VT03 Program Sound Generator Users Manual

FMDemo.rar 是 6502 原文件(Source Code)，midi2vt.rar 是工具软件，请自行使用 Cakewalk 来制作 midi 文件。

操作简介：

1、用 Cakewalk 制作 midi 文件，注意此 midi 文件最多为 5 轨，分别命名对应 "Square1","Square2","Triangle","Noise","DPCM"。具体可参考附件 midi2vt.rar 中的 midi02.mid。
(只能用 Midi Type2)欲转换之 midi 文件必须要有 5 个音轨,转换程序会按照由高到低依序对应到 VT03 的 Square1,Square2,Triangle,Noise and DPCM,如果该声部不用则空一个音轨,转换 midi 文件是按照 NTSC 的速率匹配,如果是 PAL 制式下需手动调整音乐速度.
2、用 midi2vt.exe 转换制作好的 midi 文件，可生成 bin 文件供即时播放，同时生成 asm 和 h 文件可供编译使用。
3、关于 asm 文档的添加请参阅 FMDemo.rar 中的 2MusData.asm,压缩包中 3EffData.asm 同时包括 50 种常用音效可用手柄选择播放,其代码数据可供用户参考使用，播放驱动程序请参阅 5Music.asm，相关说明如下：

==========

程序变量说明

==========

>>> Music_Flag1：音乐播放程序同用户程序交互标志 1，主要用于声画同步，在每个音乐播放时设置为零.
>>> Music_Flag2：音乐播放程序同用户程序交互标志 2，必须嵌入用户程序控制.

>>> Music_Tone　：音调基准值，在整个音乐程序初始化时设置为零

>>> Music_MuteFlag：静音标志，由 BIOS 设置
　　bit 0　音效静音标志
　　　0 = Normal
　　　1 = Mute
　　bit 1-6　Null
　　bit 7　　音乐静音标志
　　　0 = Normal
　　　1 = Mute

>>> System_Flag：系统综合标志
　　bit 0　运行状态
　　　0 = Normal
　　　1 = Demo
　　bit 7　电视制式
　　　0 = NTSC
　　　1 = PAL

>>> Music_SpeedCount：音乐播放速度递减计数器
>>> Music_SpeedIndex：音乐速度索引
>>> Music_SpeedValue：音乐速度值，由 4 字节组成，由播放程序依次索引使用

>>> Music_SquareIndexTable：预置方波音色数据索引表

>>> Music_TriangleIndexTable：预置三角波音色数据索引表

>>> Music_NoiseDataTable：预置噪声音色数据表

>>> Music_DPCMIndexTable：预置 DPCM 音色数据索引表

>>> Music_PlayBuffer ds 5*8：音乐播放数据表

>>> Music_PlayControl1 : 音乐播放控制寄存器 1
bit 0-6 节拍等待计数器
0 = 128
bit 7 本组控制数据处理状态
0 = Disable
1 = Enable

>>> Music_PlayControl2 : 音乐播放控制寄存器 2
bit 0-3 当前音量设置
bit 4-5 音乐循环计数器

>>> Music_PlayAddress : 当前音乐控制数据地址

>>> Music_TimbreAddress : 当前音色数据地址

>>> **Music_ToneBuffer ds 8*10 ：音色输出控制**

>>> Music_ToneControl1 : 音色输出控制寄存器 1
bit 0-6 声道等待计数器
0 = 64
bit 5 音色输出状态
0 = 正常
1 = 索引处理
bit 7 本声道处理状态
0 = Disable
1 = Enable

>>> Music_ToneControl2 : 音色输出控制寄存器 4
bit0-3 循环计数器
bit4-7 索引输出 Mask 值

>>> Music_ToneControl3 : 音色输出控制寄存器 2
bit 0-5 输出间隔值
bit 6,7 索引处理状态
00 = 索引输出
01 = 音量索引加输出
10 = 音符索引加输出
11 = 音量和音符索引加输出

>>> Music_ToneControl4 : 音色输出控制寄存器 3
bit 0-3 索引输入输出计数器
bit4-5 索引输入 Mask 值

>>> Music_ToneAddress : 音色控制数据表地址

>>> Music_ToneOutData0 : 输出数据，对应$40x0

>>> Music_ToneOutData1 : 输出数据，对应$40x1

>>> Music_ToneOutData2 : 输出数据，对应$40x2

>>> Music_ToneOutData3 : 输出数据，对应$40x3

==========

程序函数说明

==========

>>> MusicInitial : 音乐程序初始化

入口数据： Music_AddrL,H 指向音乐数据索引表

音乐数据索引表格式：

adr 乐曲索引表地址
adr 音效索引表地址
adr Square1 声部音色索引表地址
adr Square2 声部音色索引表地址
adr Triangle 声部音色索引表地址
adr Noise 声部音色索引表地址
adr DPCM 声部音色索引表地址

乐曲索引表格式：

adr 第一首乐曲数据地址
adr 第二首乐曲数据地址
... .
adr 第 n 首乐曲数据地址

乐曲数据格式：

byt s1,s2,s3,s4 音乐速度值
adr Square1 声部音乐数据
0 = 没有对应数据
其它正常
adr Square2 声部音乐数据
0 = 没有对应数据
其它正常
adr Triangle 声部音乐数据
0 = 没有对应数据
其它正常
adr Noise 声部音乐数据
0 = 没有对应数据
其它正常
adr DPCM 声部音乐数据
0 = 没有对应数据
其它正常

音效索引表格式：

adr 第一首音效数据地址
adr 第二首音效数据地址
... .
adr 第 n 首音效数据地址

音效数据格式：

adr Square1 声部音效数据
0 = 没有对应数据
其它正常
adr Square2 声部音效数据
0 = 没有对应数据
其它正常
adr Triangle 声部音效数据
0 = 没有对应数据
其它正常

adr Noise 声部音效数据
0 = 没有对应数据
其它正常

>>> MusicPlayEnable : 音乐播放

>>> MusicPlayDisable : 音乐静音

>>> MusicEffectEnable : 音效播放

>>> MusicEffectDisable :音效静音

>>> MusicPlay :音乐播放处理
入口参数:
REG A = 音乐编号

>>> MusicEffect : 音效播放处理
入口参数:
REG A = 音效编号

==========

播放控制数据格式：MPCC....Music Play Control Code

==========

>>> MPCC_Wait : 等待节拍
Square/Triangle/Noise/DPCM 声部
db %0ccccccc
c = 等待的节拍数(0-127)

>>> MPCC_Timbre : 设置音色
Square/Triangle 声部
db %1000iiii
i = 预置音色索引
Noise/DPCM 声部
无效

>>> MPCC_Volume : 设置音量
Square/Noise 声部
db %1001vvvv
v = 音量
Triangle/DPCM 声部
无效

>>> MPCC_Inside : 按预置数据播放
Square/Triangle 声部
db %1010tttt , %cccccIII
t = 音符
$0 1
$1 1#
$2 2
$3 2#
$4 3
$5 4
$6 4#
$7 5

$8 5#
$9 6
$A 6#
$B 7
$C 空音符
$D 空音符
$E 空音符
$F 空音符
l = 音高
0 -...
1 -..
2 -.
3 =
4 +.
5 +..
6 null
7 null
c = 节拍长度
0-31
Noise & DPCM 声部
%1010iiii
i = 预置音色索引(0-15)

>>> MPCC_Loop : 在指定地址到当前地址循环播放
Square/Triangle/Noise/DPCM 声部
db %1011cccc
dw Address
c = 循环次数
Address 为指定的循环地址（不可以嵌套）

>>> MPCC_ToneSet : 设置音调基准值
Square/Triangle/Noise/DPCM 声部，一般在 Square 声部中使用
db %1100tone
tone = 新的音调基准值

>>> MPCC_FlagSet :设置 Music_Flag1 为 data
Square/Triangle/Noise/DPCM 声部，一般在 Square 声部中使用
db %11110000,data
音乐播放开始时 Music_Flag 被清零

>>> MPCC_FlagInc Music_Flag1 加一
Square/Triangle/Noise/DPCM 声部，一般在 Square 声部中使用
db %11110001

>>> MPCC_FlagDec Music_Flag1 加一
Square/Triangle/Noise/DPCM 声部，一般在 Square 声部中使用
db %11110010

>>> MPCC_FlagBne 如果 Music_Flag1 不等于 data 跳转
Square/Triangle/Noise/DPCM 声部，一般在 Square 声部中使用
db %11110011,data
dw Address
Address = 转移地址

>>> MPCC_FlagBeq 如果 Music_Flag1 等于 data 跳转
Square/Triangle/Noise/DPCM 声部，一般在 Square 声部中使用
db %11110100,data
dw Address
Address = 转移地址

>>> MPCC_FlagBneCall 如果 Music_Flag1 不等于 data 执行外部程序
Square/Triangle/Noise/DPCM 声部，一般在 Square 声部中使用
db %11110101
dw Address
Address = 外部程序地址

>>> MPCC_FlagBeqCall 如果 Music_Flag1 等于 data 执行外部程序
Square/Triangle/Noise/DPCM 声部，一般在 Square 声部中使用
db %11110110
dw Address
Address = 外部程序地址

>>> MPCC_SpeedSet 重新设置播放速度
Square/Triangle/Noise/DPCM 声部，一般在 Square 声部中使用
db %11110111

>>> MPCC_SpeedUp 加快播放速度
Square/Triangle/Noise/DPCM 声部，一般在 Square 声部中使用
db %11111000

>>> MPCC_SpeedDown 减慢播放速度
Square/Triangle/Noise/DPCM 声部，一般在 Square 声部中使用
db %11111001

>>> MPCC_ToneUp 播放升调
Square/Triangle/Noise/DPCM 声部，一般在 Square 声部中使用
db %11111010

>>> MPCC_ToneDown 播放降调
Square/Triangle/Noise/DPCM 声部，一般在 Square 声部中使用
db %11111011

>>> MPCC_User 用户自定义音色
Square/Triangle/Noise 声部
db %11111100
dw Address
Address 为用户定义的音色数据表
DPCM 声部
db %11111100,a,b,c,d
a 对应于$4010
b 对应于$4011
c 对应于$4012
d 对应于$4013

>>> MPCC_Call 执行外部程序

Square/Triangle/Noise/DPCM 声部，一般在 Square 声部中使用
db %11111101
dw Address
Address = 外部程序地址

>>> MPCC_Jump 跳到指定的地址继续播放
Square/Triangle/Noise/DPCM 声部
db %11111110
dw Address
Address = 转移地址

>>> MPCC_End 播放数据结束
Square/Triangle/Noise/DPCM 声部
db %11111111

==========

音色控制数据格式:MTCC....Music Tone Control Code

==========

>>> MTCC_wait 等待 NMI
db %0-cccccc
c = 等待的 NMI 数(0-127)

>>> MTCC_Out 输出 PSG 缓冲器
db %1000abcd[,A][,B][,C][,D]
abcd = 输出 mask
0 = 无对应数据输出
1 = 有对应数据输出

>>> MTCC_Put 设置 PSG 缓冲器
db %1001abcd[,A][,B][,C][,D]
abcd = 设置 mask
0 = 无对应数据
1 = 有对应数据
A = ToneOutData0
B = ToneOutData1
C = ToneOutData2
D = ToneOutData3

>>> MTCC_Or 或 PSG 缓冲器
db %1010abcd[,A][,B][,C][,D]
abcd = 或 mask
0 = 无对应数据
1 = 有对应数据
A = ToneOutData0
B = ToneOutData1
C = ToneOutData2
D = ToneOutData3

>>> MTCC_Output 设置并输出 PSG 缓冲器
db %1011abcd[,A][,B][,C][,D]
abcd = 设置和输出 mask
0 = 无对应数据
1 = 有对应数据
A = ToneOutData0
B = ToneOutData1

C = ToneOutData2
D = ToneOutData3

>>> MTCC_Setup 索引输出设置
db %1100abcd,--cccccc
abcd = 输出 Mask，对 IndexOut 系列命令有效
cccccc = 输出间隔值

>>> MTCC_IndexOut 索引设置并输出 PSG 缓冲器
db %1101--00, %abcdnnnn
db data1,data2....data?
abcd = 设置 mask
nnnn = 输出数量
先输出再等待间隔值

>>> MTCC_VolumeIndexOut 音量索引加并输出 PSG 缓冲器
db %1101--01, %----nnnn
db data1,data2....data?
nnnn = 输出数量
先输出再等待间隔值

>>>MTCC_ToneIndexOut 音符索引加并输出 PSG 缓冲器
db %1101--10, %----nnnn
db data1,data2....data?
nnnn = 输出数量
先输出再等待间隔值

>>> MTCC_VolumeToneIndexOut 音量和音符索引加并输出 PSG 缓冲器
db %1101--11, %----nnnn
db data1,data2....data?
nnnn = 输出数量
先输出再等待间隔值

>>> MTCC_Loop 循环
db %1110cccc
dw Address
cccc = 循环次数
Address = 循环地址

>>> MTCC_VolumeInc 音量加一
db %11110000

>>> MTCC_VolumeDec 音量减一
db %11110001

>>> MTCC_VolumeBne 音量不等于 data 跳转
db %11110010, data
dw Address
Address = 转移地址

>>> MTCC_VolumeBeq 音量等于 data 跳转
db %11110011, data

dw Address
Address = 转移地址

>>> MTCC_VolumeReset 音量恢复为默认值
db %11110100

>>> MTCC_VolumeClear 音量清除
db %11110101

>>> MTCC_Flag2Beq Music_Flag2 等于 data 跳转
db %11110110,data
dw Address
Address = 转移地址

>>> MTCC_Flag2Bne Music_Flag2 不等于 data 跳转
db %11110111,data
dw Address
Address = 转移地址

>>> MTCC_ToneAdd 音符加 data
db %11111000,data

>>> MTCC_SetpIndexOut 单步索引设置并输出 PSG 缓冲器，每调用一次输出一次
db %11111001, %abcd----
dw Address
abcd = 设置 mask
Address = 索引地址
先输出再等待间隔值，索引偏移为 Music_ToneControl4，每索引一次自动增加，
Music_ToneControl4 在从未使用过 IndexOut 系列命令情况下为零

>>> MTCC_Flag1Beq Music_Flag1 等于 data 跳转
db %11111010, data
dw Address
Address = 转移地址

>>> MTCC_Flag1Bne Music_Flag1 不等于 data 跳转
db %11111011, data
dw Address
Address = 转移地址

>>> MTCC_LoopBeq 如果循环递减计数器等于 data 跳转
db %11111100,data
dw Address
Address = 转移地址

>>> MTCC_Call 执行外部程序
db %11111101
dw Address
Address = 外部程序地址
在不使用 IndexOut 系列命令的前提下，外部程序可以使用的寄存器有：
Music_ToneControl2 高四位

Music_ToneControl3 全部
Music_ToneControl4 全部
Music_Flag1 全部
Music_Flag2 全部

>>> MTCC_Jump 无条件转移
db %11111110
dw Address
Address = 转移地址

>>> MTCC_End 音色控制数据表结束
db %11111111
