した。プログラムが不当に書き換えられている可能性がある。

File not found.
原因と処置:コンパイルしようとしたZMSファイルが見つからない。そのファイルが実在するかどうか，環境変数'zmusic'が正しく設定されているかどうかを確認してみる。

Write error.
原因と処置:コンパイルされて生成されたZMDの書き込みに失敗した。異常なファイルネームを設定していないか，記憶デバイスの容量は十分か書き込み禁止のファイルに対して上書きしようとしていないかを確認してみる。

Unable to read.
原因と処置:コンパイルしようとしたZMSの読み込みに失敗した。ファイルサイズが0でないかどうか，ファイルが破壊されていないかどうかを確認してみる。

Work area is not enough.
原因と処置:ZMSのコンパイルがワークエリア不足で中断された。ワークを大き目に確保してもう一度コンパイルしてみる。

Track buffer is not enough.
原因と処置:ZMSのコンパイルがトラックバッファ不足で中断された。トラックバッファを大き目に設定してもう一度コンパイルしてみる。

Block ADPCM data '~' couldn't be found.
原因と処置:指定されたZPDファイルが見つからない。そのZPDファイルが実在するか，環境変数'zmusic'が正しく設定されているかどうかを確認してみる。

Start up file '~' couldn't be found.
原因と処置:指定されたスタートアップファイルが見つからない。そのスタートアップファイルが実在するか，環境変数'zmusic'が正しく設定されているかどうかを確認してみる。

MEASURE12
ZMD/ZPDフォーマット

ここではZMDフォーマット,ZPDのフォーマットについて解説します。一般的な使用をする限りは読み飛ばして構いません。

12.1. 解説を読むにあたって

文中の(.b)(.w)(.l)などはデータ長を表している。

(.b) バイト
(.w) ワード(2バイト)
(.l) ロングワード(4バイト)

12.2. ZMDフォーマット

ZMDデータの内部構成は以下のようになっている。

```
$10(.b)                 ダミーコード
"ZmuSiC"                ZMDファイルID
VERSION NUMBER(.b)      現在のバージョンナンバー
共通コマンド群…          FM音源音色設定やAD PCM設定など
$FF(.b)                 共通コマンド終了コード
 (次の番地が奇数の場合はもう1個$FF(.b)がくる)
総トラック数(.w)
トラック1のデータアドレスまでのオフセットアドレス(.l)
$00(.b),トラック1の絶対チャンネル番号(.b)
トラック2のデータアドレスまでのオフセットアドレス(.l)
$00(.b),トラック2の絶対チャンネル番号(.b)
        :
総トラック数分ある
        :
トラックnのデータアドレスまでのオフセットアドレス(.l)
$00(.b),絶対チャンネル番号(.b)
各トラックの演奏データ....      総トラック数分ある
```

12.3. 共通コマンドのZMDコード

●FM音源音色設定 (ZMS (V)コマンド)

$04(.b),音色番号(.b),音色データ(55バイト) 計57バイト
音色番号=1～200

音色のデータフォーマットはMEASURE10 ファンクション$04 mvset参照

●テンポ (ZMS (O)コマンド)

$05(.b),テンポ値(.w) 計3バイト
テンポ値=20～300

●ベースチャンネル設定 (ZMS (B)コマンド)

$15(.b),モード値(.b) 計2バイト
モード値=0:FM音源チャンネル基準モード
モード値≠0:MIDIチャンネル基準モード

チャンネルモードについてはMEASURE3 m_ch(), MEASURE4 (Bn)コマンドを参照

●MIDIデータ転送 (ZMS (X)コマンド, .MIDI_DATAコマンド)

$18(.b),データ数(.w),データ(.b),…… 合計は不定

ローランドエクスクルーシブコマンドやMIDI楽器機種別コマンドはこのコードに展開される

●FM音源音色設定 (ZMS (@)コマンド, .FMVSETコマンド)

$1B(.b),音色番号(.b),音色データ(55バイト) 計57バイト
音色番号=1～200

音色のデータフォーマットはMEASURE10 ファンクション$04 mvset2参照

●AD PCMコンフィギュレーション

```
$40(.b),note number(.w)
        ,pitch parameter(.w),volume parameter(.w)
        ,mix delay parameter(.w),mix note number(.w)
        ,cut offset(.w),cut size(.w)
        ,reverse switch(.b)
        ,fade in/out offset(.w),fade in or out(.b)
        ,fade in/out level(.b)
        ,filename....,0(.b)
         合計不定
```

あるいは,

```
$40(.b),note number(.w)
        ,pitch parameter(.w), volume parameter(.w)
        ,mix delay parameter(.w), mix note number(.w)
        ,cut offset(.w), cut size(.w)
        ,reverse switch(.b)
        ,fade in/out offset(.w), fade in or out(.b)
        ,fade in/out level(.b)
        ,0(.w),操作対象ノート番号(.w)
          計24バイト
      note number(.w):0～511
      fade in or out(.b):(-1:fade in,+1:fade out)
```

その他のパラメータの範囲についてはMEASURE10ファンクション$10 (adpcm_read) 参照

●全音符クロックのセット(ZMS(Z)コマンド)

$42(.b),master clock(.b),tempo base counter(.l) 計6バイト
master clock:(Zn)のnの値
tempo base counter=(16×4000×60000)/(master clock×256)

●波形メモリデータの登録(ZMS .WAVE_FORMコマンド)

$4a(.b),number of data(.w),登録波形番号(.b)
,loop type(.b),loop point(.w),data(.w),…
合計は不定

●AD PCMコンフィギュレーションファイルの読み込み
(ZMS .ADPCM_LISTコマンド)

$60(.b),FILENAME文字列…,$00(.b) 合計は不定

●文字列表示(ZMS .PRINTコマンド)

$61(.b),文字列…,$00(.b) 合計は不定

●MIDIダンプデータ(MDD)の読み込みと送信(ZMS .MIDI_DUMPコマンド)

$62(.b),FILENAME文字列…,$00(.b) 合計は不定

●ZPDデータの受け渡し(ZMS .ADPCM_BLOCK_DATAコマンド)

$63(.b),FILENAME文字列…,$00(.b) 合計は不定

●ダミーZMD

$7e(.b) 計1バイト
なんの機能も果たさないZMDコード 機械語のNOPに相当

●コメント (ZMS .COMMENTコマンド)

$7F(.b),文字列…,$00(.b) 合計は不定

12.4. MML用ZMDコード

■$00～$7F

●音階 (MML A～G)

ノート番号(.b),step time(.b),gate time(.b) 計3バイト
ノート番号:0～127
step time :1～254
gate time :1～255(255はタイ/スラーとして扱われる)

■$80～$8F

●休符 (MML R)

$80(.b),step time(.b),gate time(.b) 計3バイト
step time :1～254
gate time :1～255(255はタイ/スラーとして扱われる)

●ノイズモードオフ (MML Y15,0, @O)

$82(.b) 計1バイト

●同期待ちコマンド (MML W)

$83(.b) 計1バイト

●臨時ベロシティ

$84(.B) 計1バイト

■$90～9F

●テンポ (MML @T)

$90(.b),タイマ値(.w) 計3バイト
タイマ値の有効範囲は常駐時に指定したタイマの種類によって変動する

タイマA：0～1023
タイマB：0～255

●テンポ （MML T）

$91(.b),テンポ値(.w)　計3バイト

テンポ値の有効範囲は常駐時に指定したタイマの種類によって変動する

タイマA：77～300
タイマB：20～300

●相対タイマUP （MML @T+）

$92(.b),相対タイマ値(.w)　計3バイト

相対タイマ値(タイマAモード)：1～1023
相対タイマ値(タイマBモード)：1～255

●相対タイマDOWN （MML @T−）

$93(.b),相対タイマ値(.w)　計3バイト

相対タイマ値(タイマAモード)：1～1023
相対タイマ値(タイマBモード)：1～255

●相対テンポUP （MML T+）

$94(.b),相対テンポ値(.w)　計3バイト

相対テンポ値(タイマAモード)：1～230
相対テンポ値(タイマBモード)：1～280

●相対テンポDOWN （MML T−）

$95(.b),相対テンポ値(.w)　計3バイト

相対テンポ値(タイマAモード)：1～230
相対テンポ値(タイマBモード)：1～280

●相対ピッチベンドUP （ZMD専用命令）

$96(.b),相対ピッチベンド値(.w)　計3バイト

相対ピッチベンド値：1～16383

●相対ピッチベンドDOWN （ZMD専用命令）

$97(.b),相対ピッチベンド値(.w)　計3バイト

相対ピッチベンド値：1～16383

●モジュレーション波形選択 （MML S）

$98(.b),ピッチモジュレーション/拡張ピッチモジュレーションの波形番号(.b),アンプリチュードモジュレーション/拡張ARCCの波形番号(.b)　計3バイト

波形番号 0：鋸歯波 1：矩形波 2：三角波 3：鋸歯波シングル

各パラメータとも−1を指定した場合は省略を意味する

●MIDIのピッチモジュレーションモードの設定 （MML M）

$99(.b),ピッチモジュレーションモード(.b),ARCCモード(.b)　計3バイト

ピッチモジュレーションモード：
0＝通常モード/1＝拡張モード#1/2＝拡張モード#2
ARCC：0＝通常モード/1＝拡張モード

各パラメータとも−1を指定した場合は省略を意味する
拡張モードについてはMEASURE5 Mコマンドの項参照

●ARCC用コントロールチェンジ番号登録 （MML @C）

$9A(.b),CONTROL CHANGE number(.b)
,ARCC reset value(.b)
,ARCC neutral value(.b)　計4バイト

CONTROL CHANGE number：0～127
ARCC reset value：0～127
ARCC beutral value：0～127

各パラメータとも−1を指定した場合は省略を意味する

●AD PCM再生 （MML Y2,）

$9B(.b),note number(.w)　計3バイト

note number：0～511

●モジュレーション非同期/同期設定 （MML H）

$9c(.b),ピッチモジュレーション同期スイッチ(.b),アンプリチュードモジュレーション/拡張ARCC同期スイッチ(.b)　計3バイト

同期スイッチ＝−1：省略
＝ 0：同期オン
＝−1：同期オフ(非同期)

■ $A0～AF

●音色切り替え （MML @）

$A0(.b),音色番号(.b)　計2バイト

音色番号：1～200

●音色切り替え （ハードLFOをいじらない音色設定）

$A1(.B),音色番号(.B)　計2バイト

音色番号：1～200

●AD PCM再生周波数変更 （MML Y13,）

$A2(.b),周波数値(.b)　計2バイト

周波数値：0＝3.9kHz
1＝5.2kHz
2＝7.8kHz
3＝10.4kHz
4＝15.6kHz
5＝16bit PCMデータ方式
(PCM8独立チャンネルモード時)
6＝8bit PCMデータ方式
(PCM8独立チャンネルモード時)

●チャンネルアサイン変更 （MML N,@N）

$A3(.b),絶対チャンネル番号(.b)　計2バイト

絶対チャンネル番号＝
0～7　内蔵FM音源チャンネル 1 ～ 8
8　内蔵AD PCM音源
9～24　MIDIチャンネル 1 ～16
25～31　AD PCMチャンネル 2 ～ 8
(PCM8独立チャンネルモード時)

●ノイズ （MML Y15,）

$A5(.b),ノイズ周波数(.b)　計2バイト

ノイズ周波数：128～159(128＋0～128＋31)

●フェードイン/アウト （MML ¥）

$A6(.b),フェードイン/アウトスピード(.b)　計2バイト

フェードイン/アウトスピード
−85～−1：フェードイン
0：フェードアウトモード解除
1～85：フェードアウト

パラメータの機能と意味についての詳しい解説はMEASURE10 ファンクション$1a fade_outを参照

●ダンパーペダル （MML @D）

$A7(.b),ダンパー値(.b)　計2バイト

ダンパー値：0～127

●ベンドレンジ変更 （MML @G）

$A8(.b),ベンド幅(.b)　計2バイト

ベンド幅：0～127

●AD PCM周波数変更 （MML @F）

$A9(.b),周波数値(.b)　計2バイト

周波数：0＝3.9kHz
1＝5.2kHz
2＝7.8kHz
3＝10.4kHz
4＝15.6kHz
5＝16bit PCMデータ方式
(PCM8独立チャンネルモード時)
6＝8bit PCMデータ方式
(PCM8独立チャンネルモード時)

●相対ボリュームUP （MML ^）

$AA(.b),相対ボリューム値(.b)　計2バイト

相対ボリューム値：1～127

●相対ボリュームDOWN （MML _）

$AB(.b),相対ボリューム値(.b)　計2バイト

相対ボリューム値：1～127

●キーオフなしモード （MML @R）

$AC(.b),mode(.b)　計2バイト

mode：0＝通常モード
1＝キーオフなしモード

●絶対音長 0 による強制発音 （MML ＊0）

$AD(.b),note number(.b)　計2バイト

note number：0～127

●アンプリチュードモジュレーション/ARCC/拡張ARCC （MML @A）
$AE(.b),振幅値(.b)　計2バイト
振幅値：-128～127
●同期信号送信コマンド （MML W）
$AF(.b),トラック番号(.b)　計2バイト
トラック番号：0～79

■$B0～$BF

●パンポット0 （MML P0）
$B0(.b)　計1バイト
●パンポット1 （MML P1）
$B1(.b)　計1バイト
●パンポット2 （MML P2）
$B2(.b)　計1バイト
●パンポット3 （MML P3）
$B3(.b)　計1バイト
●多段階パンポット （MML @P）
$B4(.b),パンポット値(.b)　計2バイト
パンポット値：0～127
●Yコマンド （MML Y）
$B5(.b),レジスタ番号(.b),データ値(.b)　計3バイト
レジスタ番号:0～255
データ番号:0～255
●ボリューム （MML V,@V）
$B6(.b),ボリューム値(.b)　計2バイト
ボリューム値:0～127
ただしは値は(127－実際の設定ボリューム)に相当する
●AD PCM PAN （Y3,）
$B7(.b)パンポット値(.b)　計2バイト
パンポット値＝0:消音
1:左
2:右
3:中央
●AD PCM効果音モード設定 （Y14,）
$B8(.b),モード値(.b)　計2バイト
モード値＝0:通常
＝1:効果音モード再生指定
●ベロシティ （MML U,@U）
$B9(.b),ベロシティ値(.b)　計2バイト
ベロシティ値：0～127
●ピッチモジュレーションスイッチ （MML @M）
$BB(.b),スイッチ値(.b)　計2バイト
スイッチ値＝0:OFF
＝1:ON
●アンプリチュードモジュレーション/ARCCスイッチ （MML @A）
$BC(.b),スイッチ値(.b)　計2バイト
スイッチ値＝0:OFF
＝1:ON
●オートベンドスイッチ （MML @B）
$BD(.b),スイッチ値(.b)　計2バイト
スイッチ値＝0:OFF
＝1:ON
●アフタータッチシーケンススイッチ （MML @Z）
$BE(.b),スイッチ値(.b)　計2バイト
スイッチ値＝0:OFF
＝1:ON
●強制キーオフ （MML）
$BF(.b)　計1バイト

■$C0～$CF

●［～］コマンド系 （MML［～］）
$C0(.b),コマンド番号(.b)　計2バイト

コマンド番号		
	＝0～2	RESERVED
	＝3	[D.C.]
	＝4	[SEGNO][$]
	＝5	[D.S.]
	＝6	[CODA]
	＝7	[TOCODA][＊]
	＝8	[FINE][^]
	＝9	[DO]
	＝10	[LOOP]
	＝11	[!]
	＝12	[@]

●リピートコマンド開始 （MML |:）
$C1(.b),$CF(.b),repeat counts(.b)　計3バイト
repeat counts:1～255
●リピートコマンド終端 （MML :|）
$C2(.b),オフセット(.w)　計3バイト
オフセット値は必ず正。次のZMDを指し示しているポインタからこのオフセットを減算することになる。減算の結果，ポインタはリピート開始コマンドZMDの2バイト目の$CFを指している。
●リピートコマンドスキップ##1 （MML |）
$C3(.b),回数指定値(.b),抜け出しオフセット値(.w)
計4バイト
回数指定値:1～255
抜け出しオフセット値は必ず正
●リピートコマンドスキップ##2 （MML |）
$C4(.b),抜け出しオフセット値(.w)　計3バイト
抜け出しオフセット値は必ず正
●MIDIのタイ/スラーモード指定 （MML "）
$C5(.b),モード値(.b)　計2バイト
モード値＝0:従来モード
＝1:FM音源互換モード
●特殊コマンド制御スイッチ （MML =）
$C7(.b),スイッチ値(.b)　計2バイト
ビットオン(＝1)でスイッチオン
オフ(＝0)でスイッチオフ
第0ビット：ピッチモジュレーション
第1ビット：ARCC/AMOD
第2ビット：オートベンド
第3ビット：アフタータッチシーケンス
●相対パンポットUP （MML @P＋）
$C8(.b),相対パンポット値(.b)　計2バイト
相対パンポット値:1～127
●相対パンポットDOWN （MML @P－）
$C9(.b),相対パンポット値(.b)　計2バイト
相対パンポット値:1～127
●相対ベロシティUP （MML U＋）
$CA(.b),相対ベロシティ値(.b)　計2バイト
相対ベロシティ値：1～127
●相対ベロシティDOWN （MML U－）
$CB(.b),相対ベロシティ値(.b)　計2バイト
相対ベロシティ値：1～127
●全トラックフェードイン/アウト （ZMD専用命令）
$CC(.b),フェードイン/アウトスピード(.b)　計2バイト
フェードイン/アウトスピード
－85～－1:フェードイン
0:フェードアウトモード解除
1～85:フェードアウト
パラメータの機能と意味についての詳しい解説はMEASURE10ファンクション$1a fade_outを参照
●絶対音長0による和音発音 （MML ＊0&）
$CD(.b),note number(.b)　計2バイト
note number:0～127
●強制再演奏コマンド （MML J）
$CE(.b),トラック番号(.b)　計2バイト
トラック番号：0～79
未使用のトラックに対して本命令を実行した場合の動作は保証されない
●リピートコマンドのZMDコードの2バイト目

$C1(.b),$cf(.b),repeat counts(.b)
↑
この部分

■$D0～$DF

●ウェイト （MML @W）

$D0(.b),step time(.b),$00(.b) 計3バイト

step time:1～254

●キートランスポーズ/デチューン （MML @K,@B）

$D1(.b),トランスポーズ値#1(.w),トランスポーズ#2値(.w) 計5バイト

トランスポーズ値#1は半音＝64として算出された値(－768～＋768),トランスポーズ値#2は半音≒683として算出された値(－8192～＋8191)

●NRPN設定 （MML @Y）

$D2(.b),アドレス(.w),データ(.w) 計5バイト

アドレス：$00,$00～$7F,$7F

データ：$00,$00～$7F,$7F

●バリエーション/バンク変更（MML I）

$D3(.b),バリエーション番号(.w) 計3バイト

バリエーション番号：$00,$00～$7F,$7F

●ワーク直接書き換えコマンド （MML ?）

$D5(.b),offset(.b),data(.b) 計3バイト

offset:0～255

data:0～255

●モジュレーションスピード設定コマンド （MML @S）

$D6(.b),ピッチモジュレーション/拡張ピッチモジュレーションのスピード(.w),アンプリチュードモジュレーション/拡張ARCCのスピード(.w) 計5バイト

PM/拡張PMのスピード:1～32767

AM/拡張ARCCのスピード:1～32767

※各パラメータとも0を指定した場合は省略を意味する

●ワーク直接書き換えコマンド相対アップ （MML ?a,+n）

$D7(.B),offset(.B),data(.B) 計3バイト

offset:0～255

data:1～255

●ワーク直接書き換えコマンド相対ダウン （MML ?a,-n）

$D8(.B),offset(.B),data(.B) 計3バイト

offset:0～255

data:1～255

●臨時ベロシティ

$D9(.B),ベロシティ値(.B) 計2バイト

ベロシティ値：0～127

●臨時相対ベロシティアップ

$DA(.B),相対ベロシティ値(.B) 計2バイト

相対ベロシティ値：1～127

●臨時相対ベロシティダウン

$DB(.B),相対ベロシティ値(.B) 計2バイト

相対ベロシティ値：1～127

■$E0～$EF

●ポルタメント （MML (～)）

$E0(.b),ノート番号(.b),step time(.w),gate time(.w),delay(.w),増分(.w),補正パラメータ(.b),符号(.b) 計12バイト

step time:1～32767

gate time:1～32767(32768以上はタイ/スラーとして扱われる)

delay:0～32767

増分＝int(ベンド幅÷step time)

補正パラメータ＝

round(256×(ベンド幅 MOD step time)/step time)

増分，補正パラメータはBRESENHAMのアルゴリズムによる計算結果。ベンド幅はFM音源トラックでは半音＝64，MIDIトラックでは半音≒683とする

符号はポルタメント方向で，－1がピッチダウン，＋1がピッチアップに対応する

●オートベンド （MML @B,@K）

$E1(.b),開始値(.w),目的値(.w),開始値(.w),目的値(.w),ディレイ値(.w),符号(.b) 計12バイト

前者の開始値/目的値は半音＝64として算出された値

後者の開始値/目的値は半音≒683として算出された値

また，目的値は相対値で表される

ディレイ:0～32767

符号はベンド方向で，－1がピッチダウン，＋1がピッチアップに対応する

●和音 （MML '～'）

$E2(.b),step time(.w),gate time(.w),ディレイ値(.b),ノート番号1(.b)～8 計14バイト

step time:1～32767

gate time:1～32767(32768以上はタイ/スラーとして扱われる)

delay:0～255

ノート番号:0～127

●アフタータッチシーケンス （MML @Z）

$E3(.b),アフタータッチ値1(.b)～8 計9バイト

アフタータッチ値:0～127

:－1はノータッチ期間

●モジュレーションデプス （MML @M）

$E6(.b),振幅(.w) 計3バイト

振幅:-32768～＋32768(FM音源トラック)

: 0～127(MIDIトラック通常モード)

: －8192～＋8191(MIDIトラック)拡張モード

●ディレイセット （MML @H）

$E8(.b),ピッチモジュレーション用ディレイ(.w),アンプリチュードモジュレーション/ARCC用ディレイ(.w) 計5バイト

PM用ディレイ:0～65534

AM/ARCC用ディレイ:0～65534

※各パラメータとも－1を指定したときは省略を意味する

●ローランドエクスクルーシブ （MML X）

$EA(.b),データ(.b)…,チェックサム(.b),$FF(.b) 合計は不定

データ:0～127

チェックサム:0～127

●IDセット （MML @I）

$EB(.b),MAKER(.b),DEVICE(.b),MODEL(.b) 計4バイト

MAKER:0～127

DEVICE:0～127

MODEL:0～127

●MIDIデータ送信 （MML @X）

$EC(.b),データ数(.w),データ(.b)… 合計は不定

データ数:1～65535

データ:0～255

●エフェクトコントロール （MML @E）

$ED(.b),パラメータ#1(.b),～#3(.b) 計4バイト

パラメータ#1(REVERB LEVEL):0～127

パラメータ#2(CHORUS LEVEL):0～127

パラメータ#3:未使用

MT-32/CM-32L/CM-64(LAパート)のケース

パラメータ#1(パート番号):1～8

パラメータ#2(REVERB switch):0 or 1

パラメータ#3:未使用

※各パラメータとも－1を指定したときは省略を意味する

●ピッチモジュレーションデプス1/8モード （MML @M）

$EE(.b),省略ビットパターン(.b),振幅#1(.w),…,振幅#8(.w)×8 計18バイト

省略ビットパターン

ビット＝0 省略(これに対応するパラメータは意味をなさず)

ビット＝1 設定

振幅#1～#8:－32768～＋32768(FM音源トラック)

0～127(MIDIトラック通常モード)

－8192～＋8191(MIDIトラック)拡張モード

省略ビットパターンとはそのビットに対応したパラメータが有効か無効(省略されている)かを表したものでビット0～7がそれぞれパ

ラメータの１～８番目に対応する

●アンプリチュードモジュレーション/ARCCデプス1/8モード （MML @A）

$EF(.b),省略ビットパターン(.b),振幅#1(.b),…,振幅#8(.b)×8
計10バイト

省略ビットパターン
ビット＝0　省略(これに対応するパラメータは意味をなさず)
ビット＝1　設定
振幅#1～#8:0～127

省略ビットパターンとはそのビットに対応したパラメータが有効か無効(省略されている)かを表したものでビット0～7がそれぞれパラメータの１～８番目に対応する

■$F0～$FF

●NOP （ZMD専用命令）

$F0(.b)　計１バイト

なにも処理を行わない

●スキップコマンド１ （ZMD専用命令）

$F1(.b),offset(s)(.w)　計３バイト

offset:0～65535

オフセットバイトだけ先へスキップ

●スキップコマンド２ （ZMD専用命令）

$F2(.b),offset(s)(.w)　計３バイト

offset:0～65535

オフセットバイトだけ前に戻る

●ノートオフ （MIDIトラック専用ZMD専用命令）

$FC(.b),ノート番号(.b),ベロシティ値(.b)　計３バイト

ノート番号:0～127
ベロシティ値:0～127

●ノートオン （MIDIトラック専用ZMD専用命令）

$FD(.b),ノート番号(.b),ベロシティ値(.b)　計３バイト

ノート番号:0～127
ベロシティ値:0～127

●絶対音長指定音階(MML ＊によって絶対音長指定された音階/休符)

$FE(.b),ノート番号(.b),step time(.w),gate time(.w)
計６バイト

ノート番号:0～127,$80,$D0($80で休符,$D0でウェイトとなる)
step time :0～65534
gate time :0～65535(65535はタイ/スラーとして扱われる)

●トラック終了 （ZMD専用命令）

$FF(.b)　計１バイト

このコードを発見すると演奏を終了してそのトラックはノンアクティブ(DEAD)となる

12.5. ZPDフォーマット

ZPDデータは以下のような内容で格納されている。

$10(.b)	ダミーコード
"ZmAdpCm"	ZPDファイルID
ノート番号(.w)	登録先のノート番号(0～511)
AD PCMデータまでのオフセットアドレス(.l)	
AD PCMデータ長(.l)	
ノート番号(.w)	登録先のノート番号(0～511)
AD PCMデータまでのオフセットアドレス(.l)	
AD PCMデータ長(.l)	
:	
$FFFF(.w)	エンドコード
AD PCMデータ…	

MEASURE13 ワークエリア

ここでは演奏トラックワークとその他のワークの内容についての解説を行います。Z-MUSIC支援ツールを制作するときなどに役立ちます。

13.1. 演奏トラックワーク

演奏トラックワークとは，各トラックの演奏状態をリアルタイムに指し示すワークエリアです。この部分を参照してビジュアルに演奏状態を表示したり，直接書き換えることによって演奏状態を変化させることも可能です。

13.1.1. 解説を読むにあたって

・演奏トラックのワークの解説は，
　ワーク名(ワークサイズ)　ワークオフセット
　{ワークの説明}
の書式で記述されています。ワークサイズは，
　(.b)　バイト
　(.w)　ワード(2バイト)
　(.l)　ロングワード(4バイト)
とします。
・同時に使用されることがないと保証されているワークアドレスは，重複して2つ以上のワークが設定されていることがあります（たとえば1/8モードピッチモジュレーションと波形メモリによるピッチモジュレーションは同時に使用できないため，関連ワークは重複するオフセットアドレスに設定されている)。
・解説文中では以下の略号が用いられています。
　PM　ピッチモジュレーション
　AM　アンプリチュードモジュレーション
・具体的なワークへのアクセスはリスト1，リスト2のようにして行ってください。

オフセット値について

Z-MUSIC用のデータで扱うオフセット値はオフセット値の格納してある次のアドレスからのオフセットです。たとえば，$A000番地にワードサイズのオフセット$0E00が格納されていたとするとポイントするアドレスは$A002＋$0E00＝$AE02となります。

13.1.2. 演奏トラックワークの解説

p_on_count(.w)　$00
現在発音中の音符の音長カウンタ(ステップタイム)。定められたテンポに従って－1されていく。0になった時点でZ-MUSICは次の音符を演奏しにいく

p_gate_time(.w)　$02
現在発音中の音符の発音長カウンタ(ゲートタイム)。定められたテンポに従って－1されていく。0になった時点でZ-MUSICはキーオフ処理を行う

p_data_pointer(.l)　$04
その時点のZMDコマンドポインタ。ステップタイムが0になるたびに，そのコマンド長分，増加していく

p_fo_spd(.b)　$08
フェードイン/アウトのスピードの絶対値。1≦p_fo_spd≦85

p_ch(.b)　$09
そのトラックがアサインされている絶対チャンネル。0≦p_ch≦24。PCM8独立チャンネルモードにおけるAD PCM拡張チャンネル番号については後述のp_extra_ch(.b)を参照

p_not_empty(.b)　$0a
そのトラックの生死を表す
　－1:死亡
　0:演奏中
　1:演奏終了
　$7f:同期待ち中
　$99:演奏停止中(m_stop時)
　$EE:同期待ち停止中(m_stop時)

p_amod_step(.b)　$0b
ディレイモードAM(拡張ARCC)の波形生成用整数増分

p_mstep_tbl(.w)　$0c～$1b
1/8モードPM(拡張PM)用波形生成用整数増分(×8)

p_wvpm_loop(.l)　$0c
PM(拡張PM)用波形メモリループ開始アドレス

p_wvpm_lpmd(.w)　$10
PM(拡張PM)用波形メモリループモード(－1,0,＋1)

p_altp_flg(.b)　$12
PM(拡張PM)用波形メモリ反復モードフラグ(0:順行中,1:逆行中)

p_fo_mode(.b)　$1c
現在のフェードモード(－1:フェードアウト
　＋1:フェードイン
　$47:ファンクション$47/$4C実行中)

p_pgm(.b)　$1d
現在の音色番号(0～199)

p_pan(.b)　$1e
現在のパンポット(0～3)

p_vol(.b)　$1f
現在の音量(127最小～0最大)

p_mrvs_tbl(.b)　$20～$27
1/8モードPM(拡張PM)用波形生成用補正値(×8)

p_wvpm_point(.l)　$20
PM(拡張PM)用波形メモリポインタ

p_wvpm_end(.l)　$24
PM(拡張PM)用波形メモリ終了アドレス

p_sp_tie(.w)　$28
FM音源互換タイモード(@J1)においてスラーを行ったときの2つの音の音程の差を格納してある
例
　c&e　では683×4＝2732がここに格納される

p_om(.b)　$28
FM音源音色パラメータのOMの値。(&B0000～&B1111)

p_sync(.b)　$29
FM音源音色パラメータのSYNCの値。(0,1)

p_af(.b)　$2a
FM音源音色パラメータのAFの値

p_se_mode(.b)　$2b
そのトラックが現在効果音モードかどうか
　$ff:normal
　$00:se mode
　$01:masking mode(FNC $44 or $4B)

p_pmod_tbl(.w)　$2c～$3b
1/8モードPM(拡張PM)の振幅値(×8)(-32768～32767)

p_total(.l)　$3c
その時点までのステップタイムの合計。ファンクション$19。または，ファンクション$43を実行後のみ意味を持つ

p_fo_lvl(.b)　$40
フェーダーの出力割合(MIN)0～128(MAX・NORMAL)

p_note(.b)　$42～$49
現在発音中のノート一覧(×8)

p_extra_ch(.b)　$4a
PCM8独立チャンネルモード時のAD PCM拡張チャンネル番号(0～7)

p_aftc_n(.b)　$4b
アフタータッチシーケンスのポインタ(0～7)

p_bend_rng_f(.w)　$4c
オートベンドのベンド幅(FM音源トラック専用)
半音＝64で表される相対表現値。(-768～768)

p_bend_rng_m(.w)　$4e
オートベンドのベンド幅(MIDIトラック専用)

半音≒683で表される相対表現値(-8192～8191)

p_detune_f(.w)　$50

ディチューン(FM音源トラック専用)(-768～768)

p_detune_m(.w)　$52

ディチューン(MIDIトラック専用)(-8192～8191)

p_port_dly(.w)　$54

ポルタメントディレイ(0～32767)

p_bend_dly(.w)　$56

ベンドディレイ(0～32767)

p_port_work(.b)　$58

ポルタメント用補正値ワーク

p_port_rvs(.b)　$59

ポルタメント用補正値

p_port_work2(.w)　$5a

ポルタメント/オートベンドにおける現在のピッチベンド値

FM音源トラックでは半音＝64,MIDIトラックでは半音≒683とする

p_amod_tbl(.b)　$5c～$63

1/8モードAMの振幅値(×8)(-128～127)

p_arcc_tbl(.b)　$5c～$63

1/8モードARCC(拡張ARCC)の振幅値(×8)(-128～127)

p_arvs_tbl(.b)　$64～$6b

1/8モードAM(拡張ARCC)用波形生成用補正値(×8)

p_wvam_point(.l)　$64

AM(拡張ARCC)用波形メモリポインタ

p_wvam_end(.l)　$68

AM(拡張ARCC)用波形メモリ終了アドレス

p_pmod_work4(.w)　$6c

PM(拡張PM)用スピードワーク

p_port_flg(.w)　$6e

ポルタメント実行中かどうか

0:off

−1 or +1:補正する方向(on)

p_bend_flg(.w)　$70

オートベンド実行中かどうか

0:off

−1 or +1:補正する方向(on)

p_aftc_tbl(.b)　$72～$79

アフタータッチシーケンス値(×8)(0～127,負数は省略値)

p_aftc_dly(.w)　$7a

アフタータッチシーケンス用　音長の1/8カウント

p_aftc_work(.w)　$7c

アフタータッチシーケンス用　音長の1/8カウントワーク

p_astep_tbl(.b)　$7e～$85

1/8モードAM(拡張ARCC)用波形生成用整数増分(×8)

p_wvam_loop(.l)　$7e

AM(拡張ARCC)用波形メモリループ開始アドレス

p_wvam_lpmd(.w)　$82

AM(拡張ARCC)用波形メモリループモード(−1,0,+1)

p_alta_flg(.b)　$84

AM(拡張ARCC)用波形メモリ反復モードフラグ(0:順行中,1:逆行中)

p_pmod_step2(.w)　$86

PM(拡張PM)用波形生成用整数増分(使い捨て)

p_pmod_work(.w)　$88

PM(拡張PM)用ディレイワーク

p_pmod_work2(.w)　$8a

PM(拡張PM)用　現在の音程

・FM音源トラックでは半音＝64

・MIDIトラックでは拡張モード1では半音＝64,拡張モード2では半音≒683とする

p_pmod_work3(.b)　$8c

PM(拡張PM)用波形生成用補正値ワーク

p_pmod_n(.b)　$8d

1/8モードPM(拡張PM)用ポインタ(0～7)

p_sync_wk(.b)　$8e

トラック同期コマンド用ワーク

$00:同期信号受信

$ff:同期待ち状態

p_rpt_last?(.b)　$8f

|:～:|の繰り返しコマンドにおいて,繰り返しが最後かどうか(bit1:last)

bit0:多重ループ深い

bit1:

:

bit7:多重ループ浅い

p_@b_range(.b)　$90

ピッチベンドレンジ(0～127)

p_arcc(.b)　$91

MIDIトラック:ARCC(拡張ARCC)で制御するコントロールチェンジ番号(0～127)

FM音源トラック:AMでどのオペレータに対してモジュレーション操作をするか

bit1:操作対象オペレータ

bit0～3:オペレータ1～4

p_pmod_flg(.w)　$92

現在PM(拡張PM)が実行中かどうか

・FM音源トラックのPM,MIDIトラックの拡張PMではワードサイズ

0:off

−1 or +1:補正する方向(on)

・MIDIトラックの通常PMではバイトサイズ

0:off

0以外:on

p_pmod_sw(.b)　$94

PM(拡張PM)のスイッチ

・FM音源トラックのPM, MIDIトラックの拡張PM

0:off

−1 or +1:補正する方向(on)

・MIDIトラックの通常PM

0:off

0以外:on

p_amod_sw(.b)　$95

AMのスイッチ

0:off

−1 or +1:補正する方向(on)

p_arcc_sw(.b)　$95

ARCC(拡張ARCC)のスイッチ

・通常ARCC

0:off

0以外:on

・拡張ARCC

0:off

−1 or +1:補正する方向(on)

p_bend_sw(.b)　$96

オートベンドのスイッチ

0:off

−1 or +1:補正する方向(on)

p_aftc_flg(.b)　$97

現在アフタータッチシーケンスを実行中か

0:off

0以外:on

p_md_flg(.b)　$98

汎用ワーク

bit0 ピッチベンドをリセットするかどうか

(MIDIトラック専用 0:no 1:yes)

bit1 ピッチモジュレーションをリセットするかどうか

(MIDIトラック専用 0:no 1:yes)

bit2 ARCC(拡張ARCCをリセットするかどうか

(MIDIトラック専用 0:no 1:yes)
bit3 MIDIのタイ/スラーモード
(MIDIトラック専用 0:normal 1:extended mode)
bit4 PM(拡張PM)の波形のリセットを行うかどうか
(0:行う 1:行わない)
bit5 AM(拡張ARCC)の波形のリセットを行うかどうか
(0:行う 1:行わない)
bit6 PM(拡張PM)の波形のキーオンとの同期非同期のモード
(0:同期 1:非同期)
bit7 AM(拡張ARCC)の波形のキーオンとの同期非同期のモード
(0:同期 1:非同期)

p_waon_flg(.b) $99
現在発音中の音は和音かそれとも単音か($00:単音 $ff:和音)

p_pmod_dly(.w) $9a
PM(拡張PM)用ディレイ(0～65534)

p_amod_dly(.w) $9c
AM用ディレイ(0～65534)

p_arcc_dly(.w) $9c
ARCC(拡張ARCC)用ディレイ(0～65534)

p_port_step(.w) $9e
ポルタメント/オートベンド用　音程整数増分
FM音源トラックでは半音＝64, MIDIトラックでは半音≒683とする

p_bank_msb(.B) $a0
MIDI音色バリエーション上位バイト(0～127)

p_bank_lsb(.B) $a1
MIDI音色バリエーション下位バイト(0～127)

p_effect1(.B) $a2
MIDI汎用エフェクト1(0～127)

p_effect3(.B) $a3
MIDI汎用エフェクト3(0～127)

p_ol1(.b) $a0
FM音源音色パラメータのオペレータ1のトータルレベル

p_ol2(.b) $a1
FM音源音色パラメータのオペレータ2のトータルレベル

p_ol3(.b) $a2
FM音源音色パラメータのオペレータ3のトータルレベル

p_ol4(.b) $a3
FM音源音色パラメータのオペレータ4のトータルレベル

p_d6_last(.b) $a4
拡張ARCCにおいて前回送信したコントロールチェンジ値を保存しておくワーク

p_cf(.b) $a4
FM音源の4つのオペレータのうちどのオペレータがキャリアかを示している
bit1:carrier
bit0～3:オペレータ1～4

p_amod_step2(.b) $a5
AM(拡張ARCC)用波形生成用整数増分ワーク(使い捨て)

p_vset_flg(.b) $a7
キーオン前に音量をリセットするか(0:しない 0以外:する)

p_arcc_rst(.b) $a8
ARCC(拡張ARCC)終了時の初期化値

p_arcc_def(.b) $a9
ARCC(拡張ARCC)開始時の基準値

p_coda_ptr(.l) $aa
[CODA]のあるアドレス

p_pointer(.l) $ae
[SEGNO]のあるアドレス

p_do_loop_ptr(.l) $b2
[DO]のあるアドレス

p_pmod_work5(.w) $b6
1/8モードPM(拡張PM)用音長の1/8カウント

p_pmod_work6(.w) $b8
1/8モードPM(拡張PM)用音長の1/8カウントワーク

p_amod_flg(.b) $ba
現在AMが実行中かどうか
0:off
−1 or +1:補正する方向(on)

p_arcc_flg(.b) $ba
現在ARCC(拡張ARCC)が実行中かどうか
・通常ARCC (0:off/0以外:on)
・拡張ARCC (0:off/−1 or +1:補正する方向(on))

p_aftc_sw(.b) $bb
アフタータッチシーケンスのスイッチ($00:off $00以外:on)

p_dumper(.b) $bc
ダンパー($00＝off $00以外＝on)

p_tie_flg(.b) $bd
前の音符にタイ/スラーの指定があったか(0:なかった 0以外:あった)

p_pmod_dpt(.w) $be
ディレイモードPM(拡張PM)の振幅値(1/8モード時は0になる)

p_seq_flag(.b) $c0
bit0:[D.C.]処理をしたことがあるか(0:ない 1:ある)
bit1:[FINE]処理をすべきかどうか(0:すべきでない 1:すべきである)
bit2:[CODA]を以前に設定したことがあるか(0:ない 1:ある)
bit3:[SEGNO]がいままでにあるかないか(0:ない 1:ある)
bit4:[D.S.]処理をしたことがあるか(0:ない 1:ある)
bit5:reserved
bit6:key off bit(キーオフ処理をするたびに, このビットが1になる)
bit7:key on bit(キーオン処理をするたびに, このビットが1になる)

p_do_loop_flag(.b) $c1
[DO]が以前に設定されているか($00:されていない $01～:現在の繰り返し回数)

p_pmod_spd(.w) $c2
PM(拡張PM)の波長の1/4カウント(1～32767)

p_amod_spd(.w) $c4
AM(拡張ARCC)の波長の1/4カウント(1～32767)

p_total_olp(.l) $c6
[DO]～[LOOP]外のステップタイムの合計。ファンクション$19または, ファンクション$43を実行後のみ意味を持つ

p_pmod_step(.w) $ca
ディレイモードPM(拡張PM)の波形生成用整数増分

p_tie_pmod(.b) $cc
タイ/スラーの途中でPM(拡張PM)関係のパラメータチェンジが行われたかどうか($00＝行われなかった $00以外:行われた)

p_tie_bend(.b) $cd
タイ/スラーの途中でオートベンド関係のパラメータチェンジが行われたかどうか($00＝行われなかった $00以外:行われた)

p_tie_amod(.b) $ce
タイ/スラーの途中でAM関係のパラメータチェンジが行われたかどうか($00＝行われなかった $00以外:行われた)

p_tie_arcc(.b) $ce
タイ/スラーの途中でARCC(拡張ARCC)関係のパラメータチェンジが行われたかどうか($00＝行われなかった $00以外:行われた)

p_tie_aftc(.b) $cf
タイ/スラーの途中でアフタータッチシーケンス関係のパラメータチェンジが行われたかどうか($00＝行われなかった $00以外:行われた)

p_pan2(.b) $d0
パンポット(L　M　R)
(0～64～127)

p_non_off(.b) $d1
キーオフなしモードかどうか($00:通常モード $00以外:キーオフなしモード)

p_frq(.b) $d2
AD PCMの再生周波数(0～6)
(ただし5,6はPCM8独立チャンネルモード時のみ有効)

p_velo(.b) $d3

ベロシティ(0～127)

p_amod_work4(.w) $d4
AM(拡張ARCC)の波長の1/4カウントワーク

p_pmod_rvs(.b) $d6
ディレイモードPM（拡張PM)用補正値

p_waon_dly(.b) $d7
和音用ディレイ値(0～255)

p_waon_work(.b) $d8
和音用ディレイワーク

p_waon_num(.b) $d9
和音において何番目のノートをキーオンするか(0～7)($ff:全ノートキーオン終了)

p_note_last(.b) $d9
単音で発音したノートのノート番号の一時退避ワーク

p_rpt_cnt(.b) $da～$e1
その時点での繰り返し回数($00は空きワーク 1～255:繰り返し回数)
$da:多重ループ深い
$db:
:
$e1:多重ループ浅い

p_maker(.b) $e2
そのトラックが操作している楽器のメーカーID
@Iコマンドの第1パラメータに相当

p_device(.b) $e3
そのトラックが操作している楽器のデバイスID
@Iコマンドの第2パラメータに相当

p_module(.b) $e4
そのトラックが操作している楽器のモデルID
@Iコマンドの第3パラメータに相当

p_last_aft(.b) $e5
前回アフタータッチシーケンスで出力されたアフタータッチ値

p_amod_work(.w) $e6
AM用ディレイワーク

p_arcc_work(.w) $e6
ARCC(拡張ARCC)用ディレイワーク

p_amod_work2(.b) $e8
AM用現在の音量(0～127)

p_arcc_work2(.b) $e8
ARCC(拡張ARCC)用現在のコントロール値(0～127)

p_amod_work3(.b) $e9
AM(拡張ARCC)用波形生成用補正値ワーク

p_amod_work7(.b) $ea
AM(拡張ARCC)用ノコギリ波生成用ワーク
(一周期終了後の，波形の開始値の記憶)

p_amod_n(.b) $eb
1/8モードAM用ポインタ(0～7)

p_arcc_n(.b) $eb
1/8モードARCC(拡張ARCC)用ポインタ(0～7)

p_arcc_work5(.w) $ec
1/8モードAM用音長の1/8カウント

p_amod_work5(.w) $ec
1/8モードARCC(拡張ARCC)用音長の1/8カウント

p_arcc_work6(.w) $ee
1/8モードAM用音長の1/8カウントワーク

p_amod_work6(.w) $ee
1/8モードARCC(拡張ARCC)用音長の1/8カウントワーク

p_pmod_wf(.b) $f0
PM(拡張PM)の波形形状
−31～−8:ユーザー波形番号
0:ノコギリ波
1:矩形波
2:三角波
3:ノコギリ波シングル

p_amod_dpt(.b) $f1
ディレイモードAM(拡張ARCC)の振幅値(1/8モード時は0になる)
(−128～127)

p_amod_wf(.b) $f2
AM(拡張ARCC)の波形形状
−31～−8:ユーザー波形番号
0:ノコギリ波
1:矩形波
2:三角波
3:ノコギリ波シングル

p_dmp_n(.b) $f3
FM音源トラック用ダンパー処理ワーク
p_ch+p_dmp_nしたチャンネルで次回の発音を行う
(0～7)

p_pmod_omt(.b) $f4
1/8モードPM(拡張PM)用パラメータ省略ビットパターン
bit0:省略
bit1:有効
bit0～7:第1パラメータ～第8パラメータ

p_amod_omt(.b) $f5
1/8モードAM用パラメータ省略ビットパターン
bit0:省略
bit1:有効
bit0～7:第1パラメータ～第8パラメータ

p_arcc_omt(.b) $f5
1/8モードARCC(拡張ARCC)用パラメータ省略ビットパターン
bit0:省略
bit1:有効
bit0～7:第1パラメータ～第8パラメータ

p_pmod_mode(.b) $f6
拡張PMのモード番号
−1:通常PMモード
0:半音＝64モード
1:半音≒683モード

p_arcc_mode(.b) $f7
拡張ARCCのモード番号
−1:通常ARCCモード
0:拡張ARCCモード

p_pmod_chain(.b) $f8
1/8モードPM(拡張PM)波形接続ワーク
−1:接続待機
0:波形通常実行中
1:強制接続

p_amod_chain(.b) $f9
1/8モードAM，ARCC(拡張ARCC)波形接続ワーク
−1:接続待機
0:波形通常実行中
1:強制接続

p_jump_flg(.b) $fa
[!]コマンドステータスワーク
−1:通常演奏状態
0:ジャンプ実行中
1:ジャンプ終了

p_waon_mark(.b) $fb
主チャンネルのパラメータ変更をしたか(0:していない 0以外:した)

p_marker(.w) $fc
p_marker+0(.b):このトラックは効果音トラックか
(0:通常トラック 0以外:効果音トラック)
p_marker+1(.b):フェーダー移動フラグ(オーバーフローするとフェーダーが動く)

p_amod_rvs(.b) $fe
ディレイモードAM，ARCC(拡張ARCC)用補正値

p_ne_buff(.b) $ff

効果音トラックにおけるp_not_empty(.b)の一時退避ワーク

p_user(.b)　　$ff

ユーザー汎用ワーク

13.1.3. ワーク参照にあたっての注意

ワークの不当な書き換えは，演奏に支障をきたすだけでなく，Z-MUSICの暴走を引き起こす可能性もあるため，細心の注意を必要とします。また，特殊な使用目的以外では，各ワークは読み出し専用と考え，書き換えはなるべく行わないようにすべきでしょう。

各ワークは，必ずしも使用されているとは限らず，条件によっては，そのワークが持つ値になんの意味も持たない場合があります。たとえばPMのスイッチp_pmod_sw(.b)がOFF(0)の場合，PM関連ワークに格納されている値はなんの意味も持ちません。

また，MMLやコマンドで与えたパラメータ値とワーク内部で表現される内部表現形式とは多少の食い違いがあるので注意してください。たとえば音色番号はMMLパラメータでは，

@〈音色番号:1～200〉

ですが，ワーク(内部表現)では，

p_pgm(.b):0～199

となっています。

13.1.4. プレゼンハムのアルゴリズムについて

Z-MUSICではオートベンドやポルタメント，ピッチモジュレーションやアンプリチュードモジュレーションなどの波形の生成実行にプレゼンハムのアルゴリズム(Bresenham's algorithm)を使用しています。このパラメータ計算とその実行について解説しておきます。

●パラメータの計算

図1のようなv1→v2への値変化を行いたい場合を考えます。つまり，絶対カウントst間にちょうどv1からv2へ幅yだけ変化させたいということです。

整数増分パラメータは，

〈整数増分パラメータ〉=int(y/st)

(ただしst≠0。yは正数，負数いずれも可)

そして補正パラメータは，

〈補正パラメータ〉=abs(y mod st)＊256/st

〈補正方向〉=sgn(y)

で求められます。

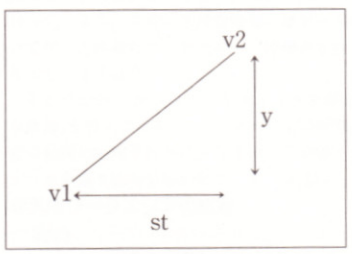


図1 パラメータ

●実行

実際の動作では，

st≠0

v1←初期値

rvs_wk(補正値ワーク):=0

として，

repeat

v1:=v1+〈整数増分パラメータ〉

rvs_wk:=rvs_wk+〈補正パラメータ〉

if (Carry flag is set) then v1:=v1+〈補正方向〉

st:=st-1

until st=0

のようなアルゴリズムにより実行されます。

なお，演奏トラックワークとプレゼンハムのアルゴリズムにおけるパラメータとの対応は，

ポルタメント

補正値:p_port_rvs(.b)　$59

補正方向:p_port_flg(.w)　$6e

整数増分:p_port_step(.w)　$9e

補正値ワーク:p_port_work(.b)　$58

計算結果:p_port_work2(.w)　$5a

オートピッチベンド

補正値:p_port_rvs(.b)　$59

補正方向:p_bend_flg(.w)　$70

整数増分:p_port_step(.w)　$9e

補正値ワーク:p_port_work(.b)　$58

計算結果:p_port_work2(.w)　$5a

ピッチモジュレーション

補正値:p_pmod_rvs(.b)　$d6

補正方向:p_pmod_flg(.w)　$92

整数増分:p_pmod_step2(.w)　$86

補正値ワーク:p_pmod_work3(.b)　$8c

計算結果:p_pmod_work2(.w)　$8a

アンプリチュードモジュレーション/拡張ARCC

補正値:p_amod_rvs(.b)　$fe

補正方向:p_amod_flg(.b)　$ba

整数増分:p_amod_step2(.b)　$a5

補正値ワーク:p_amod_work3(.b)　$e9

計算結果:p_amod_work2(.b)　$e8

のようになっています。また，1/8モードのピッチモジュレーションや，1/8モードのアンプリチュードモジュレーションでは整数増分と補正値が8つ分テーブルに格納されており，随時使用されます。

●1/8モードピッチモジュレーション

補正値テーブル:p_mrvs_tbl(.b)　$20～$27

整数増分テーブル:p_mstep_tbl(.w)　$0c～$1b

●1/8モードアンプリチュードモジュレーション/拡張ARCC

補正値テーブル:p_arvs_tbl(.b)　$64～$6b

整数増分テーブル:p_astep_tbl(.b)　$7e～$85

13.2. バッファ管理ワーク

ファンクション$45でZ-MUSICの各バッファのアドレスを知ることができます。ここでは，これらのバッファの構造や役割について解説します。

●トラックバッファ

トラックバッファはZMDを格納するために確保されていますが，MDD作成時に汎用ワークとして使用されることがあります。外部アプリケーションで確保したメモリ領域をZ-MUSICから参照させるには，

(trk_top)＝メモリ領域の先頭アドレス

(trk_buf_size)＝メモリ領域のサイズ

(trk_buf_end)＝メモリ領域の最終アドレス

を設定します。

●AD PCMバッファ

AD PCMバッファにはZPDをはじめとしたAD PCMデータが格納されています。AD PCMバッファのサイズが0のときはAD PCM関連の命令やファンクションは使用できません。外部アプリケーションで確保したメモリ領域をZ-MUSICから参照させるには，

(adpcm_buffer_top)＝メモリ領域の先頭アドレス

(adpcm_buffer_size)＝メモリ領域のサイズ

(adpcm_buffer_end)＝メモリ領域の最終アドレス

(adpcm_buffer_next)＝メモリ領域の先頭アドレス

を設定します。

●汎用ワークエリア

汎用ワークエリアはAD PCMコンフィギュレーション一時退避場所としてや，AD PCMデータ加工時のワークエリア，波形メモリのバッファ(ZMS時)，MDD作成時のワークエリアとして多目的に使用されます。

AD PCMコンフィギュレーション一時退避場所，波形メモリのバッファ(ZMS時)として使用した場合は便宜上のワークサイズ(adpcm work_size)が伸縮します。

外部アプリケーションで確保したメモリ領域をZ-MUSICから参照させるには，

(adpcm_work_top)＝メモリ領域の先頭アドレス
(adpcm_work_size)＝メモリ領域のサイズ
(adpcm_work_end)＝メモリ領域の最終アドレス
(adpcm_work_now)＝メモリ領域の先頭アドレス
(adpcm_work_true_size)＝メモリ領域のサイズ

を設定します。

●演奏トラックワーク

演奏中の各トラックのワーク。トラックは80本あり(ZMSC.Xでは32本)1本のトラックにつき256バイトのワークが割り当てられています。各ワークの役割については13.1.を参照してください。

(seq_wk_tbl)＋トラック番号(0～79)×256

で任意のトラックのワーク先頭アドレスを求めることができます。

●効果音トラックワーク

効果音トラックの演奏トラックワーク。トラックは32本あり，役割，ワークサイズはまったく演奏トラックワークと同等です。

●AD PCM管理テーブル

各ノート番号に割り当てられたAD PCMデータを管理するワーク。1バンクあたり，

ノート番号0　AD PCMデータアドレス(.L),AD PCMデータサイズ(.L)
ノート番号1　AD PCMデータアドレス(.L),AD PCMデータサイズ(.L)
ノート番号2　AD PCMデータアドレス(.L),AD PCMデータサイズ(.L)
　　：
ノート番号127 AD PCMデータアドレス(.L),AD PCMデータサイズ(.L)

という構成を取っています。これが4バンク分並んでいるわけです(8×128×4＝4096バイト)。

任意のバンク番号(1～4)，ノート番号(0～127)に対応する管理ワークアドレスは，

(adpcm_tbl)＋(バンク番号－1)×128×8＋ノート番号×8

にて求めることができます。

●波形メモリ管理テーブル

各波形番号に割り当てられた波形メモリデータ列を管理するワークです。

波形番号8　start address(.L),end address(.L),
　　loop point(.L),loop mode(.W)
波形番号9　start address(.L),end address(.L),
　　loop point(.L),loop mode(.W)
波形番号10 start address(.L),end address(.L),
　　loop point(.L),loop mode(.W)
　　：
波形番号31 start address(.L),end address(.L),
　　loop point(.L),loop mode(.W)

という構成を取っています(14×24＝336バイト)。アドレスとループポイントは絶対アドレス，ループモード値は0～2で与えたものが－1,0,＋1にパラメータシフトされている点に注意してください。

任意の波形番号に対応したワークアドレスは，

(wave_tbl)＋(波形番号－8)×14

にて求めることができます。

●FM音源音色バッファ

FM音源の音色は1音色につき55バイトで構成されています。よって，

(neiro)＋(音色番号(1～200)－1)×55

で，任意の音色の格納アドレスを参照できます。

●各トラックの先頭アドレス

各トラックの先頭アドレスはtrk_po_tblに格納されているアドレスから，

トラック1の先頭アドレス(.L)
トラック2の先頭アドレス(.L)
トラック3の先頭アドレス(.L)
　　：
トラック80の先頭アドレス(.L)

のように格納されています。よって，任意のトラック番号のトラックの先頭アドレスは，

((trk_po_tbl)＋トラック番号(0～79)×4)

で求めることができます。

●各トラックの長さ

各トラックの長さはtrk_len_tblに格納されているアドレスから，

トラック1の長さ(.L)
トラック2の長さ(.L)
トラック3の長さ(.L)
　　：
トラック80の長さ(.L)

のように格納されています。よって任意のトラック番号のトラックの長さは，

((trk_len_tbl)＋トラック番号(0～79)×4)

で求めることができます。

このワークの値は「ファンクション$01 m_alloc」が実行された場合のみ意味を持ちます。ZMDファイルを演奏した場合はm_allocされないのでワークの値は無意味です。

MEASURE14
効果音モードと映像同期モード

ここでは，Z-MUSICの特殊機能である「効果音モード」と「映像同期モード」についての解説を行います。本章を理解するにはMPU68000の機械語の知識が必要です。通常の音楽演奏のみを行うユーザーは読み飛ばして構いません。「効果音モード」の解説はオリジナルゲームプログラムで実際に使用することを想定して具体的に解説します。

14.1. 効果音モード

14.1.1. ZMUSIC.Xの組み込み

●組み込みメッセージの制御

−G

ZMUSIC.Xを組み込んだあとにZ-MUSICのロゴやバージョン番号，確保したバッファの容量などの情報が画面に出力される。このスイッチを設定するとこれらをいっさい表示しないようにすることができる。機能自体に特別な深い意味はないがアプリケーションを起動する際の画面バランスに気を使う場合は設定するとよい。

●多重割り込みに対応させる

−M

ゲームのメインプログラムでラスタースクロールやその他のMFPを使った割り込みを使用している場合には設定したほうがよい。設定しないとZ-MUSICの割り込み処理中に一切の割り込みがキャンセルされてしまい，希望どおりの処理がなされない場合がある。

しかしこのモードでMFPを使った割り込み処理が極端に重くなった場合は，音楽側のテンポに影響が出ることがある。

PCM8モード時には暗黙のうちにこのモードが選択される。

●初期化制御

−N

Z-MUSICでは通常，新たに演奏データを演奏しようとするとMIDI楽器に初期化パラメータを設定したりFM音源の初期化などを実行している。この初期化処理は新たに演奏する曲の都合を無視したものである。たとえばMIDI音源の10チャンネル以下しか使用していない曲をこれから演奏しようというときに11チャンネル以上の初期化も行ってしまう。この「余計な処理」はゲームの曲の切り換え時においてゲーム画面のスクロールの一時停止といった目に見える症状になってしまう場合がある。

そこでこの'−N'オプションスイッチを設定すると以後この「余計な処理」を省くことができる。ただしFM音源/MIDI楽器に対して一切の初期化処理を行わなくなるので各演奏データには必要な初期化データを盛り込む必要が出てくる（後述）。

●汎用ワークエリアの設定

−Wn

ゲームなどの曲で演奏のたびにAD PCMコンフィギュレーションファイルを読んだりAD PCMの加工処理を行ったりするのはまったくの非常識であるため汎用ワークは不要ということになる。よって普通は，

−W0

でいいことになる。

●AD PCMデータ

−B ZPDファイルネーム

−Pn

ゲーム中に用いるすべてのBGMがひとつのZPDデータを用いてる場合は，

−B　ZPDファイルネーム

でドライバ組み込み時に組み込んでしまうとよい。'−B'は組み込むZPDファイルのファイルサイズ分自動的にAD PCMバッファを確保してくれるため便利である。

ゲーム中に数回ZPDデータを切り換えたりする場合は'−P'で必要最大分バッファを確保して，あとからファンクションコールなどを使ってZPDデータやAD PCMデータを登録するということになる。

また，演奏する曲がまったくAD PCMを使用しない場合はAD PCMバッファを特に確保する必要はない。

●トラックバッファ

−Tn

曲データをゲームプログラム開始前に一括してメインプログラムの管理領域に読み込んでしまうのが賢明である。そして曲の切り換え時にはあらかじめ読み込んでおいた演奏データの先頭アドレスをZ-MUSICに与えてやればよいのである（具体例は後述）。ということはつまりZ-MUSICのトラックバッファはまったく不要ということであるから，その場合，

−T0

でよいことになる。

また，ゲームのメインプログラムが大きくて一括して読み込めない場合は1ステージ内に必要分を先読みするようにすればよい（例ステージBGM，ボスBGM）。

14.1.2. ゲームに負担をかけない演奏データとは?

まず，演奏データ制作者はゲームのプログラムに負担をかけないように心掛けて演奏データを作成しなければなりません。また，曲の切り換わりに時間がかからないようにしなければなりません。たとえばスクロールシューティングゲームでステージBGMからボスのテーマに切り換わるときに，スクロールが止まってしまってはスピード感を損ねてしまいます。

したがって曲データはすべて応答の高速であるZMDレベルで管理すべきです。さもないと曲が切り換わるたびにコンパイルする動作を強いられ，曲の切り換えがとても遅くなるからです。

ここでは曲の切り換わりの速い演奏データを作るためのテクニックについて述べます。

●共通コマンドは使わない

FM音源音色の定義コマンドやMIDIエクスクルーシブコマンドなどのトラックに依存しない，いわゆる「共通コマンド」は割り込みで処理されないために曲の切り換え時のウェイトになりかねません。つまりZMSファイルをコンパイルしてZMDを得たときに共通コマンド部分のZMDコードが皆無ならばよいのです。

そこでMMLで簡単に置き換えられる共通コマンドはすべてMMLに置き換えたほうがよいといえます。たとえばテンポ設定ZMSコマンド(On)は用いずMMLのテンポコマンドを用いるようにしたり，'ROLAND_EXCLUSIVE'命令や'(Xn 1,n 2,n 3,…,n i)'などはXコマンドや，@Xコマンドを用いたりしてMMLへ直していくのです。

また，どうしても置き換わらないもの，

音色登録

MIDI楽器個別コマンド

などは後述する方法で解決します。

結局ゲームに用いる演奏データには，

初期化コマンド…(I)

ベースチャンネル設定…(Bn)

トラック確保…(Mtr,size)

チャンネルアサイン…(Ach,tr)

以外の共通コマンドを用いないようにしましょう。以上の4つのコマンドはコンパイル時に消化され，共通コマンドコードとしてZMDデータに生成されません。

COMCHK.R

開発途中はいろいろと曲のデータのほうも日に日に手を入れられ改良されていくでしょうから，共通コマンドを撲滅するのは最終的な段階で構わないでしょう。

そこで最終的なチェック用としてコンパイルして生成したZMDデータに共通コマンドがあるかないかを検査するプログラム「COMCHK.R」を提供します。

使い方は，

COMCHK　ZMDファイル

とするだけです。共通コマンドがあれば「ある」と，なければ「ない」という結果を出力してくれます。

●音色登録

演奏データにはさまざまな音色が使用されますが，この音色データを1曲ごとに持たせて演奏の開始時にいちいち登録していたのでは曲の切り換わりが遅くなってしまいます（アクションゲームのような極端な高速切り換えが要求されていないRPGやADVなら別に構わないであろうが）。

Z-MUSICではFM音源200音色分(No.1～No.200)の専用バッファを持っており，ドライバが解除されたりしなければ，この領域はいつまでも保存されています。よって，そのゲーム中に使用されるFM音色をあらかじめドライバの組み込み時にオプションスイッチ'-S'で転送してしまい，演奏データにはいっさいのFM音色データを持たないようにすればよいのです。効果音で用いられるFM音色も一緒に送ってしまうとよいでしょう。

MIDI楽器の音色の場合もFM音源音色の場合と同様です。そのゲーム中のBGMで使用する音色データをMIDI楽器へあらかじめ登録してしまえばよいのです(MIDI楽器の初期化コマンドを実行すると設定した音色までも初期化されることがあるので注意)。

音色のセットアップファイルの例をリスト1に示します。このリスト1のようなものを，

```
A>zmusic  -sSETUP.ZMS
```

のようにしてドライバ常駐とともに組み込んでしまえばよいのです。

もちろんコンパイルして，

```
A>zmusic  -sSETUP.ZMD
```

のようにしても構いません。

複数のMIDI楽器に対応している場合はMUSICコンフィギュレーションのような場所で楽器を選ばせ，その時点であらかじめ用意しておいたその楽器の音色データをセレクトして転送するようにすればよいでしょう。

2通りの演奏データ形式

ゲームの演奏データを設計する場合に2通りのバリエーションが存在します。ひとつはMIDI対応曲と内蔵音源対応曲を分けてそれぞれ用意する場合（Oh!X1992年6月号付録の「SIONII」やコナミの「出たな!!ツインビー」など）。もうひとつは内蔵音源がメインでMIDIボードがある場合はMIDI音源もこれに重なって鳴るといった，ひとつのデータで両方をまかなうタイプのもの（コナミの「パロディウスだ!」など）。

Z-MUSICでは常駐処理のときにMIDIボードの有無を検査してMIDIボードがなければ以後，MIDI部の演奏を自動的にミュートして内蔵音源部のみ演奏できるモードになります。

●MIDI楽器個別コマンド

音色データやグローバルなMIDI楽器の設定はゲームが開始される前にあらかじめ登録しておくことで解決しますが，1曲ごとに変えたい設定もあります。たとえばパーシャルリザーブなどがそうです。

楽器個別命令も共通コマンドであるためなんとかMMLへコンバートしたいところです。こういったものはエクスクルーシブメッセージで処理を実現しているためMMLの@XやXへ変換できなくはないのですが，いちいち楽器のマニュアルを見ながらMMLを書いていくのは大変です。

こういう場合には同梱の支援ツール「ZMD18.R」を使用するとよいでしょう。このZMD18.RはZMDデータ中にあるMIDI楽器個別コマンドをMMLに変換するものです。具体的な使い方を以下に示します。

たとえばMUSIC.ZMSというMIDI楽器個別コマンドが多用されたZMSファイルがあったとしましょう。これを，

```
A>zmusic -c MUSIC
```

としてコンパイルし，MUSIC.ZMDを得ます。

次に，

```
A>zmd18 MUSIC
```

とすると画面に，

```
(T1)@X$F0,$41,$10,$16,$12,$7F,$00,$00,$00,$01,$F7
(T1)@X$F0,$41,$10,$16,$12,$08,$00,$00,$43,$48,$4F,$52,$44,$20,$20,$20
(T1)@X$20,$20,$00,$00,$03,$00,$24,$35,$0B,$01,$02,$00,$24,$09,$00,$00
(T1)@X$00,$00,$00,$00,$00,$32,$32,$32,$32,$32,$3E,$17,$55,$64,$17,$04
(T1)@X$27,$08,$18,$64,$00,$00,$00,$0D,$18,$4D,$53,$64,$59,$4C,$48,$5E
(T1)@X$4A,$5B,$0C,$1B,$0C,$00,$00,$00,$00,$00,$09,$2B,$64,$64,$64,$63
(T1)@X$24,$30,$0B,$01,$03,$00,$00,$00,$00,$00,$00,$00,$00,$00,$32
(T1)@X$32,$32,$32,$32,$40,$17,$38,$64,$17,$05,$67,$06,$2B,$64,$00,$00
(T1)@X$00,$07,$16,$64,$62,$64,$5E,$54,$4F,$5E,$46,$5B,$0C,$1B,$0C,$00
(T1)@X$00,$00,$04,$0B,$0E,$0F,$64,$60,$5C,$5B,$6B,$F7
(T1)@X$F0,$41,$10,$42,....
(T1)@X.....
        :
```

といったものが出てきます。これらがZMD中のMIDI楽器個別コマンドがMMLへとコンバートされたものです。

これをリダイレクトしてどこかに保存し，MUSIC.ZMSのトラック1の先頭へ挿入します。また，EOX($F7)の後ろには若干のウェイトが必要なので休符を添付してください。あとは，MUSIC.ZMS中の共通コマンドをすべて削除すれば作業の終わりです。

●コメント

演奏データのタイトルや作者名を記述するのに使われている.comment文も演奏時には文字列をスキップする動作を強いられるため，

リスト1

```
*****************  M14_L1.ZMS  *****************
                    効果音FM音色データ
(v1.0                                                    /自機ミサイル
/   AF  OM  WF  SY  SP PMD AMD PMS AMS PAN
    60, 15,  0,  0,  0,  0,  0,  0,  3,  0
/   AR  DR  SR  RR  SL  OL  KS  ML DT1 DT2 AME
    27, 15,  5,  2,  0,  0,  0,  3,  1,  0
    31, 18, 18,  6,  7,  0,  0,  0,  3,  2,  0
    22, 31,  0, 10,  0, 42,  0,  7,  7,  0,  0
    15, 31,  0,  8,  0,  0,  2,  1,  0,  0)
(v2.0                                                    /敵レーザー
/   AF  OM  WF  SY  SP PMD AMD PMS AMS PAN
    58, 15,  0,  0,  0,  0,  0,  0,  3,  0
/   AR  DR  SR  RR  SL  OL  KS  ML DT1 DT2 AME
    12,  5,  1,  5,  5, 24,  1,  0,  0,  1,  0
    16,  4,  1,  5,  2, 30,  1,  0,  3,  1,  0
    29,  6,  5,  5,  2, 16,  2,  0,  3,  2,  0
     9,  4, 31,  8, 10,  0,  2,  4,  3,  1,  0)
                        :
                        :
                        :
(v29.0                                                   /敵ダメージ
/   AF  OM  WF  SY  SP PMD AMD PMS AMS PAN
    61, 15,  0,  0,  0,  0,  0,  0,  3,  0
/   AR  DR  SR  RR  SL  OL  KS  ML DT1 DT2 AME
    31,  5,  3,  3,  1, 18,  0,  0,  3,  0,  0
    18,  0,  0,  5,  3,  0,  0,  1,  0,  2,  0
    31,  0,  0,  5,  2,  0,  0,  3,  7,  3,  0
    18,  0,  0,  5,  2,  0,  0,  3,  7,  1,  0)
                    楽器FM音色データ
/  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME   BASS 1
(@30,       21,  0,  0,  0,  1, 20,  1,  0,  7,  0,  0
    21,  0,  0,  4,  0,  0,  1,  1,  7,  0,  0
    21,  0,  0,  0,  0, 20,  1,  0,  3,  0,  0
    21,  0,  0,  4,  0,  5,  1,  1,  3,  0,  0
/  AL  FB  OM
    4,  5, 15)
/  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME   SYNTH1
(@31,       21,  0,  0,  0,  0, 25,  0,  2,  0,  0,  0
    21,  0,  0,  9,  0,  6,  0,  4,  0,  0,  0
    21,  0,  0,  9,  0,  0,  0,  2,  0,  0,  0
    21,  0,  0,  9,  0,  5,  0,  2,  0,  0,  0
/  AL  FB  OM
    5,  7, 15)
                        :
                        :
                        :
                        :
/  AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME   BRASS1
(@100,      19, 17,  0,  8,  1, 24,  1,  1,  1,  0,  0
    19, 13,  0,  8, 15, 37,  1,  2,  0,  0,  0
    19, 17,  0,  8,  1, 36,  1,  1,  0,  0,  0
    19,  4,  5,  8,  2,  0,  1,  1,  2,  0,  0
/  AL  FB  OM
    2,  7, 15)
/-------------------------------------------------------------
/          MIDI楽器音色
/       Chord (Saw&Squ)
.roland_exclusive 16,22 =[8,0,0
         67, 72, 79, 82, 68, 32, 32, 32, 32, 32
          0,  0,  3,  0
/ ** PART1AL1 **
         36, 53, 11,  1,  2,  0, 36,  9
          0,  0,  0,  0,  0,  0,  0, 50, 50, 50, 50, 50
         62, 23, 85
        100, 23,  4, 39,  8, 24
        100,  0,  0,  0, 13, 24, 77, 83,100, 89, 76, 72
         94, 74, 91, 12, 27, 12
          0,  0,  0,  0,  0,  9, 43,100,100,100, 99
/ ** PART1AL2 **
         36, 48, 11,  1,  3,  0,  0,  0
          0,  0,  0,  0,  0,  0,  0, 50, 50, 50, 50, 50
         64, 23, 56
        100, 23,  5,103,  6, 43
        100,  0,  0,  0,  7, 22,100, 98,100, 94, 84, 79
         94, 70, 91, 12, 27, 12
          0,  0,  0,  4, 11, 14, 15,100, 96, 92, 91]
.mt32_patch 1,16 =[2,0,24,52,2,0,1]
/-------------------------
/       keyboard (Harpsichord)
.roland_exclusive 16,22 =[8,4,0
         72, 65, 82, 80, 83, 73, 67, 79, 82, 68
          2,  5,  3,  0
/ ** PART1AL1 **
         36, 46, 16,  1,  0, 43,  0,  7
          0,  0,  0,  0,  0,  0,  0, 50, 50, 50, 50, 50
          0,  0,  0
          0,  0,  3,  0,  7,  0
          0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0
         87, 77, 91,  6, 27, 12
          3,  0,  0, 23, 60, 83,  0,100, 88, 88, 88
/ ** PART1AL2 **
         48, 51, 16,  1,  2,  0,100,  7
          0,  0,  0,  0,  0,  0,  0, 50, 50, 50, 50, 50
          0,  0,  0
        100, 16,  9,103,  8,  0
          0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0
        100, 75, 27, 12, 15,  5
          1,  1,  0, 23, 60, 92,  0,100, 88, 88, 88]
.mt32_patch 2,16 =[2,2,24,50,2,0,1]
                        :
                        :
```

これがウェイトになりかねません。できるだけ用いないようにしましょう。

●テンポ

曲のテンポは共通コマンド(On)を用いずにMMLのTコマンドを用いて変化させたほうがいいということは，すでに述べたとおりです。

ゲーム中に用いる曲の場合，テンポがステージごとにあまりにも速かったり遅かったりすると効果音もそのテンポにつられて速くなったり遅くなったりしてしまいます。たとえばあるステージでBGMがテンポ100で演奏されていてレーザーの発射音が「ちゅーんちゅーん」だったとしましょう。そして別のステージに突入し，BGMのテンポが倍の200になったとすると「ちゅんちゅん」になってしまうのです。

これを防止（というか辻褄をあわせる）するのが(Zn)コマンドです。この命令は全音符の絶対音長をデフォルトの192からnへ変更するものです。

テンポが速くなるというのはタイマ割り込みの割り込み間隔が短くなるということですから，この割り込み間隔を一定にしつつ曲のテンポを変えることを考えればいいのです。

先ほどの，BGMのテンポが速くなったために効果音も早く鳴ってしまった例は，(効果音と，元のBGMが全音符の絶対音長が192基準で作られているならば)，テンポ200の曲のほうを全音符の絶対音長を96にしてやれば，テンポ200でも割り込み間隔はテンポ100とまったく同じものとなり，この曲はテンポ200，効果音はテンポ100で演奏され万事解決となります。

つまりタイマ値をだいたい一定（あんまり厳密でなくて構わないが）にし，ひとつの音符の絶対音長の長短でテンポを調整するイメージです。テンポMMLコマンドTはこの全音符の絶対音長変更コマンドの(Zn)を考慮してタイマ値を算出してしまうのでタイマ値を直接設定できる@Tコマンドを用いるとわかりやすいでしょう。

ところでこの(Zn)コマンドは共通コマンドですからZMDデータとしても生成されてしまいます。共通コマンドの削減をこれまで繰り返しいってきましたが，このコマンドは実行にたいして時間を食わないので，例外として扱って構わないでしょう。

●初期化

前節で解説したようにZMUSIC.Xを'−N'オプションをつけて初期化なしモードで常駐させている場合，各演奏データに楽器初期化などのコマンドを盛り込まないと前の演奏データの設定状態が受け継がれることになってしまいます。

Z-MUSICでは初期化なしモードでない通常モードの場合，新たな演奏データの演奏開始直前に以下のようなメッセージを送信しています。

●FM音源部

ノイズモードオフ(Y15,0相当)

●AD PCM音源

特になし。

●MIDI音源部

リセットオールコントローラーズ($Bn,$79,$00)
オムニモードオン($Bn,$7D,$00)
モノモードオフ($Bn,$7F,$00)
ローカルオン($Bn,$7A,$7F)
ファインチューニングをニュートラルへ
($Bn,$65,$00,$Bn,$64,$01,$Bn,$06,$40,$Bn,$26,$00)
コースチューニングをニュートラルへ
($Bn,$65,$00,$Bn,$64,$02,$Bn,$06,$40)
ピッチベンドレンジ＝1オクターブ
($Bn,$65,$00,$Bn,$64,$00,$Bn,$06,$0C)
ボリューム＝63($Bn,$07,$3F)
パンポット＝中央($Bn,$0A,$40))
ピッチモジュレーション＝0($Bn,$01,$00)
ダンパーオフ($Bn,$40,00)
ピッチベンドをニュートラルへ($En,$00,$40)

以上から必要なものをピックアップして演奏データに盛り込むべきでしょう。初期化なしモードとそうでない場合の違いはこれらの初期化動作が行われるか，行われないかの違いになってきます。こうしてみるとFM音源とAD PCM音源は初期化なしモードと通常モードの区別を特に意識する必要はないかもしれません。

演奏データをゲームプログラムなどに取り込む前に一度，初期化なしモードで演奏してみるとよいでしょう。それで演奏がおかしかったならば，どの初期化メッセージが足りなかったのかを見極めて，随時それらを盛り込んでいくとよいでしょう。

ローランド系のDTM楽器（CMやMT，SCシリーズなど）では楽器のメモリアドレスのある箇所にダミーデータを書き込むと本体内の全パートを初期化してくれる便利な機能があります。これを使えば上のような初期化メッセージをいちいち送らなくても少ない手間で初期化が行えます。いくつか例を挙げておきましょう。
(初期化したあとの楽器の状態については各楽器のマニュアルを参照し確認すること)

●MT-32/CM-32L/CM-64/CM-32P/CM-500のLA音源部

リスト2

```
*************** M14_L2.ZMS ******************
  1: .comment An old picture. by Z.Nishikawa (for SCS5/SC155/CM300/CM500)
  2: {i}
  3: {b0}
  4: {m1,2000} {a midi1 , 1}
  5: {m2,2000} {a midi2 , 2}
  6: {m3,2000} {a midi3 , 3}
  7: {m4,2000} {a midi4 , 4}
  8: {m5,2000} {a midi5 , 5}
  9: {m7,2000} {a midi10, 7}
 10: {m8,2000} {a midi10, 8}
 11: {m9,2000} {a midi10, 9}
 12: {m10,2000}{a midi10,10}
 13: {m11,2000}{a midi10,11}
 14: {o125}
 15: .roland_exclusive $10,$42 ={$40,0,$7f,0}          / initialize SC55
 16: .mc55_v_reserve={4,2,6,3, 3,0,0,0 ,0,6,0,0 ,0,0,0,0}/パーシャルリザーブ
 17:
 18: {t1}@1$41,$10,$42 @e70,70 @36 v10 @p40        r16|do|
 19: {t2}@1$41,$10,$42 @e80,60 @36 v15             r16|do|
 20: {t3}@1$41,$10,$42 @e70,70 @49 v10 @p99 @k-8   r*7|do|
 21: {t4}@1$41,$10,$42 @e80,80          @p54 @h36  r16|do|@m40
 22: {t5}@1$41,$10,$42 @e80,80          @p74 @h36 r*23|do|@m40
 23: {t7}@1$41,$10,$42 @e80,10 @1                  @r1   r16|do|
 24: {t8}                                          @r1   r16|do|
 25: {t9}                                            @r1   r16|do|
 26: {t10}                                         @r1   r16|do|
 27: {t11}                                         @r1   r16|do|
 28:
 29: /  DS.GUITAR
 30: {t1} o2 q8 @u99 L1 'e-a-<d-' 'd-g-b' 'g-b<d-' 'e-a-<d-'
 31:
 32: /  PRETLESS BASS
 33: {t2} o2 q8 @u110 L1 e- d- >b<  d-
 34:
 35: /  STRINGS
 36: {t3} o3 q8 @u99 L1 'e-a-<d-' 'd-g-b' 'g-b<d-' 'e-a-<d-'
 37:
 38: /MELODY
 39: {t4}@63 o4 q8 v12 @u110@k0L8r4e-d-c4>b-<d-& d-d-d-cd-e-r4 r4g-g-g-4g-g- a-g-rd-4&(d-,e-)r4
 40: {t4}[!127@4810 o4 @u120L8r4e-d-c4>b-<d-& d-d-d-cd-e-r4 r4g-fg-4g-g- b-<d-re-4&(e-,d-)r4
 41: {t4}@4 @m o5 @u110L8r4d-e-g-a-rb-& b-a32b-16,b-4b-a-g-4 a-4g-e-2e-& e-g-a-g-32a-16,b-r4L8r4e-e-d32e-16,Ae-4d-& d-&d-2>b-<d-e-& e-4e-2.b-& b-a-g32a-16,g-a-g-4c-&
 42: {t4}LBe-&e-2>b-<d-e-& e-&e-2>b-<d-d32e-16,& e-&e-4f32g-..e-d->b-& b-a-g-d32e-..d->b-<c-&  e-1& e-1& e-1& e-1
 43:
 44: {t5}@63 o4 q8 v6 @u90  @k12L8r4e-d-c4>b-<d-& d-d-d-cd-e-r4 r4g-g-g-4g-g- a-g-rd-4&(d-,e-)r4
 45: {t5}[!127@4810 o4 @u120L8r4e-d-c4>b-<d-& d-d-d-cd-e-r4 r4g-fg-4g-g- b-<d-re-4&(e-,d-)r4
 46: {t5}@4 @m o5 @u99 L8r4d-e-g-a-rb-& b-a32b-16,b-4b-a-g-4 a-4g-e-2e-& e-g-a-g-32a-16,b-r4L8r4e-e-d32e-16,Ae-4d-& d-&d-2>b-<d-e-& e-4e-2.b-& b-a-g32a-16,g-a-g-4c-&
 47: {t5}LBe-&e-2>b-<d-e-& e-&e-2>b-<d-d32e-16,& e-&e-4f32g-..e-d->b-& b-a-g-d32e-..d->b-<c-&  e-1& e-1& e-1& e-1
 48:
 49: /drums
 50: {t7}   o2 q8 v16 @u99  L8c4r4ccr4 rcrcrcc4 c4r4ccrc rcr4rcr4
 51: {t8}   o2 q8     @u110 L8r4d4r4d4 r4d4r4d4 r4d4r4d4 r4d4r4d4
 52: {t9}   o2 q8     @u110 L8<c>+0>@u127[:12 r1 r2'd+d+r:]
 53: {t10}  o2 q8     @u40  L8f+f+f+f+f+f+f+f+ f+f+f+f+f+f+a+a+
 54: {t11}  o3 q8     @u70 L4g+
 55:
 56: {t1}[loop]
 57: {t2}[loop]
 58: {t3}[loop]
 59: {t4}[loop]
 60: {t5}[loop]
 61: {t7}[loop]
 62: {t8}[loop]
 63: {t9}[loop]
 64: {t10}[loop]
 65: {t11}[loop]
 66: {p}
```

リスト3

```
*************** M14_L3.ZMS ******************
  1: {T1}@X$F0,$41,$10,$42,$12,$40,$00,$7F,$00,$41,$F7
  2: {T1}@X$F0,$41,$10,$42,$12,$40,$01,$10,$06,$04,$02,$06,$03,$03,$00,$00
  3: {T1}@X$00,$00,$00,$00,$00,$00,$00,$00,$17,$F7
```

リスト4

```
*************** M14_L4.ZMS ******************
  1: /.comment An old picture. by Z.Nishikawa (for SCS5/SC155/CM300/CM500)
  2: {i}
  3: {b0}
  4: {m1,2000} {a midi1 , 1}
  5: {m2,2000} {a midi2 , 2}
  6: {m3,2000} {a midi3 , 3}
  7: {m4,2000} {a midi4 , 4}
  8: {m5,2000} {a midi5 , 5}
  9: {m7,2000} {a midi10, 7}
 10: {m8,2000} {a midi10, 8}
 11: {m9,2000} {a midi10, 9}
 12: {m10,2000}{a midi10,10}
 13: {m11,2000}{a midi10,11}
 14:
 15: /{o125}
 16: /.roland_exclusive $10,$42 ={$40,0,$7f,0}          / initialize SC55
 17: /.mc55_v_reserve={4,2,6,3, 3,0,0,0 ,0,6,0,0 ,0,0,0,0}/パーシャルリザーブ
 18:
 19: {t1}l125
 20: {T1}@X$F0,$41,$10,$42,$12,$40,$00,$7F,$00,$41,$F7 r16
 21: {T1}@X$F0,$41,$10,$42,$12,$40,$01,$10,$06,$04,$02,$06,$03,$03,$00,$00
 22: {T1}@X$00,$00,$00,$00,$00,$00,$00,$00,$17,$F7 r16
 23: {t1}@g12@1$41,$10,$42 @e70,70 @36 v10 @p40          |do|
 24: {t2}@g12@1$41,$10,$42 @e80,60 @36 v15               r8|do|
 25: {t3}@g12@1$41,$10,$42 @e70,70 @49 v10 @p99 @k-8r*19|do|
 26: {t4}@g12@1$41,$10,$42 @e80,80          @p54 @h36   r8|do|@m40
 27: {t5}@g12@1$41,$10,$42 @e80,80          @p74 @h36r*35|do|@m40
 28: {t7}@g12@1$41,$10,$42 @e80,10 @1                    @r1   r8|do|
 29: {t8}                                                @r1   r8|do|
 30: {t9}                                                  @r1   r8|do|
 31: {t10}                                               @r1   r8|do|
 32: {t11}                                               @r1   r8|do|
 33:             :
 34:             :
```

$7F,$00,$00にダミーデータを書き込む

例

共通コマンド

.ROLAND_EXCLUSIVE $10,$16= {$7F,$00,$00,$00}

MML

@I$41,$10,$16 X$7F,$00,$00,$00

●SC-55/SC-155/CM-300/CM-500のGS音源部

$40,$00,$7Fにダミーデータを書き込む

例

共通コマンド

.ROLAND_EXCLUSIVE $10,$42= {$40,$00,$7F,$00}

MML

@I$41,$10,$42 X$40,$00,$7F,$00

(SC-55は初期化するとベンドレンジが2半音範囲になってしまう点に注意すること)

14.1.3. BGMデータの改変具体例

ここでは，いままで解説してきた事柄を実際の音楽プログラムに対して行ってみることにしましょう。

リスト2のようなミュージックプログラムがあったとして，これをゲーム向きのデータへ修正していくとします。

まず，コンパイルをしてZMDを得ます。

A>zmusic -c LIST2

このZMDデータに対してZMD18.Rを実行しリダイレクトして共通コマンド部分をMMLに変換します。

A>zmd18 LIST2 > LIST3

リスト3のようなものが得られます。再びリスト2を読み込んで上から順番に共通コマンドをつぶしていきます。まずいちばん上のcomment文を'/'に置き換えて無効化します。そこから下に見ていくと，テンポコマンドがありますがこれを無効化しMMLに直してください。

その下のMIDI楽器個別コマンドを'/'で無効化します。ちょうどこれらの共通コマンドの働きと同じものがZMD18.Rによって生成されたリスト3です。

このリスト3をリスト2のMML部分の先頭に挿入します。前述のようにEOX($F7)のあとには多少のウェイトが必要ですから，休符'r16'を2つあるEOXの後ろにそれぞれ挿入します。このままでは1トラックだけr16×2=r8分演奏が遅れてしまうためほかのトラックに辻褄あわせを施してください。

こうしてできたのがリスト4で，これをコンパイルすれば共通コマンドなしのZMDの完成となります。

14.1.4. 演奏開始ルーチン

できた曲データ(ZMD)をゲームのメインプログラムから演奏するためには，Z-MUSICを呼び出すための簡単なマネージメントプログラム(つまりは演奏開始ルーチン)が必要です。さまざまな演奏開始ルーチンが考えられますがサンプルとしてリスト5を示しておきます。

この例では各演奏データがmusic0, music1, music2……とそれぞれ偶数番地に読み込まれているとします。そこでmusic0を演奏したいならばd0.1に0, music1を演奏したいならばd0.1に1を入れ，このルーチンplay_musicをサブルーチンコールするようにします。これで演奏が開始されます。

演奏データに埋め込まれているドライバのバージョンIDをa1レジスタが指標していなければならないというファンクション$11の仕様から，演奏データの格納アドレスa1に7を足しています。詳しくはファンクション$11の解説を参照してください。

14.1.5. 効果音データの作成と活用方法

ゲームのBGMに重ねて効果音を鳴らす場合はその演奏データのトラックとチャンネルのアサイン関係をしっかり把握していなければなりません。演奏データのプログラマと効果音データのプログラマが異なる場合には，データ仕様を事前に打ち合わせをしておくことをおすすめします。

リスト5

```
*******************  M14_L5.S  *******************
ISIC macro   func              *ドライバへのファンクションコール
     moveq.l func,d1
     trap    #3
     endm

y_music:
     *  < d0.l=曲番号(0~16383)
     movem.l d0-d2/a0-a1,-(sp)
     add.l   d0,d0
     add.l   d0,d0
     movea.l music_data_tbl(pc,d0.l),a1
     addq.w  #7,a1
     moveq.l #0,d2                          *高速応答モード
     Z_MUSIC #$11
     movem.l (sp)+,d0-d2/a0-a1
     rts

ic_data_tbl:
     dc.l    music0
     dc.l    music1
     dc.l    music2
     dc.l    music3
     dc.l    music4
     dc.l    music5
     dc.l    music6
     dc.l    music7
             :
             :
```

リスト6

```
*****************  M14_L6.S  ******************
(v12,0                          /音色データは事前に登録してある
/          AF  DM  WF  SY  SP PMD AMD PMS AMS PAN
           59, 15,  0,  0,  0,  0,  0,  0,  3,  0
/          AR  DR  SR  RR  SL  OL  KS  ML DT1 DT2 AME
           23,  8,  1,  8,  7, 31,  2,  1,  5,  3,  0
           30,  8,  8,  7,  5, 20,  1,  2,  1,  2,  0
           22,  3,  7,  8,  1, 20,  1,  1,  3,  0,  0
           27,  0,  0,  9,  0,  0,  1,  2,  5,  1,  0)


(1)
(m8,1000)
(a8,8)
(t8)@12 o3 q8 v14 p3 @k0 @m60 @h24 @n3 (a16<a)&a4.&(a16>a)
(p)
```

リスト7

```
**********  M14_L7.S  ********************
            この部分は不要なので切り捨て
           ┌──────────────────────────────┐
00000000  10 5A 6D 75 53 69 43 11 FF FF 00 01 00 00 00 02 .ZmuSIC.........
00000010  00 07 A0 0C B6 05 B3 D1 00 00 00 00 E6 00 3C E8 ...ｦ.ｳﾑ....ｦ.<ﾈ
00000020  00 18 FF FF D6 00 03 00 00 E0 39 00 0C FF FF 00 ....ﾖ.....9.....
00000030  00 00 40 00 01 45 48 FF E0 45 00 0C 00 0C 00 00 ..@..EH.｠E......
00000040  FF C0 00 FF FF                               .ﾀ...
```

リスト8

```
**********  M14_L8.S  ********************
        .dc.b   $00,$01                 演奏開始トラック数(.W)
        .dc.b   $00,$00,$00,$02         演奏データまでのオフセット(.L)
        .dc.b   $00,$07                 演奏絶対チャンネル(.W)
        .dc.b   $a0,$0c,$b6,$05,$b3,$d1,$00,$00                 演奏データ
        .dc.b   $00,$00,$e6,$00,$3c,$e8,$00,$18,$ff,$ff,$d6,$00,$03,$00,$00,$e0
        .dc.b   $39,$00,$0c,$ff,$ff,$00,$00,$00,$40,$00,$01,$45,$48,$ff,$e0,$45
        .dc.b   $00,$0c,$00,$0c,$00,$00,$ff,$c0,$00,$ff,$ff
```

リスト9

```
                **********  M14_L9.S  ********************
                (1)
                (m7,1000)
                (m8,1000)
                (a7,7)
                (a8,8)
                (t7)@7 o7 q8 v15 p3 @k0 @l1c<g>>cd<<e>(f8<f)p0
                (t8)@4 o7 q8 v12 p3 @k0 L32(c>c)&(c<e)r1
                (p)
```

リスト10

```
**********  M14_L10.S  *******************
        .dc.b   $00,$02                 演奏開始トラック数(.W)
        .dc.b   $00,$00,$00,$08         演奏データまでのオフセット(.L)
        .dc.b   $00,$06                 演奏絶対チャンネル(.W)
        .dc.b   $00,$00,$00,$29         演奏データまでのオフセット(.L)
        .dc.b   $00,$07                 演奏絶対チャンネル(.W)
        .dc.b   $a0,$07,$b6,$02,$b3,$d1,$00,$00                 0チャンネル用演奏データ
        .dc.b   $00,$00,$64,$01,$ff,$73,$01,$ff,$54,$01,$ff,$55,$01,$ff,$70,$01
        .dc.b   $ff,$e0,$65,$00,$18,$00,$18,$00,$00,$00,$20,$00,$01,$b0,$ff
        .dc.b   $a0                     7チャンネル用演奏データ
        .dc.b   $04,$b6,$0a,$b3,$d1,$00,$00,$00,$00,$e0,$60,$00,$06,$ff,$ff,$00
        .dc.b   $00,$ff,$80,$00,$ff,$e0,$54,$00,$06,$00,$06,$00,$00,$00,$aa,$ab
        .dc.b   $01,$80,$c0,$c0,$ff
```

●効果音データの制約

効果音データも基本的にはBGMデータと同じようにZMSファイルをコンパイルしたZMDで表現されます。しかし，BGMデータとの大きな違いは，

1) 共通コマンドを持てない
2) 効果音用の演奏トラックワークには初期値がない
3) 以下のMMLコマンドを使用したときの動作が保証されない
   - [] 系コマンド([DO]～[LOOP]は使用可)
   - 和音コマンド

となります。

効果音の演奏制御は，高速応答を要求されるのでパラメータの有効範囲チェックなどはいっさい行っていません。不当なパラメータで関連ファンクションをコールした場合は暴走する危険性があります。また，2)にもあるとおりワークの初期化も行わないためディチューンやボリュームなどの基本パラメータの初期値も不定です。よって必ず設定するようにしてください。

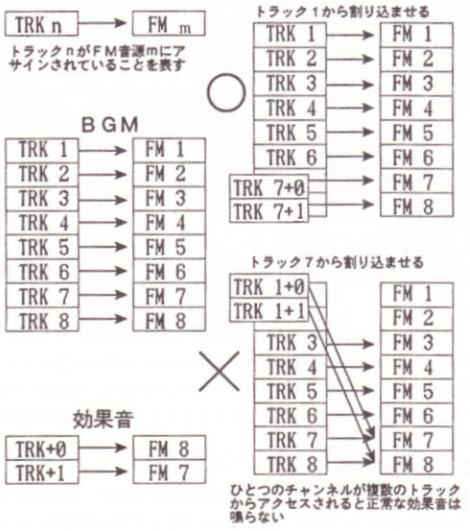
また，効果音にMMLのテンポコマンドを用いることは禁止していませんが，その効果音が鳴るたびにBGMのテンポが変化してしまうため使用しないほうがよいでしょう。

効果音データの実例をリスト6に示します。音色データはすでに述べたように事前に登録してあるものとします。これをコンパイル

リスト11

```
********** M14_L11.S **********
Z_MUSIC macro   func            *ドライバへのファンクションコール
        moveq.l func,d1
        trap    #3
        endm

se_play1:
        * < d0.l=sound number
        * - all
        movem.l d0-d2/a0-a1,-(sp)
        add.w   d0,d0
        move.w  se_tbl(pc,d0.w),d0
        lea     se_tbl(pc,d0.w),a1
        moveq.l #7,d2                   *割り込むトラック番号
        Z_MUSIC #$12
        movem.l (sp)+,d0-d2/a0-a1
        rts

se_play2:
        * < d0.l=sound number
        * < d2.l=ch number
        * - all
        movem.l d0-d2/a0-a1,-(sp)
        add.w   d0,d0
        move.w  se_tbl(pc,d0.w),d0
        lea     se_tbl(pc,d0.w),a1
        move.b  d2,7(a1)                *絶対チャンネルセット
        addq.b  #1,d2                   *割り込むトラック番号
        Z_MUSIC #$12
        movem.l (sp)+,d0-d2/a0-a1
        rts

se_play3:
        * < d0.l=sound number
        * - all
        movem.l d0-d2/a0-a1,-(sp)
        add.w   d0,d0
        move.w  se_tbl(pc,d0.w),d0
        lea     se_tbl(pc,d0.w),a1
        lea     which_ch(pc),a0
        eor1.b  #1,(a0)
        move.b  (a0),d2
        addq.b  #6,d2
        move.b  d2,7(a1)                *絶対チャンネルセット
        addq.b  #1,d2                   *割り込むトラック番号
        Z_MUSIC #$12
        movem.l (sp)+,d0-d2/a0-a1
        rts

which_ch:       dc.w    0

se_tbl:
        dc.w    se0-se_tbl              *$00
        dc.w    se1-se_tbl              *$01
        dc.w    se2-se_tbl              *$02
        dc.w    se3-se_tbl              *$03
        dc.w    se4-se_tbl              *$04
                :
                :

se0:
        .dc.b   $00,$01                 演奏総トラック数(.W)
        .dc.b   $00,$00,$00,$02         演奏データまでのオフセット(.L)
        .dc.b   $00,$07                 演奏絶対チャンネル(.W)
        .dc.b   $a0,$0c,$b6,$05,$b3,$d1,$00,$00         演奏データ
        .dc.b   $00,$00,$e6,$00,$3c,$e8,$00,$18,$ff,$ff,$d6,$00,$03,$00,$00,$e0
        .dc.b   $39,$00,$0c,$ff,$ff,$00,$00,$00,$40,$00,$01,$45,$48,$ff,$e0,$45
        .dc.b   $00,$0c,$00,$0c,$00,$00,$ff,$c0,$00,$ff,$ff
        .even

        .dc.b   $00,$02                 演奏総トラック数(.W)
        .dc.b   $00,$00,$00,$08         演奏データまでのオフセット(.L)
        .dc.b   $00,$06                 演奏絶対チャンネル(.W)
        .dc.b   $00,$00,$00,$29         演奏データまでのオフセット(.L)
        .dc.b   $00,$07                 演奏絶対チャンネル(.W)
        .dc.b   $a0,$07,$b6,$02,$b3,$d1,$00,$00         6チャンネル用演奏データ
        .dc.b   $00,$00,$64,$01,$ff,$73,$01,$ff,$54,$01,$ff,$56,$01,$ff,$70,$01
        .dc.b   $ff,$e0,$65,$00,$18,$00,$18,$00,$00,$00,$20,$00,$01,$b0,$ff
        .dc.b   $a0                     7チャンネル用演奏データ
        .dc.b   $04,$b6,$0a,$b3,$d1,$00,$00,$00,$00,$e0,$60,$00,$06,$ff,$ff,$00
        .dc.b   $00,$ff,$80,$00,$ff,$e0,$54,$00,$06,$00,$06,$00,$00,$00,$aa,$ab
        .dc.b   $01,$80,$c0,$c0,$ff
        .even
```

図1



して得たZMDをダンプデータに変換したものがリスト7です。ZMDにはヘッダ'ZmuSiC'などがつきますが，これはまったく不要であるため取ってしまいます(残しておいても構わないがメモリのムダになるので)。そしてリスト7の10バイト以降をアセンブラのソースプログラム形式に変換してやるとリスト8のようなものになります。

リスト9のように複数トラックの効果音データも作成可能です。これをコンパイルし同様に変換してやるとリスト10のようになります。

●効果音の演奏における2つの形式

効果音を鳴らす場合には2通りの形式が存在します。

ひとつは効果音が鳴るチャンネルをあらかじめ空けておく形式です。つまり効果音を鳴らすチャンネルをあらかじめ決めておいてBGMデータを作成するわけです。たとえばFM音源の1チャンネルから7チャンネルまでをBGM専用チャンネルとし，8チャンネル目を効果音チャンネルと割り当てる，といった場合です。

2つ目はBGMを演奏しているチャンネルに割り込んで効果音を演奏する形式です。たとえばFM音源の1から8チャンネルすべてのチャンネルでBGMを演奏し，そのうち8チャンネルが効果音に切り換わったりする，といった場合です。この場合，効果音が頻繁に演奏されると8チャンネル目はほとんど効果音しか鳴らなくなりBGMの1声が欠けて聞こえることになります。よってBGMの設計をする際に，効果音が被ってくるチャンネルにはメロディやベースなどの重要なパートを割り当てないようにすべきです。

●1番目の形式

リスト11は効果音のマネージメントプログラムの一例です（リスト11中のサンプル効果音は，それぞれリスト8，リスト10とまったく同じもの)。

se_play1を見てください。

トラック1から6までをFM音源1チャンネルから6チャンネルに割り当ててこれをBGMで使用し，FM音源7～8チャンネルを効果音として使用する場合を考えます。たとえばd0.1に効果音番号0を入れてse_play1を呼ぶと，トラック7番からFM音源第7チャンネルでse0のデータが演奏されます。se_play1中の，

```
moveq.l #7,d2
```

というのは，何番トラックで効果音を演奏するかを設定しています。いま，トラック1～6までBGMを演奏しているのでトラック7は未使用です。よってなんにもジャマされずごく普通に演奏がなされます。

なぜこの例で効果音se0がFM音源第7チャンネルで演奏されるかというと，それはse0の効果音データに演奏絶対チャンネル番号が6(0～7のうちの6)というのが埋め込まれているからです(囲み参照)。

同様の条件で，se1を鳴らす場合も同様です。se1は複数トラックによって構成された効果音データです。

```
moveq.l #7,d2
```

とあるためトラック7から演奏するのです。まず，se1の第1トラックはd2.1で指定したとおりトラック7で演奏されます。次にse1の第2トラックはd2.1で指定したトラック番号＋1のトラック8で演奏されます。

このように複数トラックで構成された効果音は，d2.1で指定したトラック番号から順番にトラックが割り当てられ演奏されます。

●2番目の形式

では今度はトラック1～8がFM音源1～8チャンネルにアサインされBGMが演奏されていたとして，これに効果音を鳴らす場合を考えましょう。こちらは前節で解説した2番目の形式です。

se_play1は結論からいえばこのままこの形式に対応できます。単一トラックのse0も複数トラック構成のse1にも対応できます。正しく効果音が鳴っている間は，そのチャンネルはBGM演奏を中断し，効果音を演奏し，演奏が終了するとなにごともなかったようにそのチャンネルのBGM演奏を再開するのです。

では，

```
moveq.l #7,d2
```

を，

```
moveq.l #1,d2
```

にしてse0を鳴らしたとしたらどうなるでしょうか。つまりse0がBGMのトラック1から割り込むという場合です。

確かにBGMのトラック1（FM音源第1チャンネル）の演奏が中断されますが，効果音se0はFM音源第7チャンネルで演奏されることになっているので，そのチャンネルで効果音を演奏しようとすると，BGM側のFM音源第7チャンネルにアサインされたトラック7がFM音源第7チャンネルをアクセスしてきます。いい換えると複数の違った演奏データを同時にたった1個のFM音源チャンネルで演奏しようする状況が起こるわけです。そこで効果音，BGM，ともに演奏が異常状態に陥ります。

se1は2本のトラックで構成された効果音ですがこの2本目のトラックがFM1チャンネルに割当てられていたとしましょう。つまりse1の上から5段目が$00,$07でなくて$00,$00となっている場合です。これを,

moveq.l #7,d2

で7トラックから割り込ませるとしましょう。

さて，実行されるとBGMのトラック7に，FM音源第7チャンネルにアサインされたse1の第1トラックが割り込んできます。以後FM第7チャンネルは効果音を演奏します。FM音源第7チャンネルにアサインされたBGM側のトラックは効果音に占領されているトラック7しかないので，どのトラックからも効果音演奏を妨害されずうまくいきます。

そして次にBGM側のトラック8に，FM音源第1チャンネルにアサインされたse1の第2トラックが割り込んできます。BGM側のトラック8はそれまでFM音源第8チャンネルで演奏していたBGMを中断しFM音源第1チャンネルで効果音を演奏し始めます。しかしその後，BGM側のトラック1がFM音源第1チャンネルをアクセスしてくるのでちょっとおかしなことになってきます。

つまりse1の第1トラックは正常に演奏されるが，第2トラックは正常に演奏されないのです。

つまりこの「効果音トラックがBGM演奏に割り込む」形式ではひとつ大前提があるのです。これから演奏する効果音のチャンネルが，割り込むトラックにアサインされているチャンネルと同一でなければなりません。この部分はたいへん重要です。

以上の話を図式化したものを図1に示します。

●その他の応用

効果音データはその演奏絶対チャンネルを持っていました。ですから同じ効果音を違ったチャンネルで鳴らすためには，その絶対チャンネルの違ったデータを別に持たなければならないのでしょうか。それでも構いませんが，メモリを浪費することになるので，効果音データのその絶対チャンネルが書かれている部分を書き換えてひとつの効果音データを流用したほうが賢明です。

リスト11のse_play2はその一例です。パラメータは効果音番号と演奏チャンネルの2つ。効果音データのアドレスを得たあと，そのアドレス+7に演奏させたい絶対チャンネルを書き込んでいます。効果音が複数トラックで構成されている場合は，たとえばse1ならば(効果音データのアドレス+7)のほかに(効果音データのアドレス+13)にも絶対チャンネルを設定しなければなりません。

se_play2では割り込むトラックはその絶対チャンネル+1の値にしています。これはBGMのトラック1～8がFM音源の第1チャンネルから第8チャンネル(絶対チャンネルにして0～7)にアサインされていることを前提としています。

se_play3はse_play2を少し改良したものです。これは効果音をFM音源第7チャンネルと第8チャンネルを交互に用いて鳴らす処理をつけてみたものです。こちらもBGMのトラック1～8がFM音源の第1チャンネルから第8チャンネル(絶対チャンネルにして0～7)にアサインされていることを前提としています。

こういった，状況に応じて効果音データを書き換えるような処理を工夫すると，さまざまな効果が実現可能になります。3Dのゲームならば効果音データのパンポットを，その3次元座標にあわせて書き換えて，疑似立体音響を実現させることも可能でしょう。

絶対チャンネル

内部表現の絶対チャンネルはベースチャンネルに無関係に操作対象デバイスを決定するものです。絶対チャンネル0～7がFM音源の1～8，絶対チャンネル8がADPCM1，絶対チャンネル9～24がMIDI1～16。そしてPCM8独立チャンネルモード時には絶対チャンネル25～31がADPCM2～8に対応します。

●MIDI対応曲にFMの効果音

これまではFM音源のBGMにFM音源の効果音を割り込ませる話でしたがMIDI対応の曲にFM音源の効果音を割り込ませるにはどうしたらよいのかを解説します。

BGMをMIDIで，効果音はFM音源で，という場合は各音源の目的がすでに区別されているので，効果音の使用チャンネルがBGMの使用チャンネルとかち合わないため，前述の形式1のように実現できます。たとえばトラック1～16までをMIDIでBGMに使用している場合なら，効果音は未使用トラックであるトラック17以降に割り込ませれば（割り込んではいないが）よいことになります。この方式はわかりやすく管理も簡単です。

では，FM音源＋MIDI楽器でBGMを，そしてFM音源で効果音を，という場合はどうでしょうか。複雑そうですがこれは結局FM音源のみのBGMにFM音源の効果音を割り込ませた方法がほとんどそのまま使えます。

たとえばFM音源の第1チャンネルから第6チャンネルにアサインしたトラック1～6，そしてMIDIにアサインしたトラック7～12でBGMを演奏していたとしましょう。つまりFM音源第7，8チャンネルは未使用というときに，この空いている2つのチャンネルを使って効果音を鳴らす場合を考えます。これはチャンネルがまったくかち合わないことから，未使用トラックで効果音を鳴らせばよいので前述の形式1がそのまま適用できます。よって効果音はトラック13以降へ割り込ませてやれば（割り込んではいないが）よいことになります。

最後にFM音源第1チャンネルから第8チャンネルまでをトラック1～8にアサインし，MIDIをトラック9～16にアサインし，これでBGM演奏して，これにFM音源の効果音を割り込ませる場合を考えます。FM音源の効果音はMIDIに対してなんの影響も与えないことからトラック9～16を除外して考えていいわけですので，そうなると話はまったく前述の形式2と同じになってきます。つまり，効果音をFM音源第8チャンネルで演奏するならトラック8に割り込ませればいいことになります。

●MIDIの効果音

MIDIで効果音を鳴らす場合にはもっと話が単純化します。ただMIDIチャンネルというものは単一チャンネルでもFM音源チャンネルでいう複数分として動作するので，ひとつのMIDIチャンネルに対して複数のトラックをアサインするケースが頻繁に出てきます。というわけで前述の形式1，2ともたいした区別はなくなるわけです。

しかし，管理の容易さから形式1を奨励します。つまり効果音専用のMIDIチャンネルを取っておいてこのチャンネルを用いて効果音を鳴らすようにするわけです。

●AD PCM音の効果音

AD PCMはPCM8.Xモードでないときは単音であるため形式2のみ有効です。

FM音源とAD PCMが連続するようなチャンネルアサイン構造に

リスト12

```
********** M14_L12.S **********

Z_MUSIC macro   func            *ドライバへのファンクションコール
        moveq.l func,d1
        trap    #3
        endm

ADPCM_se_play1:                                 *ADPCM効果音
        *
        *任意のアドレスに格納されたADPCM音を効果音として鳴らす
        *
        * < d0.l=効果音番号(0～255)
        *
        movem.l d0-d3/a0-a1,-(sp)
        add.w   d0,d0
        add.w   d0,d0
        move.w  d0,d1
        add.w   d0,d0
        add.w   d1,d0           *d0.w=d0.w×12
        movem.l adr_frq_tbl(pc,d0.w),d2-d3/a1
        Z_MUSIC #$13
        movem.l (sp)+,d0-d3/a0-a1
        rts

adr_frq_tbl:
        *       ADPCMデータのサイズ(.L)         *0
        *       優先レベル&PAN&FRQ(.L)  *4
        *       ADPCMデータの開始アドレス(.L)   *8
        *
        dc.l    2212,$0000_0203,adpcm_se0       *0
        dc.l    3478,$0001_0403,adpcm_se1       *1
        dc.l    5453,$0000_0301,adpcm_se2       *2
        dc.l    1401,$0001_0402,adpcm_se3       *3
        dc.l    8799,$0000_0403,adpcm_se4       *4
                        :
                        :

ADPCM_se_play2:
        *
        *Z-MUSICに登録したADPCM音を効果音として鳴らす
        *
        * < d0.l=効果音番号(0～255)
        *
        movem.l d0-d3/a0-a1,-(sp)
        move.l  d0,d2
        add.w   d0,d0
        add.w   d0,d0
        move.l  frq_pan_tbl(pc,d0.w),d3
        Z_MUSIC #$14
        movem.l (sp)+,d0-d3/a0-a1
        rts

frq_pan_tbl:                            *効果音番号
        *       優先レベル&PAN&FRQ(.L)
        dc.l    $0000_0203              *0
        dc.l    $0000_0303              *1
        dc.l    $0000_0201              *2
        dc.l    $0001_0402              *3
        dc.l    $0001_0203              *4
        dc.l    $0001_0103              *5
                        :
                        :
```

しておいて，FM音源とAD PCM音源を両方使った効果音というのも演奏できます。

また，複雑なものでないワンショットのAD PCMの効果音の演奏ならば専用のファンクションコール(ファンクション$13とファンクション$14)が設けてあるので，それらを用いたほうが高速応答が実現できます。リスト12にその一例を示します。まず，リスト中のADPCM se_play1を見てください。これは任意のアドレスに格納されているAD PCMデータをファンクションコール$13を使って，BGMに割り込ませて鳴らすためのサブルーチンです。BGM側でAD PCMをドラムなどで使用している場合はそれを一時停止して効果音を演奏します。効果音が鳴り終わると自動的にBGMの演奏に戻ります。

ADPCM_se_play1ではアドレスに配置されたAD PCMデータに便宜的に番号をつけ，呼び出すときにはd0.lにその番号を代入して呼び出すようになっています。その後ろにあるadr_frq_tblは各効果音AD PCMデータのサイズ，優先レベル(囲み参照)，データの開始アドレスの管理テーブルです。

ADPCM_se_play2はZ-MUSICにすでにZPDデータなどによって登録済みのAD PCM音を効果音で鳴らすためのサブルーチンです。ファンクションコール$14をそのまま使っただけの単純なものです。d0.1に指定する効果音番号はAD PCMコンフィギュレーションで設定したノートナンバーに相当します。たとえばコンフィギュレーションで，

12＝bomb.pcm

のように書き，これをZPDとしてZ-MUSICに登録してあれば，このbomb.pcmを効果音で鳴らすには，d0.1に12を入れればよいわけです。

後ろについているfrq_pan_tblというテーブルはADPCM_se_play1のadr_frq_tblからデータサイズとデータアドレスを取り去ったような書式をしています。

AD PCM効果音の優先レベル

X68000のAD PCMは(PCM8独立チャンネルモード時を除いては)単声です。ですから，効果音の発音要求は基本的に後着優先，つまり新しく発音要求されたものが発音されることになっています。しかし，ゲームなどで面と面とのつなぎのイベントの音声メッセージなど，どうしても爆発音などでかき消されたくない場合があります。そこでZ-MUSICでは非PCM8モードでは効果音を発音させる時に優先レベルというものを設定可能になっています。優先レベルは0～255まで設定可能ですが実用上は0，1程度で十分でしょう。

以下に優先レベルの例を示します。

発音中の優先レベル	発音要求の優先レベル	結果
0	0	発音要求採択
0	1	発音要求採択
1	0	発音要求却下
1	1	発音要求採択

14.2. 映像との同期

Z-MUSICには映像と音楽演奏を同期させるためのファンクションコールや機能が装備されています。ここではこの機能について解説を行います。

14.2.1. ファンクションコール$43を用いる

ファンクション$43で映像同期モードをオンにしてから演奏を開始すると，演奏開始からある時点までの絶対音長の合計値が演奏トラックワーク，

p_total(.l) $3C

に格納されていきます。これをユーザープログラム側で監視すれば，曲に合わせて画面などを切り換えることができます（たとえばオーケストラヒットが鳴るたびに絵を換えるなど)。

ちなみにワークp_totalの増加には以下の規則があります。

1) p_total()はMMLの[do]命令で初期化される。また，この時点までのp_totalの値を，

p_total_olp $C6

へ退避する

2) p_totalは同期モードオンを行ったときにのみ初期化される。'minit()'命令，(I)命令，ファンクション$00などでは初期化されない

3) p_totalは同期モードオンを行ったときから値に意味を持ち始める

具体的な使用プロセスは以下のようになります。

1) ファンクション$43で同期モードをオンにする

2) 曲の演奏を開始する

3) 演奏トラックワークp_totalを参照し(MEASURE13参照)，狙ったタイミングになるまで待つ

4) 狙ったタイミングになったならその仕事をする

5) やりたいことがすべて終わったらファンクション$43で同期モードをオフにする

演奏データのトラック1が4小節演奏されたら(合計ステップタイムにして192×4＝768)次の処理をするようなサンプルを以下に示します。

```
Z_MUSIC macro func   ＊ファンクションコールマクロ
        moveq.l func,d1
        trap    #3
        endm

        moveq.l #1,d2   ＊トラック番号＝1
        Z_MUSIC #$3c    ＊a0＝トラック1の演奏トラックワーク
                        ＊の先頭アドレス
wait_lp:                ＊4小節演奏するまでループ
        cmpi.l  #768,p_total(a0)
        bcs     wait_lp
        ＊次の処理へ
```

14.2.2. ユーザー開放ワークを使用する

演奏トラックワークの，

p_user(.b) $ff

は，ユーザーが自由に使用してよいワークとされています。これに対してMMLのワーク直接書き換えコマンド，

?a,d a:ワークオフセット(0～255) d:データ(0～255)

を用いて書き込みを行い監視プログラムとの通信を行い，映像の同期を行うことができます。このp_userは'm_init()'命令，(I)命令，ファンクション$00などでは初期化されず，値は不安定となるので，演奏データの冒頭でこれを初期化する必要があります。

具体的な使用プロセスは，

1) 演奏データは曲の先頭で

?$ff,0 (書き込む値は0でなくてもよい)

を実行してp_userを初期化する

2) ユーザープログラム側でp_userを監視し，希望の値が書き込まれるまで待つ

3) 演奏データは(目的のタイミングで)

?$ff,1 (書き込む値は1でなくてもよい)

を実行し，監視プログラム側に次の指示を促す

のようになります。

以下に演奏データのトラック1が，p_userに0以外の値を書き込むまで待ち，書き込みを確認後，次の処理へ移る，というようなサンプルを示します。

```
(i)
(m1,1000)(a1,1)
(t1)@1 o4 q8 v15 L4 ?$ff,0 cdef gab<c ?$ff,1 cdef gab<c
(p)                 ↑p_user_初期化    ↑p_userマーク

Z_MUSIC macro   func    ＊ファンクションコールマクロ
        moveq.l func,d1
        trap    #3
        endm

        moveq.l #1,d2   ＊トラック番号＝1
        Z_MUSIC #$3c    ＊a0＝トラック1の演奏トラックワーク
```

```
                    *の先頭アドレス
wait_lp:            *p_userが0以外になるまでループ
        tst.b   p_user(a0)
        beq     wait_lp
        *次の処理へ
```

MEASURE15
PCM8モード

PCM8.X（江藤啓氏作，以下PCM8）はX68000本体をいっさい改造せずに高度なソフトウェア処理でAD PCMの発声数を8倍に拡張してしまう画期的なアプリケーションです。ここではこのPCM8のZ-MUSIC上での使い方について解説します。

15.1. ZMUSIC.Xの組み込み

MEASURE1に示したようにZ-MUSICにはPCM8に対して2通りの対応の仕方を持っています。

●ポリモード

PCM8を先に組み込み，そのあとオプションスイッチ'−O'を添付してZMUSIC.Xを組み込みます。これで従来のAD PCM 1声のZMS/ZMD演奏データをPCM8を用いてポリフォニックに演奏するモードとなります。

従来の演奏データでは，たとえばシンバルを叩いたあと，このシンバルが鳴り終わらないうちにスネアを叩いたとすると，AD PCMの単音発声の制約からシンバルの音がブツリと切れてスネアの音に切り換わっていました。ここをブツ切りにしないで(PCM8.Xを用いて)と前後の音を重ねて演奏してしまおうというのがこのモードの趣旨です。

このモードはZ-MUSICではPCM8ポリモードと呼びます。

組み込み例

```
A>pcm8
A>zmusic -o
```

'-O'の後ろに数値を書くと8声あるAD PCMチャンネルのうち何チャンネルを音楽演奏に割り当てるかを設定できます。余ったチャンネルは効果音専用に扱われます。

組み込み例

```
A>pcm8
A>zmusic -o6
```

（2チャンネル(8-6=2)効果音用に使用する）

●独立チャンネルモード

PCM8を組み込み，そのあと通常どおりにZMUSIC.Xを組み込みます。これでPCM8が管理する疑似的な8つのAD PCMチャンネルを個別に使えるモードになります。

ハード的な制約によりパンポット(音場)は各チャンネル独立には機能しませんが，音量やキーオン/キーオフ，周波数(5段階)は完全に独立にコントロールできます。

組み込み例

```
A>pcm8
A>zmusic
```

15.2. 独立チャンネルモード

●チャンネルアサイン

AD PCMを指すチャンネル番号を異なったトラックにアサインすることによってそれぞれのトラックが順番にADPCM1チャンネル，ADPCM2チャンネル，…，ADPCM8チャンネルのように割り当てられていきます。

MUSICZ.FNCでは，たとえばトラック9〜16をADPCM1〜8チャンネルに割り当てたい場合は，

●m_ch("FM")のとき

```
for i=9 to 16
    m_alloc(i,1000)
    m_assign(9,i)
next
```

●m_ch("MIDI")のとき

```
for i=9 to 16
    m_alloc(i,1000)
    m_assign(25,i)
next
```

あるいは'm_assign2()'を用いて，

```
for i=9 to 16
    m_alloc(i,1000)
    m_assign2("ADPCM"+str$(i-8),i)
next
```

のようにします。

ZMS書式では，

●m_ch("FM")のとき

```
(m9,1000) (a9,9)
(m10,1000)(a9,10)
(m11,1000)(a9,11)
(m12,1000)(a9,12)
(m13,1000)(a9,13)
(m14,1000)(a9,14)
(m15,1000)(a9,15)
(m16,1000)(a9,16)
```

●m_ch("MIDI")のとき

```
(m9,1000) (a25,9)
(m10,1000)(a25,10)
(m11,1000)(a25,11)
(m12,1000)(a25,12)
(m13,1000)(a25,13)
(m14,1000)(a25,14)
(m15,1000)(a25,15)
(m16,1000)(a25,16)
```

または，

```
(m9,1000) (a ADPCM1,9)
(m10,1000)(a ADPCM2,10)
(m11,1000)(a ADPCM3,11)
(m12,1000)(a ADPCM4,12)
(m13,1000)(a ADPCM5,13)
(m14,1000)(a ADPCM6,14)
(m15,1000)(a ADPCM7,15)
(m16,1000)(a ADPCM8,16)
```

のように行います。

●PCM8独立チャンネルモードで使用解禁となるMML

従来AD PCMトラックで使用可能のMMLはすべて使用可能です。PCM8独立チャンネルモードでは従来ではAD PCMトラックでは使用が禁止されていたV，@Vのボリュームコマンドが使用可能になります。ボリュームは0〜16が有効範囲で原音量は9。@V使用時は1〜16へ換算されます。

●機能拡張されるMML

独立チャンネルモードでは周波数設定コマンドである@F命令は各トラック独立に機能するようになります。

また，PCM8が扱うことのできる新データ方式「16ビットPCMデータ」「8ビットPCMデータ」を扱うことができるようになります。

```
@F5   16ビットPCMデータ方式
@F6    8ビットPCMデータ方式
```

15.3. 注意

Z-MUSICの音楽演奏も割り込み処理，またPCM8の発音処理も割り込み処理です。よって10MHzのX68000だとお互いの割り込み処理がフル回転（？）したときには音楽のテンポに影響が出る場合があります。

MEASURE16
付録

Z-MUSIC用のC言語ライブラリについて解説します。

16.1.Z-MUSIC用C言語ライブラリの構成

Z-MUSIC用のC言語ライブラリは以下の3つのファイルで構成されています。

ZMUSIC.L　C言語用ライブラリ
　　(ソースリストファイル名はZMLIB.HAS)
ZMUSIC.H　C言語用インクルードファイル
ZMUSIC.DEF　BC.X用DEFファイル

ZMLIB.HASからZMUSIC.Lを生成するには,

A>HAS ZMLIB
A>LIB /U ZMUSIC.L ZMLIB.O

とする(HASはフリーソフトのハイスピードアセンブラ, LIBはC CompilerPRO-68Kに付属のライブラリアン)。

16.2.ZMUSIC.Hの利用法

自分のシステムのC言語関係のインクルードファイルが格納してあるディレクトリZMUSIC.Hをコピーします。

例

A>COPY ZMUSIC.H A:¥INCLUDE

ZMUSIC.XをC言語のプログラムから使いたい場合は, そのプログラムの冒頭に,

#INCLUDE　<ZMUSIC.H>

の1行をつけます。

なお, ZMUSIC.Hに含まれる関数の名前はMUSICZ.FNCとまったく同じです。各関数の詳しい仕様はZMUSIC.H本体, またはMEASURE3のMUSICZ.FNCの項を参照のこと。

基本的にX-BASIC用外部関数であるMUSICZ.FNCにコンパチに作られていますが, 一部の言語仕様の違いから互換でない関数もあります。X-BASICでは配列の要素数を関数側で自動認知することができますがC言語ではそれができません。そのため, MUSICZ.FNCではパラメータの個数などを省略できた関数がZMUSIC.Hでは省略できない, といった仕様変更のなされた関数があります(以下3つ)。

m_dirout()
m_exc()
m_roland()

またさらに, X-BASICではパラメータ個数を書かなくてもよかった関数もC言語では書かせるような仕様に変更されたものがあります(以下11個)。

sc55_reverb()
sc55_chorus()
sc55_part_setup()
sc55_drum_setup()
mt32_reverb()
mt32_part_setup()
mt32_drum_setup()
mt32_common()
mt32_patch()
mt32_partial()
u220_drum_inst()

MUSICZ.FNCにはなかったコマンドが新設されています。

m_wave_form2()
波形メモリ登録コマンド#2
(short int専用版/配列データ加工なし)

●ZMUSIC.L関数一覧

```
int   m_alloc( int track_no, int buffer_size );
int   m_assign( int channel_no, int track_no );
int   m_vget( int tone_no, char *data_ptr );
int   m_vset( int tone_no, char *data_ptr );
int   m_tempo( int tempo );     /* tempo=-1でリクエスト */
int   m_trk( int track_no, char *MML_ptr );
int   m_trk2( char *MML_ptr, int track_no1, int track_no2,
int track_no3 ,int track_no4, int track_no5, int track_no6, int
track_no7 ,int track_no8 );
          /* track_no?=0 または 'NASI' 以降を無視する。 */
int   m_free( int track_no );
int   m_play( int track_no1, int track_no2, int track_no3, int
track_no4, int track_no5, int track_no6, int track_no7, int track
no8, int track_no9, int track_no10 );
              /* track_no1=0 で全トラック演奏開始 */
              /* track_no?=0 または 'NASI' 以降を無視する*/
int   m_stat( int track_bit_pattern );
              /* track_bit_pattern=0 で全チャンネル検査 */
int   m_stop( int track_no1, int track_no2, int track_no3, int
track_no4, int track_no5, int track_no6, int track_no7, int track
no8, int track_no9, int track_no10 );
              /* track_no1=0で全トラック演奏停止 */
              /* track_no?=0または'NASI'以降を無視する */
int   m_cont( int track_no1, int track_no2, int track_no3, int
track_no4, int track_no5, int track_no6, int track_no7, int track
no8, int track_no9, int track_no10 );
              /* track_no1=0 で全トラック演奏継続 */
              /* track_no?=0または'NASI'以降を無視する */
void  m_init( void );
int   m_atoi( int track_no );
int   m_assign2( char *channel, int track_no );
void  m_ch( char *device );
int   m_pcmset( int note_no, char *filename,
      int pitch, int vol, int mix_note_no, int delay,
            int cut, int reverse, int fade_in_out );
      /* int pitch以降の引数を省略する場合は'NASI'とする */
void  m_pcmplay( int note_no, int pan, int freq );
void  m_rec( void );
void  m_rstop( void );
int   m_save( char *filename );
int   m_trans( char *filename );
int   m_fmvset( int tone_no, char *data_ptr );
int   m_out( int  d1, int  d2, int  d3, int  d4, int  d5,
           int  d6, int  d7, int  d8, int  d9, int d10 );
           /* d?=-1 以降を無視する */
int   m_dirout( char *adrs, int size ); /* sizeの省略不可 */
int   m_exc( char *adrs, int size );    /* sizeの省略不可 */
int   m_roland(int devID, int modelID, char *adrs,int size);
           /* sizeの省略不可 */
int   m_total( void );
int   m_fadeout( int speed );
int   m_pcmcnf( char *filename );
int   sc55_v_reserve( char *adrs, int devID ); /* size=16 */
int   sc55_reverb( char *adrs, int devID, int size );
                              /* size=7 */
int   sc55_chorus( char *adrs, int devID, int size );
                              /* size=8 */
int   sc55_part_setup( char part_no, char *adrs, int devID,
int size );                   /* size=119 */
int   sc55_drum_setup( char map_no, char note_no,
                    char *adrs, int devID, int size );
                              /* size=8 */
int   sc55_print( char *message, int devID );
                              /* size=32 */
int   sc55_display( int *pattern, int devID );
int   m_adpcm_block( char *filename );
int   mt32_p_reserve( char *adrs, int devID );
                              /* size=9 */
int   mt32_reverb( char *adrs, int devID, int size );
```

```
                                /* size=3 */
int   mt32_part_setup( char *adrs, int devID, int size );
                                /* size=9 */
int   mt32_drum_setup( char note_no, char *adrs, int devID,
int size );                     /* size=4 */
int   mt32_common( char timbre_no, char *timbre_name,
                    char *adrs, int devID, int size );
                                /* size=4 */
int   mt32_patch( char patch_no, char *adrs, int devID, int
size );                         /* size=7 */
int   mt32_partial( char timbre_no, char partial_no,
                     char *adrs, int devID, int size );
                                /* size=58 */
int   mt32_print( char *message, int devID );
int   m_print( char *message );
                                /* size=96 */
int   u220_setup( char *adrs, int devID );
                                /* size=7 */
int   u220_common( char *adrs, int devID );
                                /* size=17 */
int   u220_drum_setup( char *adrs, int devID );
                                /* size=7 */
int   u220_part_setup( char part_no, char *adrs, int devID);
                                /* size=13 */
int   u220_timbre( char timbre_no, char *timbre_name,
                    char *adrs, int devID );
                                /* size=26 */
int   u220_drum_inst( char note_no, char *adrs, int devID,
int size );                     /* size=20 */
int   u220_print( char *message, int devID );
                                /* size=12 */
int   m1_midi_ch( char *midi_ch_list );
                                /* size=8 */
int   m1_part_setup( char *track_param );
                                /* size=40 */
int   m1_effect_setup( char *effect_param );
                                /* size=25 */
int   m1_print( char *message );
                                /* size=10 */
int   send_to_m1( int devID );
int   zmd_play( char *filename );
void  m_debug( char mode );
int   m_count( char count );
int   fm_master( char volume );
int   m_mute( int ch_no1, int ch_no2, int ch_no3, int ch_no4,
              int ch_no5, int ch_no6, int ch_no7, int ch_no8,
              int ch_no9, int ch_no10 );
              /* ch_no1=0 で全トラック演奏継続 */
              /* ch_no?=0 または 'NASI' 以降を無視する */
int   m_solo( int ch_no1, int ch_no2, int ch_no3, int ch_no4,
              int ch_no5, int ch_no6, int ch_no7, int ch_no8,
              int ch_no9, int ch_no10 );
              /* ch_no1=0 で全トラック演奏継続 */
              /* ch_no?=0 または 'NASI' 以降を無視する */
int   m_wave_form(char wave_no,char loop_type,int loop_
                   point, int *wave_data, int size );
int   m_wave_form2(char wave_no,char loop_type,int loop_
                   point, short int *wave_data, int size);
int   sc55_init( int devID );
int   mt32_init( int devID );
void  adpcm_to_pcm(char *source,int size,int *destination);
              /* destinationの配列サイズはsourceの4倍必要
                 また, sizeはAD PCMデータの個数 */
void  pcm_to_adpcm(int *source,int size,char *destination);
                /* destinationの配列サイズはsourceの1/4倍必要
                   また, sizeはPCMデータの個数 */
void  exec_zms( char *zms_line );
int   m_inp(char inp_mode ); /* inp_mode≠0でループモード */
int   zm_ver( void );
int   m_trk2( char *MML_ptr,
      int track_no1, int track_no2, int track_no3 ,int track_no4,
      int track_no5, int track_no6, int track_no7 ,int track
      no8 );
      /* track_no?=0 または 'NASI' 以降を無視する。 */
int   zm_work( char trk_num, int work_offset );
      /* 演奏トラックワークの値をバイト単位で返す */
```

16.3.ZMUSIC.DEFの利用法

Z-MUSICを利用したX-BASICのプログラムをコンパイルするときに使用します。まず，自分のシステムの，BC.X(X-BASICをC言語へコンバートするプログラム，C言語PRO68Kに付属)が格納してあるディレクトリにZMUSIC.DEFをコピーしてください。

例

```
A>COPY ZMUSIC.DEF A:¥BC
```

そのディレクトリ内のBASIC.CNFというファイルを開き，

```
FUNC=ZMUSIC
```

の1行を加えてセーブします。BASIC.CNF中に，

```
FUNC=MUSIC
FUNC=MUSIC2
FUNC=MUSIC3
```

などが記述してある場合は，それらをすべて削除します(同名の関数が存在しているため，誤動作する可能性がある)。

16.4.ZMUSIC.Lの利用法

コンパイルするときにライブラリファイルとして利用するには，まず自分のシステムのライブラリファイルが格納してあるディレクトリにZMUSIC.Lをコピーしてください。

例

```
A>COPY ZMUSIC.L A:¥LIB
```

コンパイルしてできるオブジェクトにZMUSIC.Lをリンクし実行ファイルを得るには，

●ZMUSIC.Xを使用したX-BASICのプログラムをコンパイルする場合

```
A>CC ????.BAS ZMUSIC.L
```

●ZMUSIC.Xを使用したC言語のプログラムをコンパイルする場合

```
A>CC ????.C ZMUSIC.L
```

のようにします。

用語解説

A

AD PCM(ADDAPTIVE DIFFERENTIAL PULSE CODED MODULATION)
X68000の内蔵音源のひとつで，自然音をそのままデジタルレコーディングし，これを再生することができる。Z-MUSICではひとつの楽器音源としてコントロールが可能。

アフタータッチ(AFTER-TOUCH)
アフタータッチシーケンス(AFTER-TOUCH SEQUENCE)
鍵盤を押す強さを発音後変化させること。MIDI楽器によっては，音色のニュアンスの変化を変えたり，ビブラートやトレモロなどの効果を与えたりすることができる。Z-MUSICでは発音後，音長の1/8単位でアフタータッチ処理することができる。これをZ-MUSICではアフタータッチシーケンスと呼ぶ。Z-MUSICではFM音源に対してのアフタータッチもサポートしているが，この場合は音量の変化のみを制御することができる。(MML:@Z)

アルゴリズム(ALGORITHM)
FM音源は4つのオペレータ(発信器)で構成されているが，この各オペレータをどのように接続するかを決定するもの。X68000に内蔵されているFM音源チップOPMは8種類の「アルゴリズム」が指定できる。音色作成時にはこの8つから選ぶことになる。

アンプリチュードモジュレーション(AMPLITUDE MODULATION)
音量を周期的に変化させる音の表情付けの技法。トレモロともいわれる。Z-MUSICでは，4つのプリセット波形や，ユーザーが作成した波形にそって変化させたり，発音後，音長の1/8ごとに掛かり具合を変化させたりすることができる。Z-MUSICではこの機能はFM音源専用となっているが，MIDI楽器ではARCCをこの機能に使うことで実現が可能。(MML:@A)

ARCC(ASSIGNABLE REALTIME CONTROL CHANGE)
Z-MUSICにおけるMIDI楽器専用の特殊機能のひとつ。各トラックごとに，コントロールするコントロールチェンジを選択し，これを音長の1/8ごとや設定した波形などに沿ってこれをコントロールするもの。(MML:@A)

アルペジオ(ARPPEGGIO)
分散和音。和音を構成する1音1音を少しずつ時間をずらして演奏するもの。Z-MUSICでは和音コマンドやダンパーコマンドで表現が可能。

オートベンド(AUTO BEND)
オートピッチベンド(AUTO PITCH BEND)
Z-MUSICの特殊機能のひとつで，発音後，任意の高さまで音程をなめらかに変化させることができる機能。(MML:@B)

B

ベンド(BEND)→ピッチベンド(PITCH BEND)

C

キャリア(CARRIER)
FM音源のオペレータの名称で，主にエンベロープ/音量を決定する働きがある。

チャンネルプレッシャー(CHANNEL PRESSURE)
MIDIチャンネル単位のアフタータッチ。
→アフタータッチ

チェックサム(CHECK SUM)
ローランドエクスクルーシブの最後尾につく，データを正確に受信できたかどうかを確かめるために用いられる7ビットデータのこと。

コーラス(CHORUS)
エフェクタの一種で微妙に音程の違った音を重ねて音を厚くする。GM音源やローランドGS音源では，このエフェクトがサポートされている。

クロック(CLOCK)
絶対音長のこと。Z-MUSICはデフォルトでは4分音符＝48クロックで処理されている。

コントロールチェンジ(CONTROL CHANGE)
MIDI楽器には特殊な機能を手軽にコントロールするために定められたコマンド群があり，これをコントロールチェンジという。(MML:Y,@C)

D

ディレイ(DELAY)
ピッチモジュレーションをはじめとした，音に対して特殊な効果を与える際に，その効果が開始されるまでの遅延時間を決定するパラメータ。

デプス(DEPTH)→振幅

ディチューン(DETUNE)
音程を微調整するパラメータ。本来の音階から決定される周波数とは違った音程で演奏させて，コーラス効果を得たりする目的でよく用いられる。(MML:@K,@B)

デバイスID(DEVICE ID)
MIDI楽器に対してユーザーが設定した識別番号。

ダブルシャープ(DOUBLE SHARP)
1度(2半音)上げる変化記号(MML:＋＋)

ダブルフラット(DOUBLE FLAT)
1度(2半音)下げる変化記号(MML:--)

ダンパー (DAMPER)
ダンパーペダル (DAMPER PEDAL)
MIDIメッセージのひとつ。ダンパーオンにすると，以後発音した音はノートオフの情報を無視し，発音をダンパーオフとするまで鳴り続ける。ダンパーはMIDIチャンネルごとに設定可能。Z-MUSICではFM音源やAD PCMにも使用可能(MML:@D)。

E

エフェクト (EFFECT)
エフェクタ (EFFECTOR)
音に特殊な味付けをすること。この装置をエフェクタという。MIDI楽器には音源のほかにエフェクタを持ったものが多くある。Z-MUSICではGM/GS音源とローランドMT-32系のエフェクタを手軽に操作するための命令が備わっている。(MML:@E)

エンベロープ (ENVELOPE)
音量の時間的変化のこと。

エクスクルーシブメッセージ (EXCLUSIVE MESSAGE)
各MIDI楽器ごとに設定された特殊命令。楽器のメモリ内容を直接

いじるようなものも用意されている。(MML:@X,X)

エクスプレッション(EXPRESSION)
そのMIDIチャンネルの出力音量の割合を決定するコントロールチェンジのひとつ。コントロールチェンジ7番のボリュームとは別管理のため，最終的にはボリューム（0～127)×エクスプレッション(0～127）でそのチャンネルの出力音量が決定される。

F

フラット（FLAT)
半音下げる変化記号。(MML:-)

フェードイン（FADE IN)
無音状態からだんだんと音量を上げていくこと。(MML:/)

フェードアウト(FADE OUT)
無音状態へ向かってだんだんと音量を下げていくこと。(MML:¥)

G

ゲートタイム(GATE TIME)
ひとつの音符が実際に発音している時間。

逆転再生→リバース

GM(GENERAL MIDI)
MIDI協議会が承認した統一音源規格。ヤマハTG-100，ローランドSC-55シリーズなどがこの規格の代表的機種。

GS
ローランドが提唱したMIDI楽器の統一音源規格。GMレベル1にも適合している。SC-55やCM-300，JV-30などがこの規格の代表的機種。

H

波形(WAVE TYPE)
Z-MUSICではピッチモジュレーション，アンプリチュードモジュレーション，拡張ARCCに対して鋸歯状波，矩形波，三角波，鋸歯波シングル，ユーザーメイドの波形で制御が可能である。(MML:S)

波形メモリ(WAVE MEMORY)
ユーザーが各ポイントの任意の値を設定していき，非線形の波形を作ることができる。この機能を波形メモリといい，Z-MUSICではピッチモジュレーション，アンプリチュードモジュレーション，拡張ARCCをこの機能でコントロールすることができる。(ZMS:wave form,MUSICZ.FNC:m_wave_form)

波長
波形の1周期の長さ。Z-MUSICではモジュレーション波形であるプリセット波形の波長を変更可能である。(MML:@S)

ホールド(HOLD)→ダンパー(DAMPER)

I

インストルメント(INSTRUMENT)
楽器のこと。

K

拡張ピッチモジュレーション()
拡張ARCC()
MIDIトラックに対して有効な機能。FM音源トラックのように4つのプリセット波形とユーザー波形を用いたピッチモジュレーションやARCCが可能になる機能。(MML:M)

キーオン(KEY ON)→NOTE ON

キーオフ(KEY OFF)→NOTE OFF

キートランスポーズ(KEY TRANSPOSE)
音程を規定のものからずらすこと。同じような意味の言葉に「ディチューン(DETUNE)」がある。

M

メーカーID(MAKER ID)
MIDI楽器メーカーの識別番号。

MIDI(MUSICAL INSTRUMENT DIGITAL INTERFACE)
デジタル楽器の世界規格。メーカーや機種が違っていても，MIDI対応ならば同じコマンドメッセージで楽器をコントロールすることができる。

モデルID(MODEL ID)
MIDI楽器に対してメーカーが設定した楽器の識別番号。

モノ(MONO)
ひとつのMIDIチャンネルで一度に1音しか発音できないモード。トランペットなどのブラス系の音はこのモードにすると現実的。

モジュレータ(MODULATOR)
FM音源のオペレータの名称で入力された波形を変調し音素を作り出す働きがある。

モジュール(MODULE)→音源モジュール

マルチティンバー(MULTI TIMBRE)
1台の音源で一度に複数の楽器を演奏することができる機能のこと。いまや，たいていのMIDI楽器はマルチティンバーである。

N

NRPN(NON REGISTERED PARAMETER NUMBER)
MIDI規格では定義されていないそのMIDI楽器特有の機能パラメータ。

ノートオン(NOTE ON)
発音すること。

ノートオフ(NOTE OFF)
以前発音した音を消音すること。

O

オクターブ(OCTAVE)
ある音階から8度離れていることをいう。(MML:O)

オムニ(OMNI)
その設定された受信チャンネルとは無関係にすべてのチャンネル情報を認識するモードのこと。

音源モジュール
鍵盤のない音源のみで構成されたMIDI楽器のこと。コンピュータやシーケンサ，ほかのシンセサイザからMIDIによってコントロールすることによってのみ演奏を行うことができる。

音量モジュレーション→アンプリチュードモジュレーション(AMPLITUDE MODULATION)
音程モジュレーション→ピッチモジュレーション(PITCH MODULA

TION)

OPM
X68000に内蔵されているヤマハ製のFM音源チップ。4オペレータで構成され最大発音数8音。

P

パンポット(PANPOT)
発音する位置を設定するパラメータ。FM音源やAD PCM音源では1，2，3が左，中央，右に対応する。MIDIではコントロールチェンジの10番がパンポットの機能に相当する。パラメータの範囲は0～127までで0が最も左，64が中央，127がもっとも右に対応する。(MML:P,@P)

パーシャル(PARTIAL)
ローランドのシンセサイザ音源における音色を構成する発信源の名称。ひとつの音色が複数のパーシャルで構成されていることもある。

パーシャルリザーブ(PARTIAL RESERVE)
ローランドのマルチティンバー音源において，1パート内で保証されるパーシャル数を決定すること。一時的に音源に対して同時発音数を上回る発音要求があったとしてもパーシャルリザーブをしてあればその数のパーシャルは確実にそのパートの発音のために保証される。

ピッチモジュレーション(PITCH MODULATION)
音程を周期的に変化させる音の表情付けの技法。ビブラートともいわれる。Z-MUSICでは，4つのプリセット波形や，ユーザーが作成した波形にそって変化させたり，発音後，音長の1/8ごとに掛かり具合を変化させたりすることができる。(MML:@M)

ピッチベンド(PITCH BEND)
音程を滑らかに変化させること。Z-MUSICではパラメータを与えることでオートマチックにこの処理を行うことができる。(MML:@B)

ポリ(POLY)
ひとつのチャンネルで一度に複数の音を発音できるモード。

ポリフォニックプレッシャー(POLYPHONIC PRESSURE)
鍵盤単位で制御可能なアフタータッチ。
→アフタータッチ

ポルタメント(PORTAMENTO)
音程を滑らかに変化させること。(MML:(～))

プログラムチェンジ(PROGRAM CHANGE)
音色切り替えのこと。

Q

クオンタイズ
リアルタイムレコーディングされた演奏情報の細部を切り捨て整理すること。

R

リバース(REVERSE)
AD PCMデータをデータ末尾から先頭へ向かって再生すること。Z-MUSICではAD PCMコンフィギュレーションでこの指定が可能。

リバーブ(REVERB)
ソースの音に対して残響効果を与えて出力するエフェクタの一種。あたかもコンサートホールで演奏しているような効果を演出することができる。

RPN(REGISTERED PARAMETER NUMBER)
MIDI規格で定義されたMIDI楽器の拡張機能パラメータ。ピッチベンドの幅のような楽器の機種によらない一般的なパラメータが設定されている。

S

シャープ(SHARP)
半音上げるという変化記号。(MML:#)

振幅
ピッチモジュレーションなど，波形を用いて音に効果を与えるコマンドにおいて，その効果の掛かり具合を決定するパラメータ。(MML:@M,@A)

ステップタイム(STEP TIME)
ひとつの音符が演奏される時間。(MML:L,@L)

ソステヌート(SOSTENUTO)
速さを抑え気味に，音を一様に持続して演奏する奏法。

T

ティンバー(TIMBRE)
楽器(音色)のこと。

タイマ(TIMER)
Z-MUSICが音楽のテンポを管理するために使っているもの。X68000のOPMにはタイマAとタイマBが存在し，Z-MUSICではどちらか一方のタイマを用いてテンポを制御可能になっている。タイマBよりタイマAのほうがテンポ決定の計算の際の誤差が少ない。(MML:T,@T)

トレモロ(TREMORO)→アンプリチュードモジュレーション(AMPLITUDE MODULATION)

U

ユーザー波形→波形メモリ

V

ビブラート(VIBRATO)→ピッチモジュレーション(PITCH MODULATION)

W

ウェイト(WAIT)
処理が終わるまで待ち時間を設けること。MIDI楽器ではEOX($F7)受信後，受信内容の処理のために多少のウェイトを必要とする。MMLのXコマンドでエクスクルーシブメッセージを送信する場合は，このウェイト期間を考慮して，命令後に休符などのウェイトを設定したほうがよい。

Y

Yコマンド
音源のレジスタを直接書き換えるMML命令のこと。Z-MUSICではMIDIに対してのYコマンドはコントロールチェンジとして扱われる。

Z

絶対音長(ABSOLUTE LENGTH)
Z-MUSICが処理する音長カウンタのこと。デフォルトでは4分音符は絶対音長＝48となっている。(MML:L,@L)

Oh!X Books MUSIC SYSTEM #02
Z-MUSICシステムver.2.0

1993年12月16日 初版1刷発行
1994年2月16日 初版2刷発行

制作　Oh!X編集部
プログラム制作　西川善司
発行人　橋本五郎
発行所　ソフトバンク株式会社
〒103　東京都中央区日本橋浜町3-42-3
編集部　03(5642)8122
販売局　TEL　03(5642)8101
FAX　03(5641)3424
印刷　文唱堂印刷株式会社

Printed in Japan.

ISBN4-89052-460-6

MMLでの記号使用状況

!	デバッグコマンド
"	(BASICでの文字列指定)
#	嬰記号，共通コマンドヘッダ
$	16進数ヘッダ。[$]はセーニョ
%	2進数ヘッダ
&	タイ
'	和音指定
( )	ポルタメント
*	絶対音長ヘッダ。[*]はto coda
+	嬰記号
,	(セパレータ)
−	変記号
.	付点
/	ZMSでの注釈
0〜9	音長などの数値
:	繰り返し記号の一部
;	ZMDコードの埋め込み
A	音階名
@A	ARCC実行
B	音階名
@B	ベンド
C	音階名
@C	ARCC機能定義
D	音階名
@D	ダンパーON/OFF
E	音階名
@E	MIDIエフェクト制御
F	音階名
@F	AD PCM周波数設定
G	音階名
@G	ベンドレンジ設定
H	モジュレーション同期モード設定
@H	モジュレーションやARCCのディレイ
I	音色バンク切り換え
@I	MIDIデバイスID登録
J	強制再演奏
@J	MIDIのタイモード設定
K	キートランスポーズ
@K	ディチューン
L	デフォルト音長の設定
@L	デフォルト音長の設定（絶対音長）
M	モジュレーションモード
@M	モジュレーション
N	操作デバイス変更
@N	操作デバイス変更（絶対指定）
O	オクターブ指定
@O	ノイズモード
P	パンポット指定（左，中央，右）
@P	パンポット指定（多段階）
Q	発音時間指定（前から）
@Q	発音時間指定（後ろから）
R	休符
@R	ノートオフしない
S	モジュレーション波形設定
@S	モジュレーションスピード設定
T	テンポ設定
@T	割り込み周期設定
U	ベロシティ指定（リアルタイム）
@U	ベロシティ指定
V	音量指定
@V	音量指定（絶対音量）
W	同期待ち，同期信号発信
@W	状態保持
X	ローランドエクスクルーシブ送信
@X	MIDI生データ送信
Y	OPMレジスタ書き込み，MIDIコントロールチェンジ
@Y	MIDIのNRPN設定
Z	ベロシティシーケンス
@Z	アフタータッチシーケンス
[	特殊コマンド用括弧
¥	フェードイン/アウト指定
]	特殊コマンド用括弧
^	PC-9801式タイ
_	相対音量ダウン
`	強制キーオフ
a〜z	(大文字に同じ)
{	連符用括弧
|	繰り返し記号用
}	連符用括弧
~	相対音量アップ

ISBN4-89052-460-6

C0055 P4800E

9784890524600

定価4,800円
(本体4,660円)

1910055048003

郵便はがき

料金受取人払

日本橋局承認

1564

差出有効期間
平成7年5月
14日まで

1 0 3 - 0 0
1 6 1

(受取人)
東京都中央区
日本橋浜町3-42-3

ソフトバンク株式会社

Oh! 編集部行

電話

住所

フリガナ

氏名　　年齢

職業・勤務先
学校・学部・学年

Z-MUSICシステム ver.2.0

Z-MUSIC システム ver.2.0

弊社ソフトバンクの本をお買上げいただきありがとうございます。今後の編集の資料にさせていただきますので、下記のアンケートにお答え下さい。ご協力をお願いいたします。

■この本を何でお知りになりましたか？

1．雑誌広告（雑誌名　　）
2．雑誌の紹介記事で（雑誌名　　）
3．書店で見て　4．書店ですすめられて　5．人にすすめられて
6．その他（　　）

■この本をお買上げの書店名は？

都・道府・県　　区・市　　書店

■以下の質問にお答え下さい

内　容	1．わかりやすい	2．ふつう	3．わかりにくい
装　丁	1．よい	2．ふつう	3．わるい
価　格	1．高い	2．ふつう	3．安い

■お読みになった感想をお聞かせ下さい

■弊社発行の書籍・雑誌をお読みになったことがありますか？

書籍名（　　）　雑誌名（　　）

■今後どのような企画をお望みですか？


Oh!X BOOKS MUSIC SYSTEM #02 2HD
Z-MUSICシステム ver. 2.0
標準ドラムデータディスク disk 2
ソフトバンク出版事業部
Z. Nishikawa/SOFTBANK CORP. Oh!X
FUJIFILM
FLOPPY DISK


Oh!X BOOKS MUSIC SYSTEM #02 2HD
Z-MUSICシステム ver. 2.0
システムディスク disk 1
ソフトバンク出版事業部
Z. Nishikawa/SOFTBANK CORP. Oh!X
FUJIFILM
FLOPPY DISK


Oh!X BOOKS MUSIC SYSTEM #02 2HD
Z-MUSICシステム ver. 2.0
オーケストラヒットディスク disk 3
ソフトバンク出版事業部
Z. Nishikawa/SOFTBANK CORP. Oh!X
FUJIFILM
FLOPPY DISK


Oh!X BOOKS MUSIC SYSTEM #02 2HD
Z-MUSICシステム ver. 2.0
拡張リズムデータディスク disk 4
ソフトバンク出版事業部
Z. Nishikawa/SOFTBANK CORP. Oh!X
FUJIFILM
FLOPPY DISK


Oh!X BOOKS MUSIC SYSTEM #02 2HD
Z-MUSICシステム ver. 2.0
16ビットPCMディスク disk 5
ソフトバンク出版事業部
Z. Nishikawa/SOFTBANK CORP. Oh!X
FUJIFILM
FLOPPY DISK


Oh!X BOOKS MUSIC SYSTEM #02 2HD
Z-MUSICシステム ver. 2.0
効果音ディスク disk 6
ソフトバンク出版事業部
Z. Nishikawa/SOFTBANK CORP. Oh!X
FUJIFILM
FLOPPY DISK

Z-MUSIC システムver.2.0 Books

Oh!X編集部編

X68000用音楽ドライバZMUSIC.X ver.2.0
X-BASIC用外部関数/C言語用ライブラリ
AD PCMツール ZPCNV.X/ZPLK.X
Oh!X標準/拡張サンプリングデータ
綴じ込み付録　5"2HD6枚組

SOFT BANK

[注意]

・これらのディスクはコピーフリーです。あらかじめバックアップを取ってから使用してください。

・ディスク3のオーケストラヒットほかのデータは圧縮されていますので，LHA.Xを使って展開してください。

・ディスク1は自動起動です。使用されているZMUSIC.XはUNIVERSALバージョンです。RS-MIDI，POLYPHON MIDIを使用する方は各ZMUSIC.Xを作成して差し替えてご使用ください。

ISBN4-89052-460-6
C0055 P4800E
9784890524600
1910055048003
定価4,800円
（本体4,660円）

標準音楽ドライバ	ZMUSIC.X（ユニバーサルバージョン）
縮小版	ZMSC.X
	（RS-232C版，POLYPHON版への差分データ収録）
X-BASIC用外部関数	MUSICZ.FNC
C言語用ライブラリ	ZMUSIC.L/ZMUSIC.H/ZMUSIC.DEF
AD PCM多重化ドライバ	PCM8.X
演奏ツール	ZP.R
AD PCMデータ加工ツール	ZPCNV.R
AD PCMエディタ	ZVT.X
PCMデータリンカ	ZPLK.X
標準AD PCMサンプリングデータ	
16ビットサンプリングデータ	

SOFTBANK BOOKS

Z-MUSICシステム ver.2.0

SOFT BANK