003 – Play music with YM2413 and SN76489 sound chips (svgm driver)

Prerequisite : 000 – Create a new game project

This tutorial describes :

- How to setup a DefleMask project
- How to convert music from VGM file format
- How to play the music with the game engine

Introduction

The game engine provides multiple ways to play music on TO8.

This tutorial will explain how to use the svgm driver of the game engine.

The svgm is a simplified version of vgm format.

Data is :

```
 x00-x38 xnn           : (2 bytes) YM2413
 x40-xff               : (1 byte)  SN76489
 x39
   |____ x00           : (2 bytes) stop track
   |____ xnn           : (2 bytes) wait xnn frames (x01-x7f)
   |____ x80 xnnnn     : (4 bytes) jump to xnnnn
   |____ x81 xnnnn xnn : (5 bytes) replay at xnnnn (signed offset) xnn bytes
and return
```

Note : for the Data byte of the SN76489, the unused bit6 is set at 1 to allow an easy distinction between the two chips. This operation is handheld by the java converter.

```
%0-DDDDDD
    |``````-- Data
    `-------- Unused
```

Setup a DefleMask project

There is no convenient solution to make music for YM2413, so it was decided to use DefleMask with a genesis profile. The tool chain will convert data from YM2612 to YM2413 chip.

First, you will need to buy and install DefleMask.

Next setup a new project with “Genesis” system of type “Standard”

Use those files :

- a set of YM2612 instruments that matches YM2413 preset instruments

  YM2612 settings TO
  YM2413 instruments

- a set of drum samples from YM2413

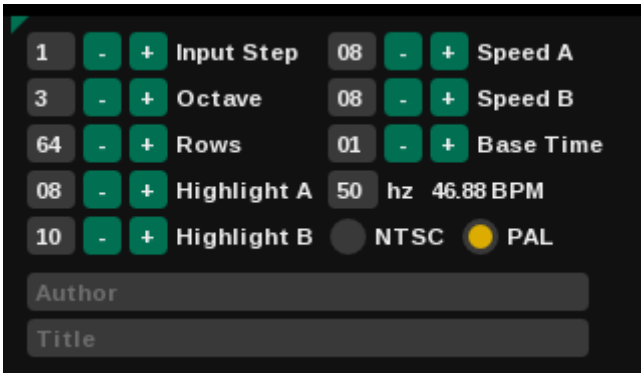
YM2413 drum samples.rar

Samples should be used on FM6 track only, and should be mapped to those notes :

- C-3 : BD
- C#3 : SD
- D-3 : HH
- D#3 : CYM
- E-3 : TOM

Warning !

You should setup DefleMask to PAL 50Hz, it's mandatory.



Adjust Base Time, Speed A and Speed B to set playback speed.

If you want to know how instruments where matched from YM2413 to YM2612, use this file :

YM2612 - YM2413 intstruments.xlsx

You may also use this information to use any software that will produce VGM with YM2612 and SN76489 data.

Now make your own music … once it's done, export your project to VGM file format.

Here is a sample file (by adnz) if you are lazy :

zmusic.vgm

Convert VGM file from YM2612 to YM2413

To convert VGM file, we will be using an external tool : vgm-conv

First, you will need to install node.js on your computer :

https://nodejs.org/en/download/

Next, run this command :

```
npm install -g vgm-conv
```

Once the tool is installed, go where the vgm file resides and run :

```
vgm-conv -f ym2612 -t ym2413 -D dacEmulation=none -o zmusic_o.vgm zmusic.vgm
```

This will produce a zmusic_o.vgm with ym2413 data instead of ym2612.

This will also strip pcm data block.

To be able to play the file on TO8 with the svgm driver, we need to change some data :

- YM2612 pcm play should be replaced with YM2413 drumkit notes
- Chip tags should be removed
- Wait frames tags should be remapped and encoded
- …

All those changes are handled by a java converter included in the game engine.

run this command from the root of the game engine (if you are elsewhere, juste change the path to the jar in the command) :

```
java -cp ./java-generator/target/game-engine-0.0.1-SNAPSHOT-jar-with-dependencies.jar fr.bento8.to8.audio.SVGMTool.SVGMTool zmusic_o.vgm zmusic.svgm -drumvol=0xF2,0x62,0x44
```

Now you should have a file named zmusic.svgm

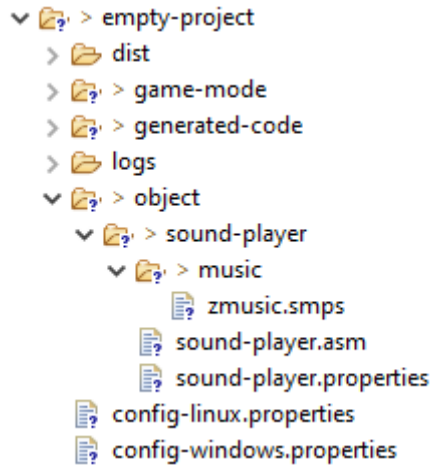
If you want to adjust volume for each drumkit instrument change the parameters in the command line, remember it's attenuation value not volume (0:max, F:min) :

```
Default values are:

0xF2 : F (unused) 2 (attenuation value for Bass drum)

0x62 : 6 (attenuation value for Hi-hat) 2 (attenuation value for Snare drum)

0x44 : 4 (attenuation value for Tom) 4 (attenuation value for Cymbal)
```

Play the music

File structure

Enter the infinite-loop directory and create the object directory structure (object -> sound-player -> music) :



- Put the .svgm file in the music directory.
- Create a sound-player.asm
- Create a sound-player.properties

An object is a routine with a set of variables and data (images, sound, music, ..)

As opposed to the main routine of a game-mode that runs from the RAM Page 1 ($6100-$9fff), an object is executed from RAM pages (4-31) mounted in the $0000-$3fff space.

Game Mode definition

infinite-loop.properties

To define a game mode object, add an object line to the **infinite-loop.properties** :

```
engine.asm.mainEngine=./game-mode/infinite-loop/main.asm

# Objects
object.sound_player=./object/sound-player/sound-player.properties
```

Remove the palette and act definition if you reused previous tutorials or keep it but you will have to set the needed code in the main source source (it could be a good exercice).

Object properties

Set those lines in the sound-player.properties :

```
code=./object/sound-player/sound-player.asm

# Sound
sound.Svgm_zmusic=./object/sound-player/music/zmusic.svgm
```

Object source code

Set this code into the object source code **sound-player.asm** :

```
; ------------------------------------------------------------------------
; Object - sound player
;
; input REG : [u] pointer to Object Status Table (OST)
```

```
; ---------
;
; ------------------------------------------------------------------------------

        jsr   IrqSet50Hz
        ldx   #Svgm_zmusic
        jsr   PlayMusic
        jmp   ClearObj
```

This code is very simple for an object code, in following tutorials, you will see a much more complex approach. However, here we are just writing a code that will :

- set an IRQ at 50Hz for the smps driver to send data at a regular speed to the sound cards
- load a register with the address of the music data (note that the label is the one used in the .properties file)
- start the playing of music
- self-destroy the object

Game mode source code

The game mode will be responsible of the object instantiation. To do so, we need to add the object id in the Object RAM location (the location of each object variables).

Note : this object will run once and clean this id, thus the object will be only run once.

Place this code into the game mode source code **main.asm**:

```
        INCLUDE "./Engine/Constants.asm"
        INCLUDE "./Engine/Macros.asm"
        org   $6100

*
*******************************************************************************
* Main Loop
*
*******************************************************************************
LevelMainLoop
        jsr   WaitVBL
        jsr   RunObjects
        bra   LevelMainLoop

Object_RAM
                            fcb   ObjID_sound_player
                            fill  0,object_size-1
Object_RAM_End

*
*******************************************************************************
* Routines
*
*******************************************************************************
```

```
        INCLUDE "./Engine/Ram/BankSwitch.asm"
        INCLUDE "./Engine/Graphics/WaitVBL.asm"
        INCLUDE "./Engine/ObjectManagement/RunObjects.asm"
        INCLUDE "./Engine/ObjectManagement/ClearObj.asm"
```

You are ready to build and run your program.

Details in 000 tutorial