FM-Sid driver v1.3 (April 2016)

Hi,

As I'm going to be incredibly busy for the next few months I've decided to release the in development version of FM-Sid "as is", until I get time to work on the proper additions for v1.4.

note : If you don't like editing music with external tools and assemblers you won't enjoy using this, it is NOT a standalone tracker and never will be. :)

thanks, 4mat / Ate Bit

1) What is this?

FM-Sid is probably the first c64 music driver that lets you use the SFX FM Expander cartridge when composing, giving you 12 channels in total. It allows you to either write songs with the basic standalone driver, or use another editor to provide the SID channels while the driver plays back the FM ones. This means using tools with multi-SID support (such as Sid Wizard of Goat Tracker) can give you up to 18 channel songs on one C64. (9 SID channels + 9 FM channels)

2) Setup

To use the driver you'll need:

- DASM assembler. (unless you want to convert the source to another assembler)
- Some sort of XM module editor. (FastTracker 2, MilkyTracker, OpenMPT etc)
- A SID editor if you don't want to use the driver that comes with it. (GoatTracker, Sid Wizard etc)
- A version of Windows (XP and above) to use the conversion tools.

The source is in DASM format. If you don't use this already you'll need to download it and put it in the src folder, I can recommend the one Lasse built here:

http://cadaver.homeftp.net/tools.htm

But any version should work. As long as the file structure stays the same you can also just compile **fmsid_driver.asm** in your own assembler setup.

3) Running the driver

1) Run the compile.bat file in the src folder.
2) Drop test.prg into emulator or your real machine.

4) Writing a song with the internal driver

While you can write a song directly in assembly it's not fun. So FM-Sid comes with a couple of conversion tools that take an .XM module format as the source.

1) Make a 12-channel XM.
2) Run fmsidxm.exe on the command line with your XM as the filename, or drop your XM on the exe.
3) A window will briefly flash up and if all goes well **musdata.asm** gets generated. There are no error messages or anything yet.
4) Load up **fmsid_driver.asm** , near the top are two TEMPO settings. Put the tempo of your song in TEMPO1 & TEMPO2 for constant tempo. If you want 'funk tempo' or an intermediate tempo change TEMPO2 to another value. (same as using other trackers)
5) If using an NTSC setup change the NTSC flag to 1. This changes the pitch of the built-in sid driver ONLY. As far as I can tell the FM cart automatically changes pitch based on the system.

Things to note with the driver/converter:

- The first 3 channels in your XM are always the SID ones. If using an external driver you can either ignore them, or use the included xm2c64.exe to convert them to a c64 sid format.
- You can have 256 FM instruments and 256 sid instruments. (with the built-in driver) Both FM & SID share the same instrument numbers so the channel used designates which hardware gets used. (channels 1-3 = sid , channels 4-12 = FM)
- The pitch offset of the instruments doesn't matter as the converter only reads the note number. (from C-0 upwards) See in the included **testfm.XM** , the XM is basically a guideline for the song structure.
- Patterns can be up to 256 beats in length, any longer and it's likely the converter will get out of sync.
- There are currently no effects, only note placement. If you want an instrument to change volume, make another instrument at that volume and use that.

5) Using a different SID driver with FM-Sid

1) Export your song from whatever tracker you use (as a prg) and put in the same folder as FM-Sid.
2) Load up **fmsid_external.asm** and change the filename of the prg to your song.
3) Load up **fmsid_driver.asm** and change:
   OWNSIDPLAYER to 1
   OWNSIDADDRESS to your song's start address (usually $1000) My driver will assume your song's init is at that address and play frame is at address+$03 (it usually is)
4) Now you have to get them to sync by changing the number in OWNSIDSKIPFRAMES . Don't worry it's straightforward. :) Most modern SID drivers either delay the output by a few frames, or use 'ghostbytes' to write all SID values at once at a later date. This offset command will call the external driver during the init phase X amount of frames before the song starts.

- Testing with GoatTracker 1.x I noticed the number is usually songtempo-1 (eg: if TEMPO1 = 6 , OWNSIDSKIPFRAMES = 5)
- Testing with GoatTracker 2.x the number was (songtempo-1)+(songtempo-2) (eg: if TEMPO1 = 6, OWNSIDSKIPFRAMES = 9)

Things to note with using external drivers:

- Only vblank drivers can be used and no multi-speed. (might support 2x at some point but there's no time for anything more!)
- The default address of the FM player is $2000 , default SID is $1000 so you might need to change these if your SID song is >4kb.

6) Using the xm2c64.exe tool

XM2C64 is a tool to convert XM data to popular C64 music formats such as Goat Tracker and Sid Wizard. The internal SID player isn't very good, this is a way to get your XM song data over to a better music editor where it can be refined.

1) Have your SID channels as the first 3 channels of your XM.
2) Run xm2c64.exe on the command line with your XM as the filename, or drop your XM on the exe.
3) Currently it only supports GoatTracker 2.x format, it will export a .sng file with the same name as your XM which you can then edit in Goat Tracker.
4) Only song/pattern and note placement is currently supported. This is just for getting the music data over so you can refine it in whatever editor you use.

7) Extra options for standalone songs

You can change the look of the standalone player. Load up **fmsid_driver.asm** and change these options at the top:

DEBUGSKIP - Set to 1 lets you fast-forward the song by holding space bar.

DEBUGRASTERTIME - Set to 1 to show rastertime amount.

DEBUGCRUNCH - If this is 0 the BASIC line will be included so the program runs automatically, if you're planning to crunch the exported file (recommended) then set this to 1 to build without the basic header. The init address for your cruncher will be $0801

DEBUGBORDERCOLOR - Border colour.

DEBUGPAPERCOLOR - Screen colour.

DEBUGRASTERTIMECOLOR - Raster time colour.

DEBUGPATTERNSTART - Skips X amount of patterns at the start of the song but playing through at full-speed. (not very useful)

You can also change the PETSCII pic that's displayed by altering the filenames in **fmsid_pictures.asm** (standard 1kb screen & colour maps)

8) Exporting song for use elsewhere (but where?)

1) Load up **fmsid_driver.asm** and set DEBUGMODE to 0
2) Set the build address of the driver with the ADDRESS variable. Useage is the standard:
   JSR ADDRESS = Init driver
   JSR ADDRESS+$03 = Play Frame

If you're using an external driver for SID music that won't get built when exporting, so you need to include that in your target source by hand. However my driver will call the external driver automatically so you don't need to do anything else with it apart from including it.

9) Making FM instruments

Each FM instrument currently sets the YM registers once when a new note is triggered. It doesn't update pitch or anything after that point. I want to add this stuff but for the moment it makes creating instruments very simple. :)

1) Load up **fmsid_fminst.asm**
2) Each instrument is 9 bytes long, these are the 9 registers available on the YM3526. In order they are: (from the adlib docs)

Mod characteristic (Mult, KSR, EG, VIB and AM flags)
Carrier characteristic (Mult, KSR, EG, VIB and AM flags)
Mod key scaling/output level (KSL in Adlib Tracker2)
Carrier key scaling/output level (KSL)
Mod attack/decay level
Carrier attack/decay level
Mod sustain/release level
Carrier sustain/release level
Feedback/connection (in Adlib Tracker2 this is Connection/FB)

3) If you add more instruments than the 16 I've put in you have to add the label address to fminstoffsetlo/hi . Just copy what's in there.
4) Personally I haven't used an adlib chip since the '90s, so I've been converting sounds made in Adlib Tracker 2. http://www.adlibtracker.net/

10) How to convert Adlib Tracker instruments (until I write an auto converter)

1) Load up AT2 and press CTRL+I to enter the instrument editor.
2) In the left pane are the hex numbers you need to type in, though you'll notice Carrier & Modulator are swapped around for each item.
3) The only one that needs changing from it's AT2 value is Connection/FB:

The FB number needs to be translated into bits starting from bit 1 rather than bit 0.
(eg: if it's 7 (1+2+4) the result would instead be 2+4+8 = 14)
If Connection is FM add 1 to the FB number to get your total.

More info on the adlib registers can be found here: http://www.shipbrook.net/jeff/sb.html

11) Making SID instruments with the internal player

If you really want to use my SID driver:

1) Load up **fm_sidinst.asm**
2) There are two frame tables, like most drivers:

insttab - waveform table

Set waveforms per frame as usual, use $fX to loop back at end of instrument. ($f1 = stop , $f2-$fe loop back X places)

** MAKE SURE there's a .byte $FF at the end of this table. It's used by the auto-find code. **

instnote - pitch/arp table

The length of instruments in this table must match the length of the same instrument in insttab.

$00 - play note as is
$01 - $ef - Play this absolute pitch. (eg: for drums)
$f0 - $fc - Arpeggio value.

3) There are a bunch of settings for instrument make-up:

instoff ** MAKE SURE this is one byte longer than the amount of sid instruments. It's used by the auto-find code. **

instadc - Attack/Decay

instsus - Sustain/Release

instdec - $XY - X = 1 means pulsewidth won't restart on subsequent notes using this instrument. (think Rob Hubbard bass style)
Y = 1 means pitch slide enabled (only on noise/tri/saw waves I think)

instadd - Amount to add to pulsewidth/pitch slide each frame.

instpul - $XY - X = 1 was vibrato enabled but vibrato has been taken out.
Y = Coarse pulsewidth setting. 0-F

volume - $XY - X = Filter type (1 = low pass etc)
Y = Global volume 0-F

filtres - $XY - X = Filter resonance , Y = channel(s) to filter (bit pattern, see c64 manual)

instfilt - Filter cut-off start position , if 0 filter settings aren't used on this instrument

instfila - Filter sweep add value. (as with other players >$80 = backwards)

Things to note with the internal sid instruments:

- There is no vibrato and slide is literally a coarse pitch slide.
- Table offsets are automatically found in the init phase, so it's vital those tips above are used. :)

12) Writing songs directly in assembly

I assume nobody wants to do that. I can write how to do it. No? :)

13) To do list:

Converter:

1) Add effects (for a start, proper note off for FM instruments, though internally there doesn't appear to be one)
2) Add more formats to XM2C64 (Sid Wizard, 2sid & 3sid support)
3) Port both converters over to BlitzMax so they can be compiled on other formats.

Driver:

1) Add frame tables for more complex FM instruments.

14) Thanks:

- MrSid for the default sid frequency tables, taken from a Codebase article.
- My beta test team, especially Mermaid for hardware testing.
- Cameron Kaiser for the FM programming info on his website: http://www.floodgap.com/retrobits/ckb/secret/

15) Notes:

Incidentally the original version of this driver was written in 2003 for 4kb intros, using just pure SID. I tweaked it recently to use in Razor 1911's “Best Intro Ever” and at the same time expanded it to try out the FM cart. I'll probably give the driver a tidy-up because code from that long ago isn't very sustainable, and the FM extensions are mostly JMPs out of the main driver.

16) Change-list:

v1.0 - Initial release.

v1.1 - FMSIDXM only recognised XMs with FastTracker header, removed that bit of code.

v1.2 - If using own driver made compile ignore first three channels of XM, now you can use them for guide tracks without any more memory being used. Added first pass of "XM2C64" tool.

v1.3 - Added NTSC support for internal player.