

# MXDRV 使用指南

夏普 X68000 电脑音源驱动使用教程

Ripo\_2006

文档编写 | 2025/9/20

目录

- 文件头 ..... 2
- 乐器制作与定义..... 2
  - FM 乐器定义..... 2
  - ADPCM 乐器..... 4
- 指令 ..... 6
  - 编写规则 ..... 6
  - 通用指令 ..... 6
  - LFO 相关指令 ..... 11

# 文件头

#TITLE "曲名"

#PCMFIL "文件名.pdx"

设定一个 PDX 文件来播放 ADPCM，如需请查看[“PDX 文件制作”](#)章节来制作 PDX 文件并使用

## 乐器制作与定义

### FM 乐器定义

格式：

```
@1~255={  
AR, DR, SR, RR, SL, OL, KS, ML, DT1, DT2, AME,  
AR, DR, SR, RR, SL, OL, KS, ML, DT1, DT2, AME,  
AR, DR, SR, RR, SL, OL, KS, ML, DT1, DT2, AME,  
AR, DR, SR, RR, SL, OL, KS, ML, DT1, DT2, AME,  
CON, FL, OP  
}
```

解释：

AR = Attack Rate (0~31)

DR = Decay Rate (0~31)

SR = Sustain Rate (0~31)

RR = Release Rate (0~15)

SL = Sustain Level (0~15)

用法同等 ADSR；可以查阅[合成器基础：ADSR 包络线 - 知乎](#)

OL = Operator Level (0~127)

Operator 的音量大小，或调制等级

KS = Key Scaling (0~3)

ADSR 包络随音高缩放；音高越低包络缩放越长，音高越高包络缩放越短

ML = Multiplier (0~15)

Operator 的音高倍数；数字为对应倍数

DT1 = Detune 1 (0~7)

设置弱失谐；0~4 为偏高音失谐，5~7 为偏低音失谐

DT2 = Detune 2 (0~3)

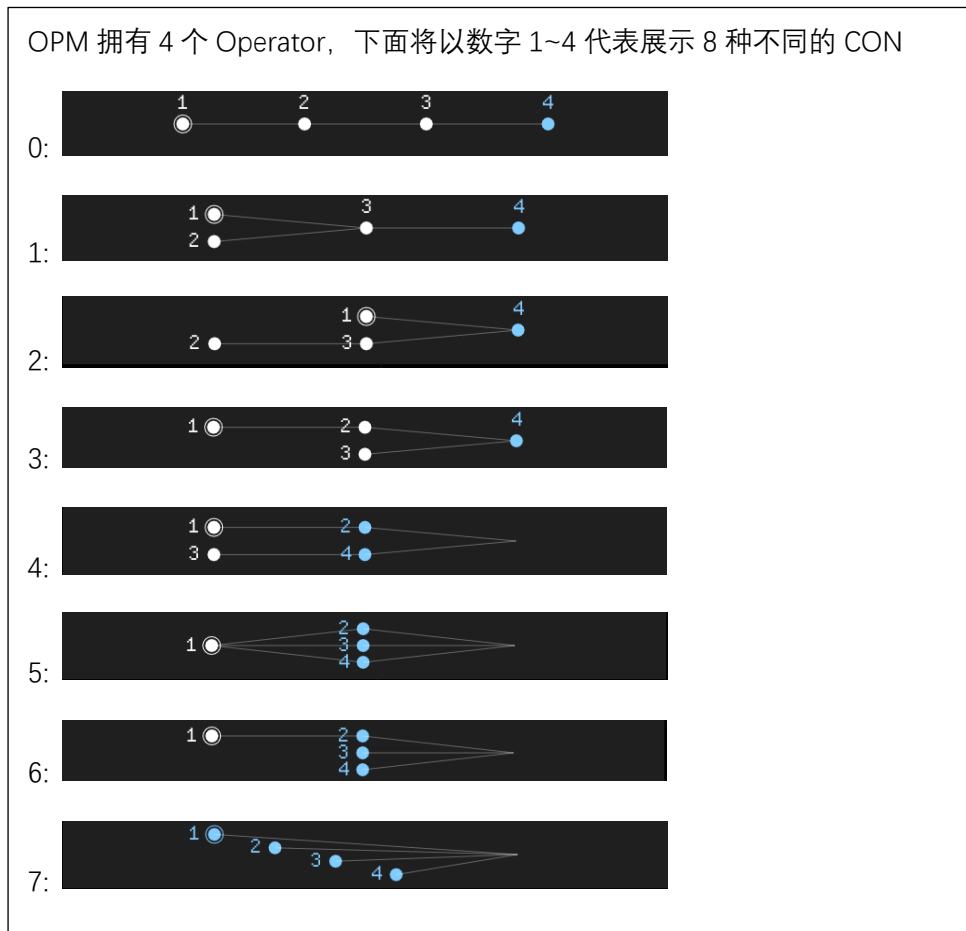
设置强失谐；数字越高越偏高音失谐

AME = Amplitude Modulation Enable (0~1)

调幅开关；0 为关，1 为开

CON = Connection. FM synthesis algorithm (0~7)

Operator 的连接方式；不同数字对应的连接方式参考下图



FL = Feedback Level (0~7)

设置 FB；只对 Operator 1 进行 FB，数字越高 FB 越强

OP = Operator Mask. 15 (binary %1111) turns on all 4 operators.

一般直接填 15 就好

3 = 1 op ; 7 = 2 op ; 11 = 3 op ; 15 = 4 op

# ADPCM 乐器

## PDX 文件制作

PDX 文件是 MXDRV 中用来制作 ADPCM 通道的重要文件，以下是如何从零制作 PDX 文件的教程

首先使用 WAV2ADP 来将.wav 文件转换为 raw ADPCM (.pcm) 文件

我们需要在终端窗口界面打入以下指令

```
run68 wav2adp [filename.wav]
```

如果你做对了的话，它会生成 .pcm 文件，在你搞定了所有采样之后，你需要把它们放进 PDX 文件中，不过在之前，我们需要一个.PDL 文件来指定所有采样来对应各自的音高符号。

一个 PDL 文件就只是一个重命名的 txt 文件。

新建一个.txt 文件，然后往里填入下信息：

```
#ex-pdx 1

@0
1=文件名 1.pcm
2=文件名 2.pcm
3=文件名 3.pcm
```

接下来我会解释这个文件的具体用处

我们将会使用 ex-pdx，在 PDX 格式被创造后，它更新了一个 pdx 文件能够载入更多采样的功能，一个 bank 能够载入大约 90 个采样文件

之后，我们使用 @0 指令来为 bank 中的采样进行定义。我们分别将文件名 1.pcm 映射到了音符音高 1，文件名 2.pcm 映射到了音符音高 2，文件名 3.pcm 映射到了音符音高 3，以此类推，我们可以按照这个顺序来映射剩下的采样。

在完成了以上步骤，我们可以准备开始制作 PDX 文件了

接下来，我们在终端输入以下指令，或是直接将.PDL 文件拖入 MKPDX

```
mkpdx filename.pdl
```

如果你都做对了的话，你应该就做出一个 PDX 文件了，你太牛逼了！现在你能够在曲子里嗯造 ADPCM 通道了

## PDX 文件使用

要在 ADPCM 通道使用 PDX 十分简单。

PDX 只能在 ADPCM 通道播放 (OKIM6258)，具体参考通用指令中的["设置 ADPCM 通道"](#)指令。

所有采样都在上文被我们以不同音高音符储存，如果你需要在目标轨道播放你想要的采样，你需要输入对应的音高，具体操作如下：

例 1: 一个四分音 kick 动次打次；假设 1=kick.pcm

```
o0 e4 e4 e4 e4  
o0l4[e]4
```

通过上面例子你会发现直接使用音符指令来播放有一点繁琐，因为你需要找到数值对应的音高，特别是遇到多个采样的情况，实在折磨人。不过别急，下面我们来学习更简单的方法！

我们可以直接通过 n# 指令来指定一个音符的数值音高，具体参考通用指令的["音符编写"](#)指令，具体操作如下：

列 1: 一个四分音 kick，八分音 hat；在做一个 clap；假设 1=kick.pcm 2=hat.pcm  
3=clap.pcm

```
l8n1n2n3n2  
n1,8n2,8n3,8n2,8
```

注意事项：

1. ADPCM 通道只能播放同一频率的采样，意思是你不能改变采样的音高
2. ADPCM 通道拥有一个统一的采样率，具体参考通用指令的["设置 PCM 频率"](#)

# 指令

## 编写规则

- 1. 所有指令遵循严格的大小写写法，请注意
- 2. `#` 符号代表数值
- 3. Ticks 是序列中使用的最小计算单位 (此文档将该单位设为 %)  
十六分音为%12；八分音为%24；四分音为%48；二分音为%96；一分音为%192

## 通用指令

<code>A~H</code>	设置 FM 通道，OPM 拥有 8 个通道，分别对应字母 <code>A~H</code>
<code>P~W</code>	设置 ADPCM 通道，OKIM6258 可以被分为 8 个通道，分别对应字母 <code>P~W</code>
在编写音符或是任何效果前，必须先以你想要编写的通道字符为开头	
<code>t#</code>	设置乐曲速度；以每秒的四分音符数量为单位
<code>@t#</code>	设置乐曲速度；直接设置 OPM register B 的数值，默认值为@t200
<code>@#</code>	<code># = 乐器编号 (0 ~ 255)</code> 设置乐器；乐器编号 0 为 ADPCM 通道专用，不可在 FM 通道使用
<code>F#</code>	<code># = 0 ~ 4</code> 设置 PCM 频率；默认值为 4，具体参考下列
<code>0 = 3.9kHz</code>	
<code>1 = 5.2kHz</code>	
<code>2 = 7.8kHz</code>	
<code>3 = 10.4kHz</code>	
<code>4 = 15.6kHz</code>	

L

设置全局循环开始点；当前通道所有指令播放完后将会重新播放从 L 符号之后的所有指令

o#

设置八度音程；范围为 o0 – o8，默认值为 o4

<

降低一个八度音程

>

升高一个八度音程

l#

设置音符默认长度，默认值为 l4

v#

# = 0 ~ 15

设置音量；范围为 v0 ~ v15，默认为 v8

@v#

# = 0 ~ 127

设置微调音量；范围为 @v0 ~ @v127

(

音量变化；降低一格当前通道音量

)

音量变化；提高一格当前通道音量

此命令不能通过后面添加数字来改变音量变化大小的幅度，但是你可以通过 [ ]# 指令或者是通过叠加音量变化指令来改变音量大小变化幅度

p#

# = 0 ~ 3

设置声像；p0=不输出，p1=左声像，p2=右声像，p3=左右声像，默认值为 p3

D#

# = -32767 ~ 32767

设置失谐；以 1/64 的半音为单位

[]#

# = 2 ~ 255

设置循环；将会循环 [] 符号内的指令 # 次

/

跳过循环；在 [] 指令内使用，在最后一次循环时，跳过 / 符号后的指令

!

忽略所有 ! 符号后的指令

;

行注释； 在同一行的 ; 指令后的所有字符都会被注释

/\* \*/

块注释； 在 /\* \*/ 指令内的所有字符都会被注释

音符编写:

格式: nal (占位字符详见下方解释)

例子: c+4.

-----

n =

c d e f g a b

音符组; 使用该数组来编写音符

r

休止符

-----

a =

+

升高半音指令

-

降低半音指令

-----

l =

1 2 4 8 16 32

x 分音组 (时值); 使用该数组来编写每个音符的分音 (时值)

.

付点符号; 用于延长当前音符或休止符时值的一半

%#

# = %

时值 (Ticks); 直接使用 Ticks 来设置音符时值; 使用 # 符号也能使用该指令

例 1: c%48 = c#48 = c4

-----

n#

# = 0~95

直接使用数字指定音高; n0 = o0d+ , n95 = o8d。使用该方式编写音符时如需指定时值, 请用 [ ] 符号分隔后填入对应时值 (该指令适合用于 ADPCM 通道)

&

绑定 macro; 用于连接前后两个音符的 macro

例 1:  $c4 \& c4 = c2$

例 2:  $c4 \& > c4$

\_

滑音; 用于音符向另一个音符的滑动效果

例 1:  $c4\_f$  = 音符 c 以四分音的速度滑向音符 f

q#

# = 0 ~ 8

设置断音; 设定音符断音长度, 范围为 q0 ~ q8, 默认值为 q8

@q#

# = %

设置断音; 以 Ticks 为单位设定音符断音长度

k#

# = %

延迟音符; 音符将会延迟 # 个 Ticks 播放

w#

# = 0 ~ 31

设置 OPM 噪声频率; 范围为 w0 ~ w31

y#1,#2

#1 = register

#2 = value

对 OPM register #1 写入数值 #2

W

等待同步；停止播放该通道指令，只有等到其他通道发送同步指令(S)后才会在该位置继续播放

S#

# = A~H (FM 通道) 或 P (ADPCM 通道) / 0~8

发送同步；当播放到 S 指令时，继续播放指定的通道指令

## LFO 相关指令

MP#1,#2,#3

#1= LFO 波形 (0=锯齿波, 1=方波, 2=三角波)

#2= 设置 #2 Ticks 的 1/4 LFO 循环

#3= 设置 LFO 最大振幅

设置音高 LFO

MPON

开启音高 LFO；当设置 MP 指令后会自动开启

MPOF

关闭音高 LFO

MA#1,#2,#3

#1= LFO 波形 (0=锯齿波, 1=方波, 2=三角波)

#2= 设置 #2 Ticks 的 1/4 LFO 循环

#3= 设置 LFO 最大振幅

设置振幅 LFO

MAON

开启振幅 LFO；当设置 MA 指令后会自动开启

MAOF

关闭振幅 LFO

MD#

# = 0 ~ 255

延迟 LFO; 延迟 LFO 在一个音符开始的时间, 单位为 Ticks, 范围为 MD0 ~ MD255, 该指令对 OPM LFO 指令无效

MH#1,#2,#3,#4,#5,#6,#7

#1= LFO 波形 (0=锯齿波, 1=方波, 2=三角波, 3=噪波)

#2= LFRQ

#3= PMD

#4= AMD

#5= PMS

#6= AMS

#7= key sync (0 = no sync, 1 = sync)

设置 OPM LFO

MHON

开启 OPM LFO; 当设置 MH 指令后会自动开启

MHOF

关闭 OPM LFO