上手な『草』の生やし方

for ppmck

MML 入力型ファミコン音源ドライバ

ppmck「w」コマンド活用読本

presented by

打込み魂

はじめに

このテキストは、ファミコン音源ドライバ[1]「ppmck」向けの MML「w」コマンドの一歩踏み込んだ使い方の解説本です。 …って、そもそも ppmck ってなんじゃらほい？

ppmck に至るまでの歴史を簡単に書きたいと思います。 日本の据え置きゲーム機の歴史を作ったといっても過言ではない、皆さんお馴染み任天堂の「ファミリーコンピューター」（以下「ファミコン」）の内蔵音源を鳴らして音楽を作るための一般向けドライバ＋MML コンパイラ[2]が、Izumi.氏の作成したドライバ「mck[3]」および Manbow-J 氏による MMK コンパイラ「mckc[4]」でした。 その後、某吉氏による機能拡張版「pmck」や、有志の手によってディスクシステム内蔵音源をはじめとするファミコンゲームの ROM カートリッジに内蔵されている各種拡張音源[5]の同時演奏の対応パッチ等が発表され、更にそれらのパッチとバグフィックス・機能追加を統合したものが、h7 氏による「ppmck[6]」です。 現在は、某吉氏の手によって、ppmck 最新版(release 9a)に更なる機能拡張を加えられた「ppmck 9a ex[7]」シリーズがリリースされています。

演奏環境も飛躍的に向上しました。 初代 mck が出た当初は、演奏用ハードを自作しなければファミコン実機演奏は困難でしたが､近年ではテラネットワークシステムさん[8]の NSF[9]プレイバックカートリッジ｢TNS-HFC」シリーズ[10]等を使い、気軽にファミコン実機演奏が可能となりました。 「mck Wiki」[11]に MML リファレンスマニュアルなどの各種情報やツール入手先等がございますので、興味ある方は是非ご覧下さい。

---

[1] ねじ回し…ではなく、チップ音源（ppmck の場合、ファミコン内臓チップ「RP2A03」の音楽演奏部分など）を制御するためのプログラムです。ppmck などは、MML コンパイラで MML を演奏用音源ドライバが認識可能なデータに変換し、演奏（＝音源制御）要プログラムと一緒に、演奏用ファイル(NSF:後述)に格納して出力します。 実機演奏ができるのは、ファミコン上で動作する演奏用プログラムが含まれているからです。

[2] 主に、人間が読める形式で書かれたプログラムやテキスト等を、機械や OS が実行できる形式に変換するプログラムを指します。 ppmck の場合は MML コンパイラ「ppmckc」を使って、演奏用 NSF ファイル(後述)を生成します。

[3] http://www.geocities.co.jp/Playtown-Denei/9628/
現在は各種機能拡張された「ppmck」が主流となっており、2003 年より更新が停止しています。 若干古いですが、その他ツールや資料等へのリンクもあります。

[4] 上記 mck と同ページにて配布されている「mck converter(mckc)」です。 この頃は、ドライバ本体とコンパイラが別開発だったんですね。

[5] ファミコンの場合、本体の RP2A03 という CPU に内蔵された音源（通称「内蔵音源」）の他に、ゲームカートリッジ等に音源チップ（正確には、音源つきカスタムチップ）を搭載することができ、ファミコン内臓音源とミキシングして出力することができました。 最初の拡張音源は、ディスクシステムの RAM アダプタに内蔵された波形メモリ音源 RP2C33（通常「FDS」）でした。 他に、各メーカーがこぞって使用しファミコン末期の音楽を支えた「MMC5」、悪魔城伝説等で有名になった「VRC6」、最初で最後のファミコン用 FM 音源「VRC7」(OPLL 下位互換)、ナムコのゲームに搭載された波形メモリ音源「N163」、ギミック！等、サンソフトのゲームに搭載された「SUNSOFT 5B」(SSG 互換音源)があります。 なお、最近の ppmck および各種プレイヤーでは、これらの音源を同時に演奏することも可能となってきました。（※一部音源の同時演奏には制限が出る場合もありますが、近年は技術革新によりそれも緩和されてきたようです）

[6] http://ppmck.web.fc2.com/ppmck.html
mck に、某吉氏の pmck および有志によるパッチを含むバグフィックスと複数の拡張音源同時演奏対応等を加えた、h7 氏による機能拡張版 mck。最新版は ppmck release9a。 動作が安定していることから、現在でも本バージョンを使っている方は多いようです。

[7] http://prgterrace.wiki.fc2.com/ （※左側メニューより「ppmck ex」をクリック）
某吉さんによる、ppmck の機能拡張版。 2015/5/17 時点の最新版は「ppmck release9a ex11.3」（ドキュメント等の表記は ex11.1）です。各種便利機能追加のほか、「エフェクトデータのバンク切り替え」と「実機演奏での処理限界オーバーによる演奏バグの抑制」が入っておりますので、ヘヴィなデータを作る方は是非 ex 版を適用したほうが良いと思います。

[8] http://www2s.biglobe.ne.jp/~tns/
初代 mck の頃より、ファミコン実機演奏用機材を発表し続けている同人ハードサークル。 筆者もいつもお世話になっております。

[9] NES(Nintendo Entertainment System=海外版ファミコン) Sound Format の略。 ファミコンの場合、音源ドライバと演奏用データファイルの統一規格として「NSF」が主流となっています。(※2015/5 現在) ドライバ＋音楽データの前に 128 バイトのヘッダ領域があり、ここで「アルバム名」「現在演奏中の曲番号」「使用音源」などの情報が格納されています。 これにより、NSF 対応のプレイヤーであれば、ある程度演奏に互換性があります。（但し、プレイヤー側のエミュレーション精度などにより、うまく演奏できない場合もあります）

[10] mck リリースの数年後に登場した、ファミコン実機演奏用カートリッジ。 現行の機種（※2015/05 現在）は「TNS-HFC5」で、三月兎さんで入手できます。(http://www.march-rabbit.jp/index.php?main_page=product_info&products_id=2950)

[11] http://wikiwiki.jp/mck/
(pp)mck 関連の技術的情報は、ほぼここに集約されているといっても過言ではありません。

MML とは？

Music Macro Language の略…すなわち「音楽記述用言語」です。　80 年台のホビーパソコンには、内蔵 ROM-BASIC 等に「PLAY 文」というものがついていたのをご存じの方も多いかと思います。　これこそが MML です！　近年の DAW[1]などとは異なり、お好きなテキストエディタ（Windows の「メモ帳」でも OK！）と MML コンパイラさえあれば、すぐ音楽演奏用ファイルが出来てしまうのが最大の魅力です。　このため、**比較的ロースペックのパソコンでも演奏用データ作成が可能です。**　（但し、PC 上で演奏する場合、専用プレイヤー[2]、もしくは汎用音楽プレイヤーソフト(Winamp[3], foobar2000[4]等)と演奏用プラグインが別途必要です）

MML は、演奏対象チャンネルに続いて、「音階」（もしくは休符）と「音長」を指定するのが基本です。　他に、基本パラメータとして、**「テンポ（演奏する速さ）」「デフォルト音長（音長を省略した時の発音の長さ）」「オクターヴ」「音量」「音色」**などの指定があります。　ppmck ですと、例えば以下のようにして演奏します。

```
A       o4v15@0a2
```

この例では、A チャンネル（内蔵音源の矩形波チャンネル１）で、オクターブ 4 (o4)の最大音量(v15)、音色 0（@0）でラ(a)の音を２分音符(2)で鳴らせ、となります。　このあたりは、どの音源ドライバでも記法は概ね同じです。（対象チャンネルの表記方法だけは、音源ドライバや対象音源によって異なります）　この書き方を基本とし、各音源ドライバ毎に多彩な表現を行うための機能が拡張されている（俗にいう「MML の方言[5]」）わけです。

ちなみに、ファミコン音源を操作する環境は、MML による音源ドライバの他、**作成ツールが一体となっている「トラッカー[6]」や、MIDNES[7]のように、ファミコン音源を MIDI で直接駆動できる装置もあります。**　また、実機演奏に拘らなければ、**VSTi 等でファミコンの音[8]を使って DAW で曲を作るという選択肢もあります。**　昔と違い、MML しか選択肢の無い状況ではありませんので、**「MML は面倒だから嫌だー！！！」**という方は、MML 以外の方法を試してみるのも良いと思います。

…以降の文章は**「せっかくだから俺は MML を選ぶぜ！」**という茨の道(?)をうっかり選んでしまった**業の深い**方向けの情報です。

---

[1] Digital Audio Workstation の略。　個人向け音楽制作環境として、主に MIDI 音源モジュールの制御等に特化したものは以前から「DTM(DeskTopMusic)」と呼ばれていますが、近年は MIDI モジュールの音楽演奏制御のみならず、VSTi 等の PCM 発音によるオーディオトラックの制御やプラグインによる各種エフェクト、更には映像との同期で PV まで作れるものもあり、「これさえあれば音楽はすべて作れる」という、まさに Workstation の名前に恥じない環境が個人でも手に入る時代となりました。

[2] 筆者は「VirtuaNSF」を愛用しています。　キー操作が若干特殊ですが、特に内蔵音源は実機に非常に近い音が鳴ります。
http://wikiwiki.jp/mck/?NSF%A5%D7%A5%EC%A5%A4%A5%E4%A1%BC#u80992c5

[3] http://www.winamp.com/
プラグインを追加することで、各種音楽ファイルの演奏に対応する老舗のフリーウェア。　NSF 演奏プラグインだけでも膨大な量があり、それぞれ音質や機能が異なります。　いろいろ試してみて、自分好みのプラグインを見つけましょう。

[4] http://www.foobar2000.org/
Winamp と同様、プラグインで各種音楽ファイルの演奏に対応するフリーウェア。　動作が軽いのとカスタマイズ性の高さで、玄人好みのプレイヤーのようです。（筆者はあまり使ったことが無いのですが…）

[5] 基本 MML でも、音源ドライバや機種によって表記が異なるものも存在します。　代表的なものがオクターヴ切り替えコマンドで、「>」で１オクターヴ上がるドライバと、下がるドライバの両方が存在します。　ppmck は通常「>」でオクターヴが上がる動作となるのですが、#OCTAVE-REV 宣言を MML の最初に書いておくことで、挙動を逆転させることも可能です。

[6] 海外ではスタンダードな音楽制作環境。　日本では FamiTracker がよく使われるようです。 http://wikiwiki.jp/mck/?FamiTracker

[7] MIDI 信号を使って、ファミコン内蔵音源を操作できる NES（海外ファミコン）用カートリッジ。 http://dic.nicovideo.jp/a/midines

[8] ファミコン音楽制作ユニット「YMCK」の Yokemura 氏による「Magical 8bit Plug」（http://www.ymck.net/download/）や、コナミ所属コンポーザー村井聖夜氏による「ファミシンセ」（http://www.geocities.jp/mu_station/）等が有名。

草って…なに？

さて、やっと本題です。　この本の表題は**「上手な『草』の生やし方」**ですが、**『草』とは何でしょうか…？**

…答えは「w」です…って、**知らない方には何のことだかサッパリわかりませんねw**

「w」とはネットスラング[1]の一種です。　「(笑)」が簡略化され、「わらい」をローマ字入力する際の最初の文字「w」だけが残ったものです。　大笑いを表す際に「www」などと複数の「w」が並んださまを、俗に**「草を生やす」**といいます…なるほど、**見た目通りですねw**

…で、ppmck の MML には、そのものズバリ「w」コマンドというものがあります！
すなわち、**上手な『草』の生やし方＝「w」コマンドの活用法**、というわけです。

では、「w」コマンドとはいったい何なのでしょう…？
初代 mck の「mckc.txt[2]」には、「w」コマンドの説明として以下のように記載されています。

> ・ウェイト
> w<len>
> **指定の時間だけ前回のコマンドを保持します。**<len>は音長です。
> <len>を省略した場合は、l コマンドの値を使用します。

「指定の時間だけ前回のコマンドを保持します。」とあります。　何なんでしょう…ぱっと見、**何のことだかサッパリ判りませんw**　実際、「w」コマンドを全く使ったことが無い方もおられると思います。　しかし、この表現で何のことだか一瞬で理解できた方は、MML を使い慣れた相当な手練れの方だと思います。

ここから先は、**この一文に秘められた無限の可能性を、ほんの少しだけ紐解いていきましょう。**
この本を読み終わる頃には、この文章の意味と「w」コマンドの体感することができるでしょう！（希望）

「w」コマンドを使いこなすことによって、ppmck の表現力は無限に増大するのです！
知らないのは勿体ない！！　是非使いこなして、より素晴らしい ppmck ライフを！！！

…というのが、この本の目的なのです。

注意！

・w コマンドの挙動は、ppmck のバージョンによって若干異なります。
　本書は、「ppmck release 9a ex11.3」をベースに記載しています。
・ppmck 以外の音源ドライバでも、類似の実装があるドライバが存在します。
　但し、ppmck 以外のドライバにて同様の動作を保証するものではありません。
　（むしろ、各自でご確認のうえ資料として後世に伝えると、大勢の方が喜ぶと思います）

---

[1] 主にインターネット（古くはパソコン通信）上の文字コミュニケーションでのみ使われる独特な言い回し。　言葉毎に生い立ちが若干異なるようですが、「w」の場合は本文中にあるとおり、省略系として発展した部類だと思います。

[2] w コマンドは、なにげに mck 初代から実装されている基本コマンドなのです。　このコマンドが実装されていない MML 表記型音源ドライバも多い中、デフォルトで実装されているのは後発となる MML ドライバの強みかもしれません。(NSD.lib 等が出た今では mck もすっかり古参ですが…)

Level1：とりあえず使ってみよう！

最も簡単な使い方は、概ねどの MML にも実装されている「タイ[1]」コマンド（ppmck の場合は「&」）の代替としてです。　具体的には、以下の２つの MML は、同じ音が鳴ります。　下線部分が表記上の差分です。

```
A       o4q8v15@0 a2&a4r4

A       o4q8v15@0 a2w4r4
```

この記法の利点は、なんといっても**「&を打つ必要が無い」ということに尽きますw**　「&」を入力するには、（Windows の標準キーボードの場合）Shift キーを押しながら「6」を押して…と、ちょいと面倒なのですが、「w」コマンドを知っていると、それこそ流れるように MML を入力することができます。　欠点を挙げるとすれば、「w」コマンドを知らない方が MML を見ると、全く意味が判らないこと、でしょうか…

なお、この記法で注意すべき点が１つあります。　**「q」「@q」コマンドで、発音時間を 100%（＝「q8」「@q0」）「以外」に指定した場合、タイとは鳴り方が変わってしまいます。**　以下は先程とほぼ同様ですが、q8 ではなく「q7」となっているのがポイントです。

```
A       o4q7v15@0 a2&a4r4

A       o4q7v15@0 a2w4r4
```

前者は「&a4」の部分まで鳴るのに対し、**後者は「a2」の途中で音が途切れてしまいます。**　これが、タイ(&)コマンドと w コマンドの決定的な違いです。　タイコマンドは、「q」「@q」コマンドが設定されていても、「&」の次の音まで繋いで鳴らすのに対し、「w」コマンドは、直前の音にタイコマンドが無いため、**直前音の途中で KeyOff[2] し、「KeyOff の状態（＝発音無し）を継続する」**という解釈になるためです。

これこそが、前章にて引用した MML マニュアルの文言

> **指定の時間だけ前回のコマンドを保持します。**

が示す真の意味なのです！

ここは意外と引っかかる部分なので、w コマンドを使っていて「鳴りが変だな」と思ったら、真っ先にチェックしましょう。

---

[1] 国の名前…ではなく、（音を）結ぶもの（tie）です。　ネクタイの「タイ」と同じ語源ですね。

[2] 音符本体の発音時間を終わらせ、リリース（余韻）に入ることを指します。　Key＝鍵盤を離す（＝Off）タイミング、とご理解戴けると判りやすいかと思います。

Level2：人間くさい演奏をシミュレートしてみよう！

ppmck には、通常の音長指定（4 分音符なら「4」、8 分音符なら「8」など）の他、**「フレーム指定」という音長指定も可能です。**[1] これは、楽譜のような音長指定とは異なり、音源ドライバの内部処理単位での指定となります。 ppmck の場合、1 フレームは 1/60 秒[2]です。 **w コマンドをフレーム指定と共に使いこなすことによって、微妙なテンポの揺らぎなどを表現することができます。** 例として、「rit.（リタルダンド＝徐々にゆっくりと）」を表現してみましょう。 以下のように、**元の発音に w コマンドで差分の音長を徐々に長くしながら加えていけば良い**のです。

```
A       t15018o4q8v15@0 c dw#2 ew#4 fw#6 gw#8 aw#10 bw#12 >cw#14
```

音階が上がっていくに連れて、徐々に発音が長く（テンポが遅く）なってくるのが判ると思います。 **「わざわざ w コマンド使わなくても、テンポを変えればいいじゃん！」**…と思うかもしれません…が！ ppmck の処理単位は 1 フレーム＝1/60 秒であるが故に、細かいテンポ指定は実は苦手[3]なのです。 若干面倒ではありますが、全チャンネルを w コマンドの発音でフレーム数まで厳密に合わせることによって、微妙なテンポの揺らぎを伴ったよりきめ細かい表現を行いつつ、テンポずれの起きない確実な発音にすることが可能です。

上記の応用編として、「前の音を少し長くし、次音の発音開始タイミングを少し遅らせる（いわゆる「タメ」を作る）」パターンもあります。

```
A       t15014o4q8v15@0 c d ew#6 f~#6 g
```

微妙な違いですが、お判りいただけたでしょうか？ ドとレの発音時間が同じであるのに対して、「ミ」の音はこれらよりも少し長め、「ファ」の音が少し短めとなっています。 **ミ(e)の発音後に w コマンドで 6 フレーム持続する代わりに、ファ(f)の音長から 6 フレーム短く[4]しています。** こうすることで、**全体の尺としては帳尻を合わせつつ、ミとファの発音時間を変えています。** このテクニックは、知っているのと知らないのでは聴感に大きな違いが出ます。 ここはタメで鳴らしたい…という「ここぞ！」の時に、是非使ってみてください。

---

[1] 音長指定の前に「#」をつけるとフレーム音長指定となります。 例えば、ラを 3 フレーム鳴らす場合は「a#3」となります。

[2] 昔のゲーム機等は、ディスプレイ描画の垂直同期信号（VSYNC）毎に、割り込み処理（俗にいう「タイマ割り込み」）を発生させることが可能で、音楽演奏など、タイミングを一定に保つ必要がある処理で使われていました。 通常は 1/60 秒単位で、これを「1 フレーム」と呼びます。

[3] 1/60 秒単位ですと、通常よく使う音符でもテンポ指定によっては割り切れない音長になってしまいます。 ファミコンのような 1/60 秒＝1 フレームの音源ドライバの場合、テンポが 150 か 180 だと（変な音長を指定しない限り概ね）ずれません。 例えば、t150 の場合、4 分音符は 24 フレームです。 8 分音符なら半分の 12 フレーム、16 分音符なら 6 フレーム、32 分音符なら 3 フレーム…となり、全て整数となるのでずれません。 t180 の場合、4,8,16,32 分音符の順に、20,10,5,2.5 フレームとなります。 32 分音符は整数で割り切れないフレーム数のため、テンポがずれる場合もあります。（ただ、最近の ppmck では自動補正をかけるため、あまりずれないようです）

[4] ppmck のコマンドでは、音階＋発音時間の後に「~#フレーム数」を指定することで、発音時間を指定フレーム数分、本来の発音時間よりも減らすことができます。 なお、「~」は「-」とも記載できますが、音階のフラットを表す「-」（半音下げる）と区別がつかない可能性がありますので、「~」を使ったほうが間違いありません。

Level3：発音途中で表情を変えてみよう！

さて、ここからが「w」コマンドの真骨頂です。

「何故、タイコマンド以外にわざわざ w コマンドがあるのか？」

…その答えがこちらです！　じゃじゃ～ん！！！(※めくり効果音)

ppmck の場合、タイコマンド（&）の直後には音階しか置けません[1]が、「w」コマンドの直前には、さまざまな MML コマンドが置けるのです！

「w」コマンドの直前にコマンドが置けるということは、何を指すのか…？　具体的には、以下のようなことができます。

1．発音途中の任意のタイミングでピッチベンド(EP)を開始・終了する
2．発音途中の任意のタイミングで LFO をかける・止める・種類を変える
3．発音途中の任意のタイミングで音量エンベロープを変える
4．発音途中の任意のタイミングで音色を変える

等々…上記に限らず、他にも様々なことが出来たり出来なかったりするわけですが、今回はこの 4 つに絞って書いていきます。

1. 発音途中の任意のタイミングでピッチベンド(EP)を開始･終了する

個人的には、**w コマンドの使い方としては最も有用だと思います。**　例えば、トーンの終盤でベンドダウン、あるいはヴォーカルの「しゃくり」を表現するのに、トーンの最後に急激なベンドアップ…など、使い方は多岐に及びます。　例として、後者（トーンの最後に急激なベンドアップ）を以下に示します。

```
@EP0 = { 25 }
A       t150o4q8@0v15 EP0Fa2~#5EP0w#5
```

音の最後にしゃくりあげるニュアンスが出ていると思いますが、如何でしょうか。　2 分音符の最後の 5 フレームだけ、EP0 の設定によって 1 フレーム毎にピッチを 25 づつ上げています。　**ピッチベンドを自由自在に操れるようになると、ヴォーカル表現やギターソロのピッチベンド、シンセサイザーのポルタメント表現など、一気に表現の幅が広がります。**

w コマンドを活用し、今まで諦めていた表現にも是非チャレンジしてみてください！

[1] ppmck 以外の音源ドライバの一部では、タイコマンドの直後に音階以外のコマンドを置いても認識されるものがあるようです。　即ち、わざわざ草(w)を生やさなくても、これから書くことと同様のことが実現できる（かもしれない）ということですね。　細かいウラは取ってませんが、興味のある方は、各自の使い慣れた音源ドライバでご確認ください。　ノウハウ自体は、本書の内容を流用できると思います。

2. 発音途中の任意のタイミングで LFO[1]をかける･止める･種類を変える

任意のタイミングで遅延ビブラートをかけたい場合などに使います。 例えば、以下の例ではビブラート無しで２分音符発音後にビブラートを開始し、そこから更に２分音符発音後、ビブラートを無効化しています。

```
@MP0 = { 0, 2, 10, 0 }
A      t150o5@0v15 MPOFc2MP0w2MPOFw2
```

MPo の定義で、ディレイフレーム数（最初の数値）を o に設定しているのもポイントです。 最初から MPo を指定した場合は、ディレイなしで発音開始時からビブラートがかかりますが、**発音開始（KeyOn）時は MPOF でビブラートを無効化し、２分音符発音後に MP0 でビブラートを開始しています。**（サンプル MML では、比較のために更にその後 MPOF でビブラートを停止して発音しています） これにより、任意のタイミングでのビブラート開始が可能のみならず、**フレーム数指定ではないビブラート開始タイミング指定（＝２分音符、４分音符等の譜面音長指定）が可能となることも大きな利点です。** ビブラートを自由に使いこなすには、是非覚えておきたいテクニックです。

3. 発音途中の任意のタイミングで音量エンベロープを変える

これは、**発音によってリリースエンベロープ[2]を変えたり、複数のソフトエンベロープ[3]を途中で切り替える場合に使います。** 前者の、通常の KeyOff(@vr＋k コマンド[4])の減衰音よりも残響音を長くする例を以下に示します。

```
@v0 = { 12, 9, 6, 3, 0 }
@v1 = { 14, 14, 13, 13, 12, 12, 11, 11, 10, 10, 9, 9, 8, 8, 7, 7, 6, 6, 5 }
A      l4o4@0q8v15@vr0 ck2 c@v1w2
```

１音目の減衰時間(@vr0=@v0)よりも、２音目の減衰時間(@v1)のほうがより長いことがお判りいただけるかと思います。 ここで挙げたのは単純な例ですが、**アタック、持続、リリース…と音量変化パターンをいくつも作っておいて、それぞれをパーツとして w コマンドで変化させながら繋ぎ合わせる…といったことも可能です。** 自由かつダイナミックな音量変化を実現するためには、是非覚えておきたいテクニックの１つです。

---

[1] シンセサイザー等で使われる Low Frequency Oscillator（低周波発振装置）の略。 発音に対して LFO をかけることで、音に揺らぎ効果を出します。 ビブラート（音程を細かく上下させる）のことを「音程 LFO」と呼ぶ場合もあります。（ちなみにトレモロは「音量 LFO」）

[2] ppmck の場合、「@vr」コマンドで、KeyOff（発音停止）からのソフトウェアエンベロープ（後述）を指定できます。 これにより、KeyOff 後の自然な残響音を表現できるようになっています。 SD コマンド等を組み合わせることで KeyOff 時に数音前の音階を@vr で指定したソフトエンベロープで鳴らす（＝1ch ディレイ）ことも可能です。

[3] 音源チップがハードウェアで持つ音量変化機能（＝ハードエンベロープ）の対比として、ユーザが任意に指定できる音量変化機能を指します。ppmck の場合は「@v」コマンドでソフトエンベロープを定義もしくは指定します。

[4] ppmck は、KeyOff 後にユーザが指定した別のソフトエンベロープ（＝@vr）を鳴らすことができるという、面白い設計になっています。 k コマンドと r コマンドは、何も指定しなければどちらも休符扱いですが、r コマンドが KeyOff 時に即無音になるのに対して、k コマンドは「@vr」で指定したソフトエンベロープに切り替わるという点が異なります。 k コマンドも、r コマンドとうまく使い分けることで面白い効果を得ることができます。 なお、w コマンドと同様、直前発音長を 100%(q8/@q0)にしないと、k コマンドも r コマンドと同様になるので注意！

4. 発音途中の任意のタイミングで音色を変える

これは、**「@@」コマンドを使わずに、@v の同一発音内で音色だけを変化させたい場合に有用です。** ごく簡単な例ですが、内蔵音源でシンセサイザのフィルタを段階的に開閉するような変化をシミュレートする例を以下に示します。

```
@v0 = { | 15, 15, 14, 14, 13, 13, 12, 12, 11, 11, 12, 12, 13, 13, 14, 14 }
A       t150l16o4@v0 [@2a @1w @0w @1w]8
```

音色が**うわ～うわ～**とうねる様子が判るかと思います。 ファミコンの内蔵音源は、duty 比[1]の異なる音色を４つ（聴感上は実質３つ[2]）持っているのが大きな特徴ですが、これを段階的に切り替えることによってシンセサイザーのフィルタっぽい効果を出すことができます。

更に！ **VRC6 を使う場合は、ファミコン内蔵音源よりも細かい duty 比が指定できます[3]ので、より自然かつ細やかな表現が可能です**…というわけで、こんな感じです。 **どーん！！！**

```
#EX-VRC6
@v0 = { | 15, 15, 14, 14, 13, 13, 12, 12, 11, 11, 12, 12, 13, 13, 14, 14 }
M       t150l32q8o4@v0 @7a [@6w @5w @4w @3w @2w @1w @0w @1w @2w @3w @4w @5w @6w @7w]8
```

先程よりも、更にうねりの粒度が細かくなり、ここまで来るとシンセサイザーのフィルタにかなり近い感じがしたのではないでしょうか。 さぁ、みんなも VRC6 で**レッツ、うにょうにょ！！！**

---

[1] 矩形波（パルス波）は、豆腐を上下に交互に並べたような波形となっています。 上側の波形が１波長の何%を占めるかを「duty 比」と呼びます。 ファミコン内蔵と MMC5 の矩形波ですと、duty 比 12.5%(@0)、25%(@1)、50%(@2)、75%(@3)の４種類を指定できます。（VRC6 の場合は、6.75%(@0)～50%(@7)の８段階） duty 比 50%が最も澄んだ音で、これは PSG(SSG)/DCSG/SUNSOFT 5B とほぼ同一の音色となります。duty 比は、50%から離れるほど濁ったアクの強い音となります。 duty 比の異なる音色の使い分けこそが、ファミコンらしい音を作るのに最も重要な要素となります。

[2] duty 比 25%(@1)と 75%(@3)は、波形の上下位相が逆転している状態ですが、聴感上はほぼ区別がつきません。 単独で鳴らす場合は、同一音色とみなしても差支えないと思います。 但し、他の音と干渉した場合（近いピッチで同一音を鳴らすなど）に、出力される合成波形が変わりますので、聴感上の変化が発生する場合があります。 波形干渉まで計算に入れて楽曲を作る場合は、使い分けると良いでしょう。

[3] VRC6 の場合、6.25%(@0)～50%(@7)までの８段階が指定できます。（音色番号は、「duty 比 6.25 の倍数-1」となっています） 代わりに、2A03 にあった duty 比 75%は存在しません。

最後に

ppmck は w コマンドを使いこなせるようになってからが本番ですｗｗｗｗｗｗ

ｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗｗ

「上手な『草』の生やし方」を、是非実践してみてください。
しかし、ここに書かれていることはまだ基本的な使い方に過ぎません。　これらを組み合わせたり、異なる MML コマンドに対して適用させることによって、更に高度な表現も可能になるはずです。　ppmck の MML の中で、**自由自在に『草』を生やせるようになれば、もはや立派な「ppmck マスター」です！**

貴方の ppmck ライフが素晴らしいものになることを願いつつ、筆を置きたいと思います。

上手な『草』の生やし方 for ppmck

著者近影（画：purii）

著者：Ｗｉｚ．（wizpsg@fmpsg.info）
発行：打ち込み魂（http://fmpsg.info/wizcd/）

Special Thanks
purii（http://airshimeji.chips.jp/）
Izumi. & Manbow-J（http://www.geocities.co.jp/Playtown-Denei/9628/）
h7（http://ppmck.web.fc2.com/）
某吉（http://clogging.blog57.fc2.com/）
mck Wiki（http://wikiwiki.jp/mck/）
FMPSG（http://fmpsg.info/）
Wikipedia（http://ja.wikipedia.org/）
Google（http://www.google.co.jp/）
and...
この本を手にとってくださった読者の皆様

2015 年 5 月 24 日　初版発行
2015 年 10 月 31 日　PDF 版発行