

# Stefano 11



**“Nuvolau friend”**

**Vice snapshot with Vice palette**

**Made with the GIMP from a MB photo  
and converted to C64 320x200  
Hires Mode Bitmap  
by Stefano Tognon  
in 2006**

**“Water splash”**

...



**Free Software Group**

~~OSD~~ *in 11*  
version 1.00  
14 February 2007

## General Index

Editorials.....	4
News.....	5
SIDPlayerDS.....	5
Goattracker V2.51-V2.59.....	5
Ninjatracker V2.0-V2.03.....	6
SID Factory 0.05 (alpha 1) .....	7
SIDId V1.0x.....	7
XMPlay SID plugin beta 24d.....	8
HARDshit 2.0.....	8
Goattraker 1.53 stereo HARDSID.....	9
SOASC Collection.....	10
SID Compo 6.....	10
SidTool.....	12
HVSC update 46.....	13
XMMS-SID v0.8.0beta17.....	15
Laxity Interview!.....	16
Tiny Sid 2 (part 2).....	19
BLOCK ACID DUB.....	19
Resolution.....	26
Conclusion.....	29
SID Factory vs Ninjatracker.....	30
SID Factory.....	30
Ninjatracker.....	35
Comparison.....	37
Conclusion.....	39

## Editorials

Stefano Tognon <[ice00@libero.it](mailto:ice00@libero.it)>

Hi, again.

It's incredible how the free time is always so little and when a problem pop up, the solution can kill it at a big rate. For me, an Internet connection problem in December due to a very low quality cable signal in my country, cause 15 days of no connection, irrelevant telephones call to the provider that does not resolve the problem, wire cable testing in the home, and finally only, *only*, by reprogramming the modem to produce a non standard tone, I get it connect again but with a low, low quality rate.

Actually it takes me 15 minutes of trying only for sending or receiving a mail, with lot of modem hangup (even if I program it to terminate connection only if the provider is...exploded by a nuclear war...), or having dialing plane signal for minutes. Sometimes the signal is stable for 30 minutes or so, and it is for this that you are reading the magazine (so I'm able to upload it), but you have to stress yourself for minutes by trying and trying before find this situation.

With this problem, the first article about the second part of Tiny Sid Compo 2 is not completed for the 4 players I planned to discuss, so I split it and another part will be written in the next number.

The second article is about a comparison of two editors: SidFactory and Ninjatracker, that as you will read in the news, were released in the same week.

Finally, this is the period I traditionally run the Tiny Sid Compo, but this time I don't start it as I cannot grant to have all the time it could took me. Maybe I will revamp the Big Sid Compo instead as having a more longer duration could be better managed.

However, don't blame for the compo not running, my time for sid activity will be shared with HVMEC (I have lot of new material that I'm adding to it), some tunes ripping, new tunes, and next number of SIDin, of course.

Bye  
S.T.

## News

Some various news of players, programs, and competitions:

- SIDPlayerDS
- Goatracker V2.51-V2.59
- Ninjatracker V2.0-V2.03
- SID Factory 0.05 (alpha 1)
- SIDId v1.0x
- XMPlay SID plugin beta 24d
- HARDshit 0.2
- Goatracker 1,53 stereo HARDSID
- SOASC Collection
- SID Compo 6
- SitTool
- HVSC update 46
- MMS-SID v0.8.0beta17

## SIDPlayerDS

On August 2006 a port for Dreamcast of Christian Bauer's SIDPlayer where done by Troy Davis(GPF). Original SIDPlayer is made from Frodo emulator and use SDL library, so It runs in lot of systems (Linux, Beos, Windows, ecc.).

Look at <http://www.neoflash.com/forum/index.php/topic,3034.msg21494.html#msg21494> thread or go to the homepage <http://gpf.dcemu.co.uk>

## Goatracker V2.51-V2.59

New version of Cadever's PC tracker is available:

### v2.51

- Fixed packing of empty patterns when not using any effects.
- Fixed differing gatetimer value in instruments causing playback going out of sync.
- /G command line option no longer has to keep gatetimer values the same.
- Added high bit of gatetimer value to control whether wavetable execution starts right on the note init frame. If used, will cause significant rastertime increase.

### v2.52

- SHIFT+ENTER in orderlist view takes the next available pattern if pressed on a repeat or transpose command.
- Added SHIFT+SPACE in pattern editor to start playback on a specific pattern row.

### v2.53

- Fixed F3 to always play the currently visible patterns.
- Added BACKSPACE in the orderlist editor to set playback end position.
- Only effective when starting playback from position, not beginning.

### v2.54

- Fixed SHIFT+F3.
- Changed SHIFT+, . to update the pattern view.
- Changed SHIFT+SPACE to remain in pattern playback mode, if it was active.

v2.55

- Added BACKSPACE in the fileselector to go to the parent dir.

v2.56

- Song filename cleared on songdata erase.
- Added /F to set custom SID clock frequency.
- Graphics output routines no longer compare text screen buffer to previous to find out if it should be updated.

v2.57

- Added alternative hardrestart method & playroutine that is used when HR attack parameter is at maximum (FF00 or F800 for example) - this can in theory give better reliability.

v2.58

- Cleanup, removal of a few questionable features:
  - \* Gatetimer high bit to start wavetable immediately
  - \* Guard 1stwave parameter

v2.59

- Songformat version update (GTS5/GTI5)
- Gateoff timer parameter bits control now hardrestart disable (\$80) & gateoff disable (\$40).
- 1stFrame Wave can be any from \$01-\$8F, \$00 to leave both waveform & gateflag unchanged, or \$FE / \$FF to control gateflag but leave waveform unchanged.

Download at <http://cadaver.homeftp.net/tools/goattrk2.zip>

## Ninjatracker V2.0-V2.03

Released on August and September 2006 the new improved version of Cadaver C64 tracker.

- V2.0 Changes to previous versions include commands (also used as instruments), 2-column tables and a slide function that stops at target pitch. As before, allows to save both normal executable musicdata and gamemusic data without the player.
- V2.01 has improvements in the editor and in the sound effect playback of the gamemusic player.
- V2.02 has the same editor features as V2.01, but has 2-frame hardrestart and reduces playroutine zeropage use to 2 bytes. However, the new playroutine is slightly slower and bigger.
- V2.03 has "hifi" style hardrestart and more optimized playroutine. Note duration range has also been changed to 3-65, but there are no limits on the duration of a pattern's last note, like in previous versions.

```
NinjaTrackerV2.02 Sng0 Oct3 00 01 Ras9/C
Trk 1 Trk 2 Trk 3 Patt01 D0000/1024 Ps2E
00 C0 00 C0 00 C0 00 D-1 01 64 filt long
01 01 01 02 01 04 01 +++ -- --
02 05 02 02 02 07 02 +++ -- --
03 06 03 02 03 0B 03 +++ -- 16
04 08 04 03 04 0C 04 AH#1 9C 08 slide 008
05 09 05 02 05 04 05 +++ -- 40
06 08 06 02 06 13 06 +++ A2 64 slide 3c0
07 0A 07 02 07 14 07 +++ -- --
08 05 08 03 08 0B 08 +++ -- --
09 06 09 0D 09 0C 09 +++ -- --
0A 0F 0A 0E 0A 00 0A G-1 04 -- filt fast
0B 0F 0B 02 0B 00 0B +++ -- --

Wave Tb1 Puls Tb1 FilT Tb1 Commands
01 41 00 01 80 20 01 91 40 01 filt long
02 FF 00 02 68 02 02 02 7F 01 CCA 01 01 01
03 81 C4 03 70 FE 03 7F FF 02 rhythmgr
04 41 00 04 08 02 04 FF 02 0067 03 06 00
05 FF 00 05 FF 02 05 91 C0 03 rh.gtr np
06 81 C8 06 80 40 06 FF 03 0067 03 00 00
07 41 n+ 07 60 01 07 -- -- 04 filt fast
08 41 n+ 08 0F FF 08 -- -- 00CA 01 01 01
```

- <http://covertbitops.c64.org/tools/ninjatr2.zip>
- <http://covertbitops.c64.org/tools/ninjatr201.zip>
- <http://covertbitops.c64.org/tools/ninjatr202.zip>
- <http://covertbitops.c64.org/tools/ninjatr203.zip>

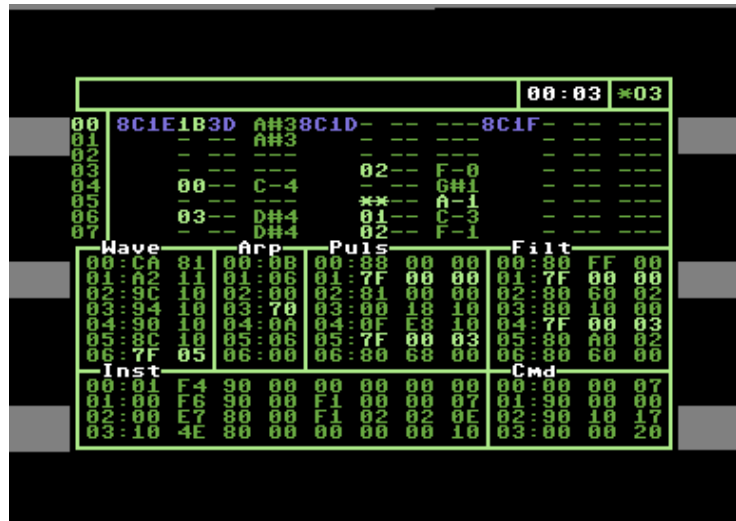
## SID Factory 0.05 (alpha 1)

SID Factory is a new editor system, developed by Laxity of Vibrants/Maniacs of Noise. The editor has inherited a lot of features and look from the JCH editor. It's been programmed from scratch and is very different from the JCH editor behind the surface.

The tracker made use of loadable players:

- v5: multispeed
- v6: low rasterline

Download the staff from <http://www.m-studios.dk>



## SIDId V1.0x

In September Cadaver has released a small DOS utility letting you know which playroutine is used in a SID file. You can add playroutine signatures manually to sidid.cfg.

### V1.0

- Original

### V1.01

- AND function added

### V1.02

- Multiscan added

### V1.03

- Listing of unidentified files added
- Scanning of all files added

### V1.04

- Added searching only for specific player

### V1.05

- Added option to not recurse subdirs

### V1.06

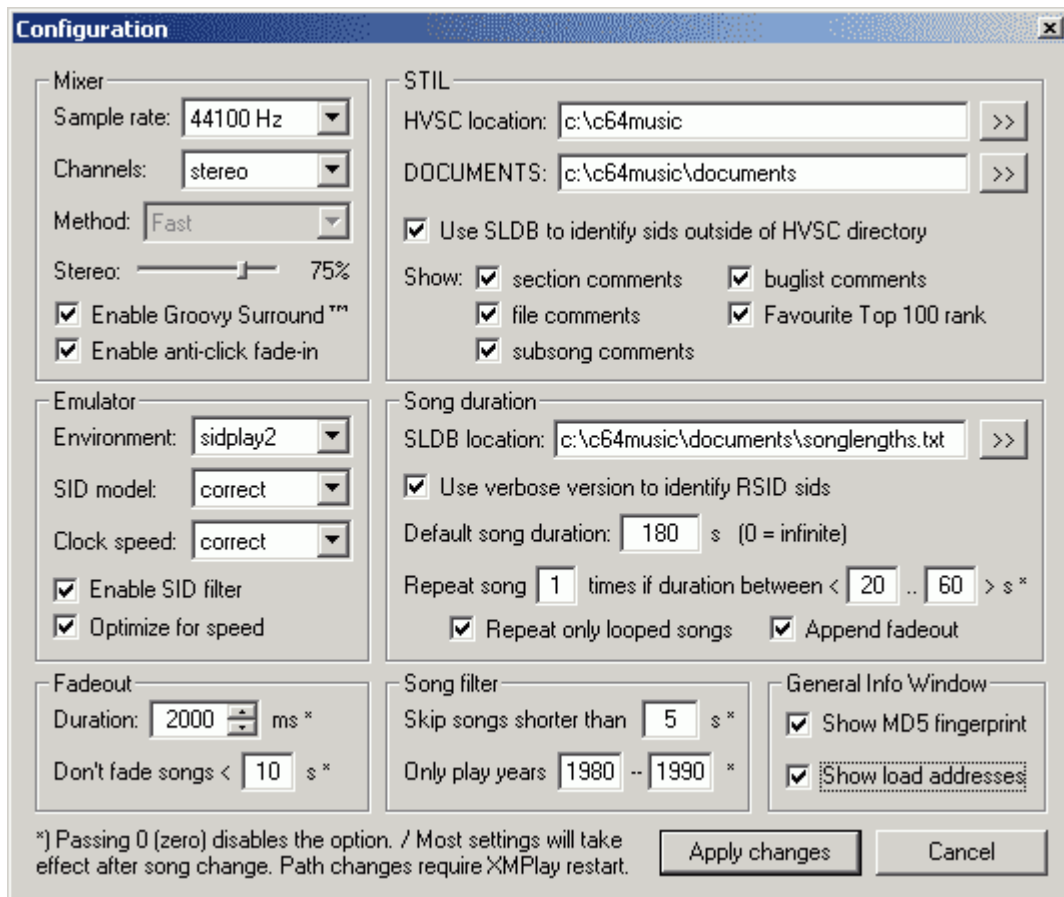
- Directory to scan can be given as an argument
- Added option -c to specify the configfile
- Added option -? to show usage information
- Added environment variable SIDIDCFG to specify the configfile

Download at: <http://cadaver.homeftp.net/tools/sidid.zip>

## XMPay SID plugin beta 24d

Released in October a new version of the sid plugin for the XMPay player:

- fix for memory leak in STIL loading
- memory optimization for SLDB loading
- sids with all songs filtered out are now rejected
- removed double displaying of start song, song numbers, song lengths if weren't changed (General Info window)
- added displaying amount of memory needed for STIL and SLDB in About window



Download at: <http://www.pieknyman.tk/>

## HARDshit 2.0

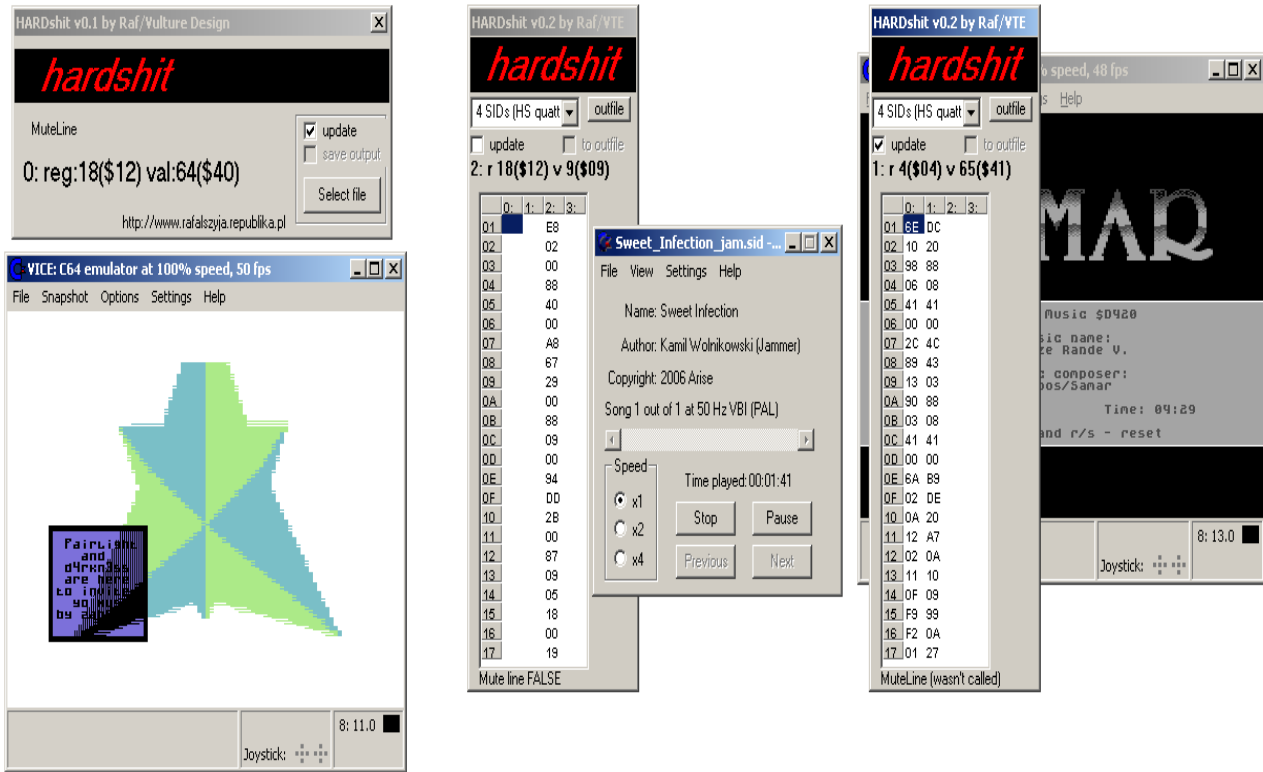
This is a fake hardsid.dll allowing realtime monitoring SID chip(s) writes. It is created by Raf of Vulture Design and was released in November.

version 0.2 changelog:

- output text have additional mark - with hardsid's deviceid.
- supports up to 4 sids (e.g. you can monitor stereo in VICE)
- realtime updated table with writeable SID registers

The main site is at <http://www.rafalszyja.republika.pl> and download the tool from <http://non-ame.c64.org/csdb/getinternalfile.php/34346/hardshit02.rar>

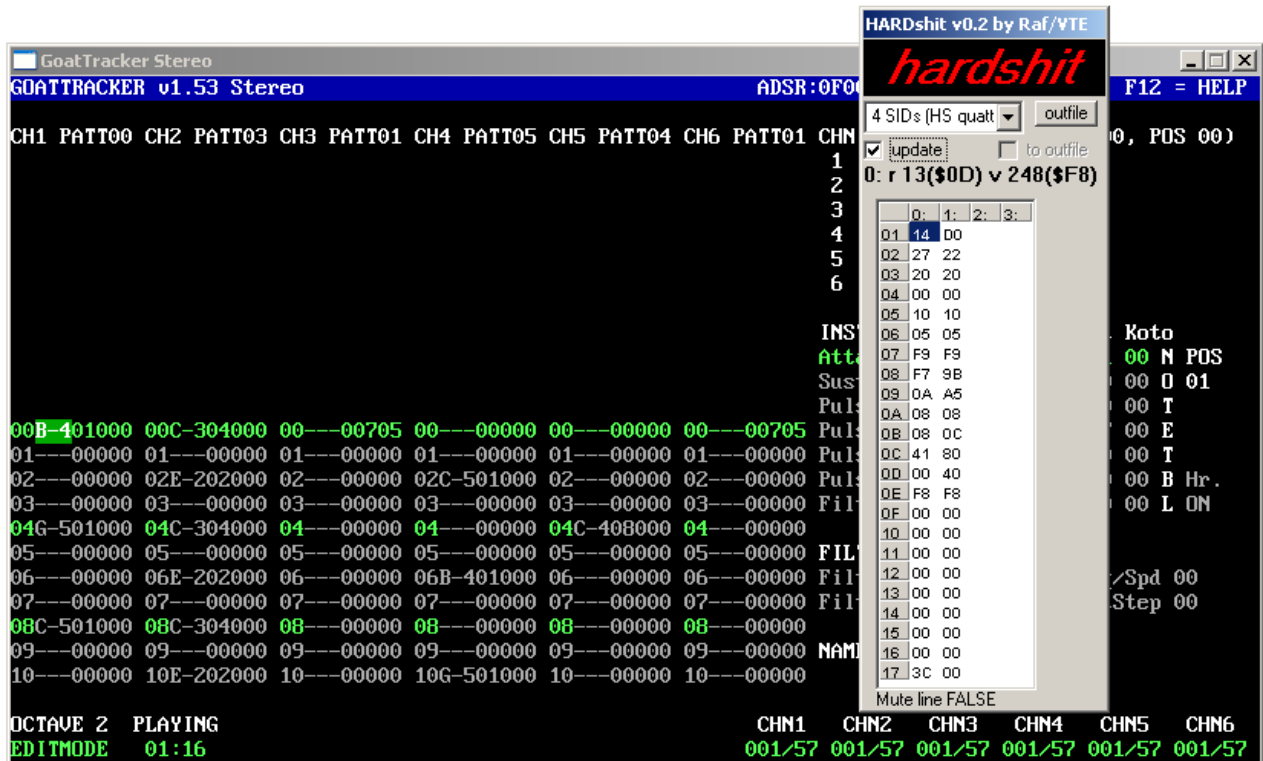




## GoatTracker 1.53 stereo HARDSID

Raf even release in November a modified version of GT 1.53 stereo for Harsid card.

Download from <http://noname.c64.org/csdb/getinternalfile.php/34366/gt153sterohardsidwin32.rar>



## SOASC Collection

The SOASC project is an automated recording technique invented by Stone Oakvalley in order to mass record music from the Commodore 64 and its SID chips (6581 and 8580).

It records all Sid from HVSC (about 93336 tunes) in both 6581 and 8580 chips without looking at what chip the tune is intended to be listen to. The recording format is in MP3 with 224kbps, mono, 44100Hz.

The final size of collection will be something like 300GB.

Look at <http://www.6581-8580.com/>



## SID Compo 6

The annual Sid Compo by <http://www.c64.sk> was done in ending of the year and result were available before Christmas.

Public voting results... (49 votesheets)

1. Arctic Circles <i>by Dane</i>	1269
2. I swore a vow on my dying breath <i>by Linus</i>	1227.5
3. Albino Human <i>by Fanta</i>	1205
4. Critical Grid Voltage <i>by CreaMD</i>	1167
5. Misunderstandings <i>by Zabutom</i>	1122
6. Teh Disco <i>by Jammer</i>	1119
7. everything seems 2 be alright <i>by Randall</i>	1074.5
8. Electric Jesus <i>by Intensity</i>	1062
9. quickndirty <i>by Syndrom</i>	1048.5
10. Laserflake <i>by Monk</i>	1043
11. Delight <i>by Nata</i>	1037
12. White Light <i>by A-Man</i>	970.5
13. Autumn Memoir <i>by Psycho</i>	968
14. a Tune <i>by Hein</i>	938
15. Kollision <i>by Maktone</i>	932
16. Triskelion <i>by Asterion</i>	926
17. Tuesday Weirdness. <i>by NO-XS</i>	924
18. Illumination <i>by Surgeon</i>	921
19. Shortcut <i>by Dr. Voice</i>	902.5
20. _hidden (XeO3 Main Theme) <i>by Luca</i>	868
21. Mountain Morning <i>by Sidder</i>	859.5
22. INCD418 <i>by Richard Bayliss</i>	852
23. Carrier Lost <i>by RaveGuru</i>	834.5
24. c606 <i>by Alex B</i>	826
25. Breath <i>by Rusty46</i>	790
26. Lazy day <i>by Rayne</i>	786
27. alt.nerd.obsessive <i>by Lordnikon</i>	759.5
28. Summer SID <i>by Uneksija</i>	732.5
29. Paisley <i>by Dalezy</i>	693
30. Ex Inferis <i>by Cadaver</i>	685.5
31. Zuo3 <i>by Raf</i>	650.5

32. City Waltz by Peter Bergstrand	457.5
33. Random Acts of Cruelty by Mermaid	434
34. Break beep by Vitnaque	414.5
35. @yourface by Josstintimberlake	370.5

Jury Rank:	PTS
1 Arctic_Circles	157
2 Critical_Grid_Voltage	128
3 I_swore_a_vow_on_my_dying_breath	127
4 Albino_Human	111
5 Electric_Jesus	95
6 Teh_Disco	88
7 everything_seems_2_be_alright	65
8 quickndirty	58
9 a_Tune	55
10 Triskelion	49
11 shortcut	48
12 Carrier_Lost	44
13 Delight	43
Autumn_Memoir	43
15 kollision	33
16 Mountain_Morning	31
17 Laserflake	29
18 _hidden	23
19 lazy_day	22
20 paisley	21
21 INCD418	20
22 alt.nerd.obsessive	19
23 c606	17
24 Summer	15
25 exinferis	9
zuo3	9
27 @yourface	7
28 Break-beep	2

Those authors didn not vote as jury	
misunderstandings	85
White-Light	51
Tuesday_Weirdness	44
breath	34
Illumination	31
Random_Acts_of_Cruelty	6
City_Waltz	5

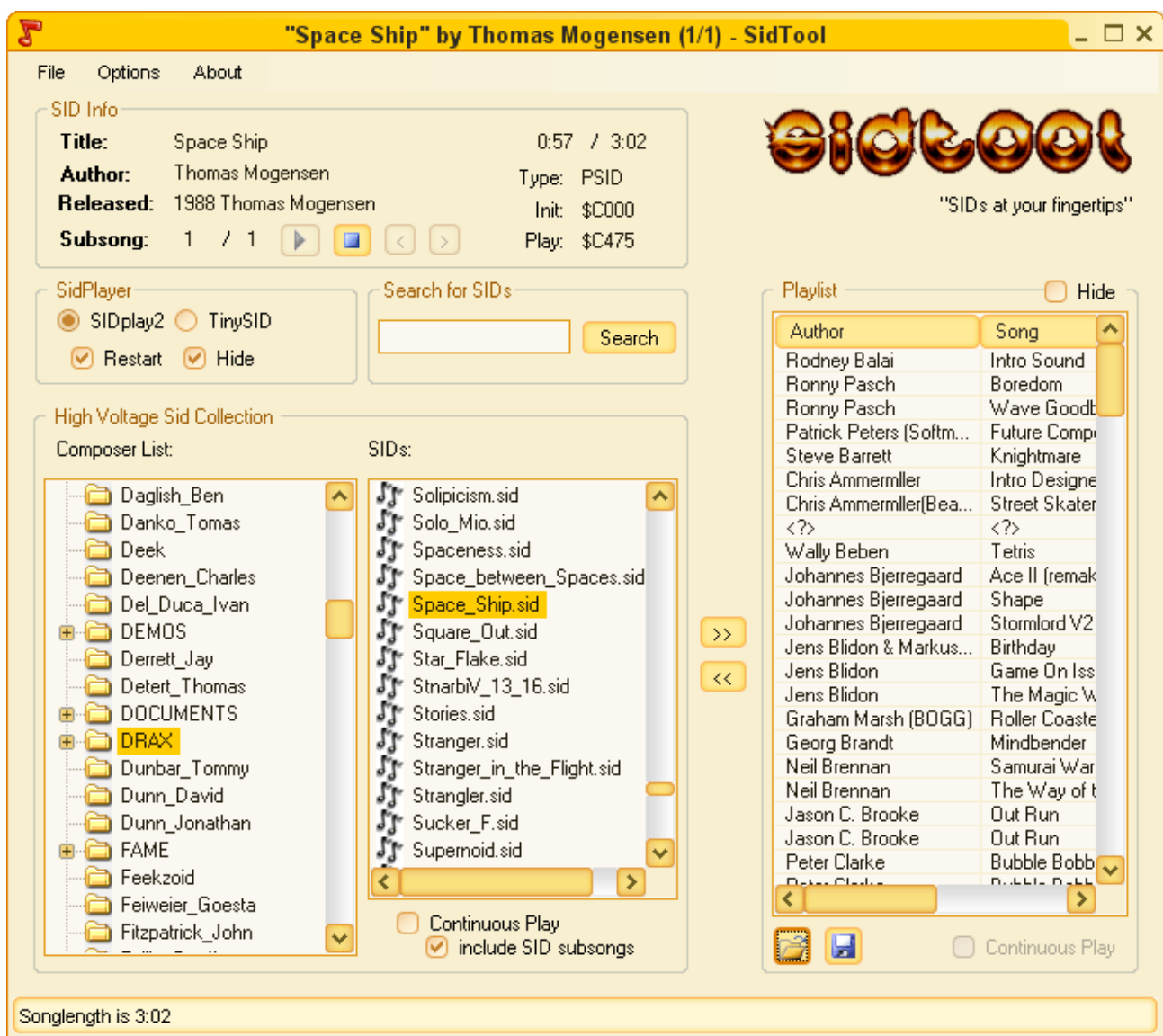
more at: <http://www.c64.sk/index.php?content=article.php&articleid=115&id=4075>

# SidTool

SidTool is a SID frontend developed by Madman that has:

- Support Sidplay2/w for playback
- Support TinySID for playback
- Easy access to the entire High Voltage Sid Collection
- Easy single-click interface
- Songlength database support
- Continuous play to discover SIDs unknown to you
- Playlist support with ability to add certain subsongs
- Experimental sid2midi support (sid2midi.exe not included)
- Built in search function
- Build on .NET framework 2.0 (required)

look at <http://sidtool.drool.de/>



## HVSC update 46

High Voltage SID Collection Update update #46 was released in January 21, 2007

After this update, the collection should contain 33,000 SID files!

This update features (all approximates):

1014	new SIDs
71	fixed/better rips
2	fixes of PlaySID/Sidplay1 specific SIDs
13	repeats/bad rips eliminated
664	SID credit fixes
160	tunes assigned a sidmodel flag
24	UNKNOWN demo tunes identified

As usual, the update and the all-in-one packages are available from <http://www.hvsc.c64.org>

This update features the tunes from the following parties:

X'06, c64.sk Sidcompo 6, Big Floppy People 2006, Breakpoint 2006, North Party 10, Primary Star 2006 and of course all the tunes from the 10 Years HVSC music collection.

Main Composers featured in this update: (Artists marked with NEW are either completely new to the HVSC or they get their own directory in this update)

- x **Alien/WoW**
- x **A-Man** -> with previously unreleased stuff exclusively available in HVSC
- x **Art of Noise**
- x **Richard Bayliss** -> we've dug up 67 new tunes from him, with his kind help
- x **Arman Behdad** -> the Trance master is active again, check out Electric\_Jesus.sid
- x **Glenn Rune Gallefoss** -> he kindly packed up some unreleased previews for us
- x **Chapelier** -> one of the guys behind [www.c64.com](http://www.c64.com), he sent us his complete works
- x **CRD** -> NEW! A promising new sid musician enters the scene!
- x **Dalez** -> still going strong with mellow minimalistic melodi
- x **DRAX / Maniacs of Noise** -> that old melody master just won't stop composing candy for our ears!
- x **Ed/Wrath Designs** -> the Aphex Twin of the sid scene is back with an activity peak
- x **Fanta** -> some fantastic goodies in there again by Onkel Fanta!
- x **Ghormak** -> NEW!
- x **Hein Holt** -> One of the multitalents of the 64 scene: He pixels, he codes and he composes brilliant music
- x **Jac** -> old Riffs hero from the 80s, good oldschool stuff
- x **Jammer** -> always pushing the sid to new limits. This time with special focus on analog speech synthesis (no sampled digis, everything made "by hand"! His HVSC.sid is probably the best tune \_ever\_ in that department
- x **Josstintimberlake** -> another avantgardistic sid musician brings us more data vibes
- x **Laxity** -> yeah, you read correctly, the old famed composer is churning out blips and blops again, as if it was 1989!
- x **Linus** -> bringing us more follin'ish gangsta tunes! :-)
- x **Maktone** -> with some good mood party tunes again, hooray!
- x **Adam Morton** -> NEW!
- x **Markus Müller** -> Superbrain sent us some old GCF tunes, cheers mate!
- x **Ne7** -> composer of the DTV Hummer game soundtrack, and generally a very sophisticated musician (Electronat will blow you away!)
- x **Psycho**-> ventures into the deep trancey department
- x **Radiantx** -> A comparably new sid composer brings us some complex and dense oldschool-ish tunes
- x **Randall** -> As usual MC Randall can't miss to place at least one or two smash hits in every

- HVSC update. Very Polished stuff, in every meaning of the word.
- x **Rio** -> NEW! Another youngster in the club of C64 composers.
  - x **Rupert Dissident** -> NEW! Another one of those home composers who just pop up now and then and ask us if we have use for some twenty+ unreleased tunes. Pretty good stuff he did, placative and complex.
  - x **Rusty46** -> We bring you 52 tunes from this swedish guy, all done in 2005-06. Innovative elements coupled with catchy hooklines!
  - x **Sidder** -> Smart arrangements, unusual instruments - Sidder is always exploring different paths, from conventional to experimental.
  - x **Stainless Steel** -> Joe cooked up some brilliantly arranged remixes of recent pop hits for us.
  - x **Jeroen Tel** -> Jeroen once again opened his treasure chest of unreleased tunes and gave the scene three "new" songs. But we also found a bit of old and forgotten stuff from him.
  - x **Xiny6581** -> Another one of those rather new sid musicians, but a valuable enforcement. Give these tunes some time, they don't release their message in the first 3 chords.
  - x **Zabutom** -> Ha, only about a year ago he entered the stage and now he's working the sid like an orchestra that was made just for him! Wow!

#### Fundamental directory structure change

In this update the HVSC Crew decided to fundamentally overhaul the HVSC directory structure and re-sort it.

There are a few disturbing things in the current HVSC directory structure:

1. Inconsistencies to our own rule as to what is root and what is VARIOUS. There are composers in VARIOUS that, according to our rules should be in the root and vice versa.
2. With more and more composers surfacing and others getting their own directories by gathering their 3 tunes, the directory trees grew to huge sizes (several screens high), which made navigating and maintaining a bit cumbersome.
3. The term VARIOUS and the fact that the tunes inside that directory were 3 directory levels further down than the glorious root composers was considered unfair by a lot of people. Discussions went in circles thinking about a better name (SCENE? POST\_1993?) or to organize the root dir similiarly (CLASSICS? PRE\_1993?), so we finally decided to go for a completely new sorting structure:

```
/DEMOS
/0-9
/A-F
/G-L
/M-R
/S-Z
/UNKNOWN
```

-> Sorting structure is now similar to the /GAMES directory. No changes to the rules were made as to what goes into DEMOS. It's still the same, it contains tunes from demos made by composers that are either unknown or that have composed less than 3 tunes overall.

```
/DOCUMENTS
-> No changes here
```

```
/GAMES
/0-9
/A-F
/G-L
/M-R
/S-Z
```

-> Again, nothing changed apart from the added /0-9 subdirectory. The GAMES directory still contains tunes from games made by composers that are either unknown or that have composed less than 3 tunes overall.

/MUSICIANS

/0-9

/A

/B

/C

-> Basically we've merged the root directory and the old VARIOUS directory giving up the A-F/G-L/M-R/S-Z sorting and going for a plain alphabetic flat sorting structure. A composer still needs 3 or more tunes to get into the MUSICIANS directory, in order to avoid cluttering up HVSC with hundreds of directories that only contain one or two tunes.

We think we found an unbiased and futureproof solution with that new sorting structure: It's easier to maintain. It's easier to navigate. It's more fair: No more discussions about "why is composer x in the root, shouldn't he be in VARIOUS?" or "composer y made 3 gametunes before '93, shouldn't he be in the root?". In addition, an almost flat structure like that hopefully encourages one or the other guy out there to program a proper frontend for HVSC one day. Be it in form of a tool or a website interface, so people can conveniently switch between several HVSC "views", assign "favourites" flags to tunes and composers etc. If you've got the skills and the time/motivation to help us in making this vision reality, drop us a mail at [hvsc@c64.org](mailto:hvsc@c64.org). Especially a motivated webdesigner with PHP/MySQL knowledge would be a welcome addition to the team.

HVSC has grown almost too big to jump around in Windows Explorer without getting lost, so we hope you get used to the new structure and like it just as we do. There were concerns that people new to HVSC would get lost in the new structure as they are usually searching for their favourite game tunes from back then, which basically boils down to: Hubbard, Galway, Follin and Tel. ;) We think that people will still find what they're looking for, only it takes them a little longer than before. And who knows how many gems they may find on their hunt for that old tune by randomly clicking into other composers directories, which were previously buried in VARIOUS.

## XMMS-SID v0.8.0beta17

New version of XMMS-SID, a plugin for UNIX audio player XMMS, was released in end of January:

- Structural cleanup of the code was made; some previous artificial limitations were removed in the process.
- Dynamic allocation is now used in all structures of SonglengthDB and STILDB, which should improve memory usage efficiency.
- Documentation was updated and extended. Code was re-indented and commenting was improved.
- Internationalization support via GNU gettext was added, with initial (but largely unfinished) Finnish translation.
- Some portability bugs were fixed, including a non-aligned memory access crash on certain non-x86 platforms.

Download from: <http://www.tnsp.org/xmms-sid.php>



## Laxity Interview!

by Stefano Tognon

This time I go to interview Laxity, a old composer that I think you already know, as it is a member of Vibrants and Maniacs of Noise groups. He recently started to compose again with sid. The interview was done in January by mail.

***Hello, could you introducing yourself and what you do in real life?***

My name is Thomas Egeskov Petersen. I'm 33 years old, married and have two children, a boy and a girl. I live near Aarhus in Denmark, where I work as a game programmer. I have worked as a programmer since 2001, where I was hired to program a game on the GameBoy Advance. Since then I've worked with games on PC/X-Box, Nintendo GameCube and presently mobile phones (Java.. yak!).

***You started composing in C64 where you were younger using Hubbard player. Wasn't this a difficult task (and maybe a good way to learn coding)?***

Yes, I started out making music that way. It was relatively difficult, but not an experience that taught me coding but merely a basic understanding of how pointers work etc.. At the time I had pretty basic coding skills, so I was able to figure out where pointers were read, parsed and so on and so forth. The whole thing was mostly a deductive process - basic learning by doing in the context of trial and error, I'd say. The most difficult was actually making the music, since I had no experience with that and found it rather difficult to make it sound as I imagined it.

***You code (and you are still coding) many music players (and editors), would you tell us something about all your work in this field?***

There's so much to tell. I think what drives me on the c64 is the fact that I'd like to not only do the music, but the whole thing. It satisfies me to both program the driver AND compose the music, as I somehow regard that as the full package of making music on the c64. Thereby not saying that those who "only" use tools by other people aren't doing it right - I'm merely trying to say that I enjoy doing the whole thing.

Looking at my source files I've identified 10 sound drivers for the c64 that I've written over the years, where the first 4 were done in the period of 87-91 (3 publicly used). There rest of them we programmed after 2005. One is the JCH player (which took a long time to do), 3 for SID Factory and 2 for the Tiny Sid Competition Challenge last year (2005).

In the period of 1991 to present I've also programmed sound drivers for the Amiga, GameBoy, PC and GameBoy Advanced, which all (except the PC one) have inherited a lot of structure and idea from the basics done with the c64 sound drivers. The GameBoy Advanced driver was probably the most complex of them all, and done in a hurry while I programmed my first game.

Most recently I've programmed my first ever music tracker. I was motivated to program it while I was developing the music driver for the JCH editor. Therefore SID Factory, as it is called, takes off in the JCH Editor platform - and the idea was to take it a bit further. I've basically tried to elaborate on the driver/editor integration protocol and enhanced tracks to have an instrument column and a command column instead of just one unified one. Those were the basic idea for SID Factory, and has proven to work pretty well as I can create table setups defined in the drivers, instead of having to write a driver that uses tables as defined in the editor. (Makes sence?.. hmm)...



### ***What are your plans for the future of SidFactory? New features?***

Well, I don't really have big plans for it, as it is mostly a spare time project (spare time, which there's VERY little of actually). When I released the version 0.5 of SID Factory, there were some brief comments on CSDb, where most of them were concerning missing features, such as STOP command in the tracks, multiple tune support, etc.. Especially the latter is a bitch to do, but I will eventually have to do it, otherwise I'll have another stranded project on my HD. Besides that, some editing features were requested, which I ought to implement. Especially the nursing features, such as press-button-while-on-pointer-move-cursor-to-position-in-table-pointed-to seem to be what some people want. A bit annoying actually, and initially I thought I wouldn't implement that. But I think I have an idea which would work on commands too, so that if a command is of a type that point to a location in a wave table, hitting the magic-button will transport the cursor to correct location in the wave table. etc. etc., all without structural changes to the driver/editor integration.. But it's hard to say when I'll have the time to do those changes. At the present I have a version 0.6 which features some changes to the driver/editor integration, and I'd rather like to have all eventual changes done before I release another version, so it's not too much of a bother to users (if there are any at all. :).)

### ***What do you think about speech synthesis using sid voices?***

I think it's interesting, but not a thing I have any experience at all. Sometimes it sound really cool, but it's not always that impressive.

### ***Now some quick final (standard) questions:***

#### ***Real machine vs emulator: what do you think about?***

Well, since the emulator is by definition trying to emulate the real thing and not vice versa, I'd have to go for the real thing - since it's the real thing :)

#### ***6581 vs 8580 chip: any (musical) preference?***

I like to use the 6581 myself, but a tune written for the 8580 I prefer to listen to on the 8580 and ditto with tunes written for the 6581.

### ***What is the worst and the better sid you composed?***

I think I wrote a whole bunch of awful sid tunes, but I did some conversions like: James Bond, The Pink Panther and I wanna dance with somebody, that suck big-time.

### ***Who are your best sid authors?***

I think there are a lot of good sid composers. I'm pretty fond of the stuff composed by Hein Holt - he's really good and doesn't do that much mainstream stuff. I like that a lot. Goto80 sometimes rocks my boat. He's really good with rhythm and sound design. Aleksii Eeben is a really good composer, also alternative - a brilliant guy. I've bumped into him numerous times over the years, doing GameBoy Color, GameBoy Advanced stuff and now c64.

From the "classic" c64 composers, my favourites are probably to be identified as the usual suspects. Hubbard (all time favourite), Galway, Fred Gray - those sorts of composers.

***What are the best sids ever in your opinion?***

I'd have to pick some and be concerned that some are missed out that deserve to be mentioned too. Here are a few I especially like for one reason or another, presented in no particular order:

Master of Magic by Rob Hubbard

The Way of the Exploding Fist by Niel Brennan

Myth Demo by Johannes Bjerregaard

Wiz by Rob Hubbard

Daley Thompsons Decathlon By Martin Galway and David Dunn

Crazy Comets by Rob Hubbard

Body Slam (tune 4) by Tim Follin

Legend of Kage by Fred Grey

Ghouls'n Ghosts (tune 1) by Tim Follin

Implosion (tune 1) by Fred Grey

***Finally, many thanks for the time you give for this interview, and now would you say something else to the our readers?***

Any time and thanks for your interest, Stefano. Thanks for reading, still nice to have a c64 family out there.

## Tiny Sid 2 (part 2)

by Stefano Tognon <[ice00@libero.it](mailto:ice00@libero.it)>

This is another part about the Tiny Sid 2 compo. Here I go to show you two more entries, looking at their engines. Next time we will see the last 2 entries.

### BLOCK ACID DUB

This is the 256 bytes entries by Frantic/Hack'n'Trade and the player is well documented with comments into the source code, so here I go to sum its features and working.

- It starts up at 326, by setting the IRQ address, and then disable (SEI) the interrupt.
- Notes frequencies are used only with the high value of frequencies byte (low value is never set, and so it is 0). It was only used notes that give less error.
- Attack/Decay is never used for voice 1 and 2 (so it is 0)
- Use of undocumented instructions SBX, ANC & ASR for save space
- Use of hardrestart for drum
- Use random sequences for made the tune more longer and different
- Use only SID data values for both instruments and notes that are fitted together

The later point need some further comments.

The pattern of music data for each voices is done by a byte with this format:

<i>Bits value</i>	<i>Description</i>
ddd.pppp	ddd=Duration ppppp=Pointer to probability "nodes"

Probability node is a group of 4 bytes that contain a sequence (instrument/note) to play and it is random selected as soon as a pattern is processed. Each of this value is a pointer to the instruments/notes declaration. Look at this table:

<i>Bits position</i>	<i>Description</i>
00000000	Delay of one frame
10000000	Set Sustain/Release
01000000	Set Attack/Decay
00100000	Set Control of voice
00010000	Set pulse high value
00001000	Set pulse low value
00000100	Set frequency high
00000010	Jump to another instrument
00000001	Stay:skip pointer update (=> end of instrument)

Those bits can be combined together and then a byte for each bit (but not for Stay) is follow in the sequence.

Here some examples, with a single note, an arpeggio (look at the Jump instruction) and a Bass that have a noise in the first two ticks:

single note to play:

```
.byte    SID_FRQHI | STAY
.byte    $06
```

arpeggio:

```
.byte    SID_FRQHI
.byte    $0c
.byte    SID_FRQHI | JUMP
.byte    $09, arpeggio
```

Acid Bass 2:

```
.byte    SID_SR | SID_CTRL
.byte    $b8, $81
.byte    TICK
.byte    SID_CTRL | STAY
.byte    $20
```

Now the source code. It is interesting that you look at the point where random sequence is taken using \$DC04 values. Here is when lot of undocumented instructions are used to limit the choose for 4 patterns.

```
*****
;
; Title: "BLOCK ACID DUB"
;
; Author: Frantic/Hack'n'Trade
;         alias Glenn Again/Kommando Knorr
;
; Size: Code + data + load address = 254 bytes
;       Two bytes reserved for the file system,
;       so the tune will fit in a single disk
;       block.
;
; Chip: 6581! (PAL)
;
; Format: DreamAss was used to assemble this source.
;
*****
; Last Tiny SID compo I aimed to make a long composition with harmonies and a
; "real" melody, rather than creating advanced SID sounds. This time I aimed for
; fatter sounds instead. Having explicit or implicit functionality for hardrestart,
; arpeggio, wavetables and so on. The player is capable of quite flexible creation
; of sounds. In fact, any combination of SID register values each frame is
; possible (but only one value per frame, and excluding lobyte of frequency).
;
*****
; Comments on some of the methods used to decrease size:
;
; - Using "random" recombinations of one single data sequence to make the tune
; somewhat varied and "longer" (filter type is also changed over time). A
; timer value is used as "random number" and this causes the tune to appear
; different each time. This player concept also allows one to use only one
; stream of data for all channels (only one "sequence") that just loops all
; the time to avoid seq-tables and code for handling such things. The
; possibilities of recombination are, however, configurable, so there is a
; certain degree of control of what might happen. (Jucke cited som dub guru
; the other day. This guru said that one part of the brain follows what is
; steady in the music, and another part follows that which is fleeting and
; passing, and therebetween a tension arises. According to the guru, that is
; what constitutes dub. Hehe.. Anyway, I hope that description fits this
; dubish likkle tune of mine. ;)
;
; - I am only using notes which can be approximated by using SID_FRQHI while
; letting Sid_FreqLo remain set to $00 all the time. That is, using base note
; of $0400, $0800, $1000, $2000 and so on. See end of source for some more
; info on that. There is a html/javascript attached in this package which
; was used for frequency calculation. Just enter a 16-bit base frequency
; in the source of the script, and view the results in a www browser.
;
; - AttackDecay is never changed in Voc 1 & 2 and remain set to $00 all the time.
;
```

```

; - Using illegal ops whenever suitable (and not just for the sake of doing it
;   of course). SBX, ANC & ASR are used here and they all do "two things in one"
;   apart from SBX, which is useful for subtraction with the X register, rather
;   than the A-register affected by standard SBC.
;
; - Not using different code structures for "notes" and "instruments". It's
;   all just SID data, and freq-setting can be included in the "instruments"
;   (in the case of a bass drum for example) or just be set on it's own, like
;   a "note".
;
;*****
;   GREETINGS TO NINJA/THE DREAMS, THE 252 BYTE PUNK!
;*****

; CONSTANTS

INSTPTR      = $02 ;= $03 +7 +7 (X offs is 1...)
TICKCOUNT   = $40 ;= $41, $48, $4f (X offs is 1...) ..is set to zero at startup.
SONGPOS      = $99 ;= is set to zero at startup
CTRLBYTE     = $60

NUMBEROFVOICES = 3 ;All three voices, of course!

;-----
; CODE

      * = $326

@binbegin:
      .word @start      ;Four byte init code to make it an executable.
      .word $f6ed

@start: sei

;--
; Main player loop

@wrast: cpx $d012 ;X=?? after init and X=$fb after loop.. both values ok..
      bne @wrast

      ;Loop once for each SID voice.
      ldx #((NUMBEROFVOICES-1)*7)+1 ; +1 because we skip the Sid_FreqLO register altogether. Freeing
one control bit.
@vocloop:

      lda TICKCOUNT+7+7+1;,x ;..but first, Filter Sweep - controlled by tickcounter from drum channel,
to achieve variable behavior.
      asl
      asl
      asl
      sta $d416

      ;Handle tick counter
      dec TICKCOUNT,x
      bpl @parseinst

;-----
;FETCH NEXT INSTRUMENT (PARSE NEXT PROBNODE)
@parseseq:
      lda SONGPOS
      inc SONGPOS
      and #%00011111 ;There are 32 seqdata bytes so just throw away the three upper bits to take
care of wrapping.
      tay ;A reg is set some lines up..

      ;Chose one of 4 possible instrument pointers
      lda $dc04 ;Get "random" value from $dc04
      .byte $2b, 3 ;ANC instruction. Same result as "and #3 / clc" in this case.
      adc @compositiongrid,y
      pha
      and #%00011111 ;Clear 3 msb
      tay
      pla
      lsr
      .byte $4b, %01110000 ;Illegal ASR (AND first, to remove instrument ptr bits, and then LSR result,
in one instruction.)

      ora #%00000111
      sta TICKCOUNT,x ;Store the new tick counter value
      lda @probnodes,y ;Get instrument number
      beq @nonewinst ;Instrument table pos 0 has the special meaning NO INSTRUMENT CHANGE (and
contains no instrument data).

      sta INSTPTR,x ;Store new instrument ptr..

@nonewinst:

;-----
;PARSE INSTRUMENT DATA AND WRITE TO SID HERE..
@parseinst:

      ;HARD RESTART - only for drum channel..

```

```

    ldy TICKCOUNT+1+7+7
    cpy #2
    bne @nohr ;Skip instrument read and use #2 as instptr instead.
    sty INSTPTR+1+7+7
@nohr:

    ;Fetch instrument data
    ldy INSTPTR,x
    lda @instdata,y ;Fetch ctrlbyte.
    sta CTRLBYTE

    stx @sidwlo ;Save voice offset directly in SID-write code.
    txs ;Store X in stackptr temporarily, for later retrieval.
    ldx #5 ;6 registers used for each SID Voice.
@parslp:asl CTRLBYTE
    bcc @nxtbyt
    iny
    lda @instdata,y
@sidwlo = *+1
    sta $d4ff,x
@nxtbyt:dex
    bpl @parslp

    tsx ;Restore voice pointer in X..

    ;Check for STAY parameter. If it's set, then just skip pointer update.
    asl CTRLBYTE
    bmi @noptrupdate

    ;Save updated instptr (either just increased or from a JUMP argument).
    iny ;This one points either to first step in next instrow or to the jump argument.
    tya
    bcc @nojmpfetch ;Skip Jump-destination fetch if jumpbit was set (carry set by the ASL
                    ;instruction above).
    lda @instdata,y ;Fetch instjmp argument
@nojmpfetch:
    sta INSTPTR,x ;Now points to next instrument row to be parsed..

@noptrupdate:
@loopend:

    ;Loopcounting (X) and filter type changes.
    lda SONGPOS
    .byte $4b,%11100000 ;Use ASR to "calculate" filtertype. Don't change filtertype too often, as a
                        ;straight AND would give.

    bne @dontforce
    lda #$10 ;Force lopass filter if no other filter type is set.
@dontforce:
    ora #$0f ;Set Global Volume (also used as AXS and-value) ;REMOVED....and
            ;always turn bandpass on
    sta $d418 ;Filter type / Global volume
    .byte $cb,7 ;AXS #7 or SBX #7 depending on who you ask.
    bpl @vocloop
    inx ;x becomes = $fb
    stx $d417 ;Set filter resonance an voice input to #$fb (doesn't matter that
            ;we also filter external input by this)
    bmi @wrast

;=====
; Composition Data
;
; Contains a table of pointers to "probability nodes". One row for each voice.

;Duration constants.
D1 = 0 << 5
D2 = 1 << 5
D3 = 2 << 5
D4 = 3 << 5
D5 = 4 << 5
D6 = 5 << 5
D7 = 6 << 5
D8 = 7 << 5 ;Max duration..

@compositiongrid:
;
;-----|-----|-----|-----|
;          |Voc1          |Voc2          |Voc3          |SeqStep|
;-----|-----|-----|-----|
; .byte D2 | @pbd,          D5 | @pabnt,  D2 | @pnote2; 0 #          |first row must set all three
channels.
; .byte
; .byte D1 | @psd,          D4 | @pabAAB          ; 1
; .byte D1 | @psd          ; 2
; .byte D1 | @pbd          ; 3
; .byte D1 | @pbd1,        D4 | @pabAB          ; 4 #
; .byte D2 | @psd,          D7 | @pabB1          ; 5
; .byte          ; 6
; .byte D2 | @pbd          ; 7
; .byte          ; 8 #
; .byte          D6 | @synx          ; 9
; .byte D2 | @psd          ; a
; .byte          ; b
; .byte D3 | @pbd          ; c #

```

```

; .byte D5 | @pnote ; d
; .byte ; e
; .byte D1 | @psd, D7 | @pabAB ; f
; .byte D1 | @pbd ;10 #
; .byte D3 | @psd ;11
; .byte D6 | @pabA1 ;12
; .byte ;13
; .byte D3 | @pbd ;14 #
; .byte ;15
; .byte D1 | @psd, D1 | @pabB1 ;16
; .byte D2 | @pbd, D6 | @synn ;17
; .byte D6 | @pabB1 ;18 #
; .byte ;19
; .byte D2 | @psd ;1a
; .byte ;1b
; .byte D3 | @pbd11 ;1c #
; .byte ;1d
; .byte D3 | @pabnt ;1e
; .byte D1 | @psd D2 | @pabAAB ;1f
;-----|-----|-----|-----|-----|
;=====
@probnodes: ;Probability "nodes" (4 bytes each) for instrument trig.
;Max 32 bytes in this table!

@pbd = *-@probnodes
; .byte @bd
@pbd1 = *-@probnodes
; .byte @bd
@pbd11 = *-@probnodes
; .byte @bd
@psb = *-@probnodes
; .byte @bd
@psd = *-@probnodes
; .byte 0; ;...means no instrument change.
; .byte @sd
@pabA1 = *-@probnodes
; .byte @n_c2
; .byte @sd
@pabAAB = *-@probnodes
; .byte @syn
@pabAB = *-@probnodes
; .byte @abA
@pabB1 = *-@probnodes
; .byte @n_e
; .byte @cpb
@pabnt = *-@probnodes
; .byte @abAx
; .byte @abB
@synx = *-@probnodes
; .byte @syn
@pnote = *-@probnodes
; .byte @abA
; .byte @n_dar
@pnote2 = *-@probnodes
; .byte @n_g
; .byte @n_e
@synn = *-@probnodes
; .byte @n_c2
; .byte @n_gd2
; .byte @syn
; .byte @syn2
;=====
; "INSTRUMENT EDITOR"
;
; Almost looks like real wavetables, although being a "text file instrument
; editor", doesn't it? ;)
;
; Some control contents.

TICK = %0 ;An empty control byte simply causes a delay of one frame.
SID_SR = %10000000
SID_AD = %01000000
SID_CTRL = %00100000
SID_PULHI = %00010000
SID_PULLO = %00001000
SID_FRQHI = %00000100 ;Note that FreqLO is not used at all
STAY = %00000001 ;STAY overrides JUMP, so only use one at a time. Else data will get out of
sync.
JUMP = %00000010

@instdata = *-2: ;Read comment for hardrestart below to get the idea with the "-2" here.
;Also, position 0 in the table is used as a "NO INSTRMENT CHANGE FLAG"
;and shall not contain data anyway.

;--- HARDRESTART
@hardrestart: ;This one should be placed the same ammounts of bytes into the instdata table as ticks to precede
instrument trig.
; .byte SID_SR | SID_CTRL |

```





```

;#print (*-@binbegin)+2+2

;=====
; Here is a list of calculated notefrequencies, assuming that base notes
; of the scale start at $0400 (C), $0800 (C2), $1000 (C3..), $2000, $4000...
; Some of them can be approximated by setting the hi byte of freq to
; the value, and just having lo byte freq set to $00 (as it is when the
; C64 is reset.) These values are calculated with the html/javascript
; file which is attached in the package.

;.byte 4 ; C error: 0 <-----
;.byte 4 ; C# error: 60
;.byte 4 ; D error: 125
;.byte 4 ; D# error: 193
;.byte 5 ; E error: 10 <-----
;.byte 5 ; F error: 86
;.byte 5 ; F# error: 168
;.byte 6 ; G error: -2 <-----
;.byte 6 ; G# error: 89
;.byte 6 ; A error: 186
;.byte 7 ; A# error: 32
;.byte 7 ; H error: 141
;.byte 8 ; C error: 0 <-----
;.byte 8 ; C# error: 121
;.byte 9 ; D error: -6 <-----
;.byte 9 ; D# error: 131
;.byte 10 ; E error: 20
;.byte 10 ; F error: 173
;.byte 11 ; F# error: 80
;.byte 12 ; G error: -4 <-----
;.byte 12 ; G# error: 178
;.byte 13 ; A error: 116
;.byte 14 ; A# error: 65
;.byte 15 ; H error: 26
;.byte 16 ; C error: 0 <-----
;.byte 16 ; C# error: 243
;.byte 17 ; D error: 245
;.byte 19 ; D# error: 6 <-- (Useable, but not used here..)
;.byte 20 ; E error: 40
;.byte 21 ; F error: 91
;.byte 22 ; F# error: 160
;.byte 24 ; G error: -7 <-----
;.byte 25 ; G# error: 101
;.byte 26 ; A error: 232
;.byte 28 ; A# error: 130
;.byte 30 ; H error: 52
;.byte 32 ; C2 error: 0

```

## Resolution

This is a 512 bytes entries by Eric Odland. Here I present a reverse engineering source of the engine, but before, we analyze some features of the player:

- It starts with basic sys call
- Irq is set by adding the pointer to \$314
- It initializes all the sid registers by a loop using a table of values
- The init phase (irq, sid and track/pattern setting) is repeated when the track is over: a way to save code by reuse all code even if not all is necessary
- Tune is composed by 13 patterns to play (*patTable* that use relative offset, so one byte for a pointer)
- The notes used by the tune are 24 and are coded with two tables for low/high values of the notes frequencies
- For each note the duration is \$0A, unless the high bit is 1 (\$15 is so used for duration)
- All the 3 voices play the same notes, but voice 3 is used a little different in release phase from 1 and 2. Also, an effect of adding 4 notes over the current is also used. This is what make the instrument timbre you listen.

Now the source:

```
.org $0801
.byte $0B, $08, $D6, $07, $9E, $32, $30, $36, $31
.byte $00, $00, $00 ; basic call: 2006 sys 2061

.org $080D
Init:
    ldy # $18
loop:
    lda sidTable,y
    sta $D400,y ; Voice 1: Frequency control (1o byte)
    dey
    bpl loop

    jsr initPointer
    lda # $15
    sta $F7
    sta $F8 ; store note duration
            ; set irq
    sei
    lda # <irq
    sta $0314 ; Vector: Hardware Interrupt (IRQ)
    lda # >irq
    sta $0315 ; Vector: Hardware Interrupt (IRQ)
    cli
    rts

irq:
    inc $0420
    dec $F9 ; dec actual duration
    bne setFiltCutHi

    lda # $15
    ldx $FE
    cpx # $02
    beq skipReadDur
    lda $F8 ; read note duration

skipReadDur:
    sta $F9 ; store actual duration
    jsr player
    beq setFiltCutHi

    jsr Init
    jmp $EA31 ; Default hardware interrupt (IRQ)

;=====
; set filter cut off freq according
; to voice 3 envelope output
;=====
setFiltCutHi:
    lda $D41C ; Generator output
    lsr
    sta $D416 ; Filter cut frequency: hi byte
```

```

    jmp $EA31 ; Default hardware interrupt (IRQ)

player:
    ldy $FE
    bne testRelease
    jmp readNext

testRelease:
    dey
    bne skipOnly3

    lda #$40 ; release for voice 3
    sta $D412 ; Voice 3: Control registers
    jmp readNext

skipOnly3:
    ldy $FB
    lda $FA ; note index for effect
    ror
    bcc makeRelease
    tya
    adc #$04 ; uses 4 notes added
    tay

makeRelease:
    lda #$20
    sta $D404 ; Voice 1: Control registers
    lda #$40
    sta $D40B ; Voice 2: Control registers
    sta $D412 ; Voice 3: Control registers
    inc $FA ; inc note index for effect
    jmp outFreq ; y=note index

readNext:
    ldy $FC ; load pattern index
    inc $FC ; increment pattern index
    lda patt,y ; read value
    bne processValue
    jsr getNextPattern
    cmp #$00
    beq readNext
    rts

processValue:
    tay
    asl ; test high bit
    bcc no8Set ; carry was set

    clc
    lsr
    tay ; store without high bit set
    ldx #$0A
    stx $F8 ; store note duration
    inc $FE

no8Set:
    asl
    bcc no4Set ; carry was set

    clc
    lsr
    lsr
    tay ; store without bit set
    ldx #$15
    stx $F8 ; store note duration

no4Set:
    asl
    bcc outFreq
    clc
    lsr
    lsr
    tay
    lda #$15
    sta $F7

outFreq:
    lda freqLow-1,y
    sta $D400 ; Voice 1: Frequency control (lo byte)
    sta $D407 ; Voice 2: Frequency control (lo byte)
    adc #$08
    sta $D40E ; Voice 3: Frequency control (lo byte)
    lda freqHigh-1,y
    sta $D401 ; Voice 1: Frequency control (hi byte)
    sta $D408 ; Voice 2: Frequency control (hi byte)
    sta $D40F ; Voice 3: Frequency control (hi byte)
    lda #$21
    sta $D404 ; Voice 1: Control registers
    lda #$41
    sta $D40B ; Voice 2: Control registers

```

```

    sta $D412          ; Voice 3: Control registers
    dec $FE
    bpl notNeg
    lda #$02
    sta $FE
notNeg:
    ldy $FC           ; load pattern index
    lda patt,y       ; load pattern value
    bne isRead
    jsr getNextPattern
    cmp #$00         ; value read?
    beq isRead
    lda #$01
    rts

isRead:
    lda #$00
    rts

;=====
; Get ext pattern and read the
; value of pattern
;=====
getNextPattern:
    inc $FD          ; pattern index
    ldy $FD
    lda patTable,y
    bne readValue
    lda #$01         ; end: value not read
    rts

readValue:
    sta $FC          ; store patten index
    tay
    lda patt,y
    sta $FB
    inc $FC          ; increment pattern index
    lda #$00         ; value read
    rts

;=====
; Init pointers to pattern
;=====
initPointer:
    lda #$15
    sta $F9          ; store actual duration
    ldy #$FF
    sty $FD         ; reset pattern index
    iny
    sty $FA         ; clear note index for effect
    ldy #$02
    sty $FE
    jsr getNextPattern
    rts

sidTable:
    .byte $30, $04   ; low/high freq voice 1
    .byte $00, $08   ; low/high pulse voice 1
    .byte $20       ; control voice 1
    .byte $05, $9F   ; attack/decay sustain/release voice 1

    .byte $8F, $0A   ; low/high freq voice 2
    .byte $00, $04   ; low/high pulse voice 2
    .byte $40       ; control voice 2
    .byte $08, $A8   ; attack/decay sustain/release voice 2

    .byte $FF, $0F   ; low/high freq voice 3
    .byte $00, $05   ; low/high pulse voice 3
    .byte $40       ; control voice 3
    .byte $0A, $B9   ; attack/decay sustain/release voice 3

    .byte $00, $50   ; low/high of filter
    .byte $67, $1F   ; resonance/filter mode/volume

patt:
    .byte $FF

pat01:
    .byte $11, $07, $0B, $06, $0A, $04, $09, $05
    .byte $0A, $07, $0B, $06, $0B, $06, $07, $09
    .byte $0A, $00

pat13:
    .byte $0F, $0B, $07, $07, $04, $04, $05, $07
    .byte $09, $08, $06, $03, $04, $04, $05, $07
    .byte $09, $00

pat25:
    .byte $13, $05, $07, $09, $0A, $05, $07, $0A

```

```

.byte $0B, $00

pat2F:
.byte $12, $06, $08, $0B, $0C, $08, $0D, $8C
.byte $0B, $4A, $00

pat3A:
.byte $11, $09, $0A, $0A, $0B, $0B, $0C, $0A
.byte $09, $00

pat44:
.byte $10, $09, $08, $08, $07, $05, $04, $02
.byte $01, $00, $13, $07, $0B, $06, $0A, $04
.byte $09, $05, $0A, $00

pat58:
.byte $13, $8E, $0D, $0B, $84, $07, $89, $0A
.byte $89, $08, $86, $04, $87, $0A, $88, $4B
.byte $00

pat69:
.byte $13, $8E, $0D, $4B, $8B, $09, $48, $06
.byte $05, $04, $04, $00

pat75:
.byte $13, $13, $13, $00

patTable:
.byte pat01-patt, pat01-patt
.byte pat13-patt, pat13-patt
.byte pat25-patt, pat2F-patt
.byte pat3A-patt, pat44-patt
.byte pat4E-patt, pat4E-patt
.byte pat58-patt, pat69-patt
.byte pat75-patt, $00

freqLow:
.byte $E9, $61, $68, $8F, $30, $8F, $18, $D2
.byte $C3, $D1, $1F, $60, $1E, $31, $5A, $A3
.byte $CC, $23, $86, $B4, $47, $98, $47, $0C

freqHigh:
.byte $07, $08, $09, $0A, $0B, $0C, $0E, $0F
.byte $10, $12, $15, $16, $19, $1C, $02, $02
.byte $02, $03, $03, $04, $05, $05, $06, $07

.byte $00

```

## Conclusion

This time we had looked at one 256 bytes and one 512 bytes engines and they are very different. In particular the 512 bytes take emphasis to the notes to play and use the 3 voices together for creating a timbre of one instrument. The 256 bytes instead uses random sequences to make the composition more longer, but all 3 voices play different notes/instruments.

# SID Factory vs Ninjatracker

by Stefano Tognon <[ice00@libero.it](mailto:ice00@libero.it)>

It is strange to see that two new editors were released about the same day.

The first is the Laxity tracker: SID Factory

The second is Cadaver editor: Ninjatracker V2

In this article I go to show you the two new editors and soft compare the features they supply.

## SID Factory

The first thing that pop up by looking at the tracker is the very similarity with JCH editor, that can be summed as:

- One editor that can use many different player drivers
- Tracker style editor based onto tables of values
- Very similar looking (even if you will note that they are different)

However the program starts with the main frame you see here in the right side: it has the main menu (enter editor, disk, packer and options) in the upper part and in the lower part it has information about the current loaded driver.

The JCH editor has the similar main frame, but in it there is even the list of the disk contents. In this editor you will see the list of files in the apposite sub-menu, but on the other side you have more information about the current used driver (in JCH there is only the last screen line for that information).

Maybe (but this is just my opinion) it could be interesting to have one frame as in JCH, but with all information that SID Factory has, in the main menu. That could be possible by restyling the interface a little bit.

```
SID Factory v0.05d

F1 Enter editor
F3 Disk menu
F5 Packer menu
F7 Options menu

Status: Work tune

Driver name: Driver 5.00d
Driver size: $002D
Sequences available: 7D
Maximum multi speed: 04

DMEM free: $0807 bytes.
```

```
00, 0K, 00, 00

F1 Read dir
F3 Save to file
F5 Disk status
F8 Main menu

Files in dir: 08

>SID FACTORY 0.5D
SID FACTOOL 0.1D
DRIVER 5.01D
DRIVER 5.02D
D5.00/25-18
D5.00/25-83
D5.00/25-86
D5.00/25-89

Blocks free: $014E
```

Entering the editor itself, you will see that it looks similar to JCH here too: in the upper part there is the patterns editor for each tracks and in the below part there are the tables for instruments.

Color is green/white like in JCH but here it uses even light green and blue, giving to the editor his different look style from JCH.

Take present that for each track, you will set the pattern to use by entering it as a number, and below you have the pattern to enter. So, all the tunes is viewed by go down in the pattern editor.

In one pattern row you can insert 3 values:

- Instrument number (32 max)
- Command number (64 max)
- Note values

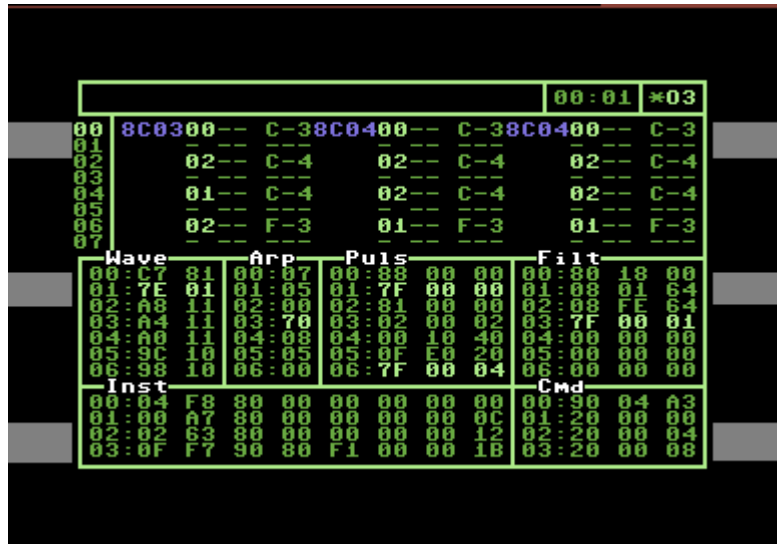
Instrument and command number refers to the fixed table of values that are in the low part of the editor.

A dynamic table (in center of editor) is activated on demand and contains other values used by the other tables.

Actually SIDFactory comes with two different driver:

- V5.0x: standard music driver with calculated vibrato and no aim at execution speed (multi-speed allowed)
- V6.0x: "fast" music driver without calculated vibrato and aims at a maximum execution spike at around \$18 scanlines

Here I'll go to report the actual meanings of values for the two drivers and so you can figure how it works:



<b>Driver v5.0x</b>		<b>Driver v6.0x</b>	
<i>Instrument: aa bb cc dd ee ff gg hh</i>		<i>Instrument: aa bb cc dd ee ff</i>	
aa	Attack/Decay	aa	Attack/Decay
bb	Sustain/Release	bb	Sustain/Release
cc	Instrument properties 1: <ul style="list-style-type: none"> <li>• Restart settings (bit 7-6)</li> <li>• Arpeggio mode enable (bit 5)</li> <li>• Oscillator reset (bit 4)</li> <li>• Hard restart table pointer (bit 3-0)</li> </ul>	cc	Instrument properties: <ul style="list-style-type: none"> <li>• Restart settings (bit 7 / \$80)</li> <li>• Filter pointer set enable (bit 6 / \$40)</li> <li>• Pulse pointer set disable (bit 5 / \$20)</li> <li>• Oscillator reset (bit 4 / \$10) (Changed from \$08 in driver 6.02 and forward)</li> </ul>
dd	Instrument properties 2 <ul style="list-style-type: none"> <li>• Pulse and filter settings (bits 7-6)</li> <li>• Arpeggio delta delay (bits 0-5).</li> </ul>	dd	Filter table pointer (Set if bit 6 of Instrument properties is set)
ee	Resonance setting (If this one is non-zero, filter will enable for the channel the instrument is set on)	ee	Pulse table pointer (NOT set if bit 5 of Instrument properties is set)
ff	Filter table pointer	ff	Wave table pointer
gg	Pulse table pointer		
hh	Wave table pointer		

<b>Driver 5.0x</b>		<b>Driver 6.0x</b>	
<i>Wave table: aa bb</i>		<i>Wave table: aa bb</i>	
aa	Note offset (If bit 7 is true, the note is fixed) if aa=\$7x it's a wave table command <ul style="list-style-type: none"> <li>• 7f = Jump to position bb</li> <li>• 7e = Set wave table delay to bb</li> <li>• 7d = Wait bb ticks (not implemented yet)</li> </ul>	aa	Note offset (If bit 7 is true, the note is fixed) if aa=\$7f, Jump to \$bb
bb	Waveform	bb	Waveform

<b>Driver v5.0x</b>	
<i>Arpeggio (not used in driver v6.0x)</i>	
aa	Note offset if aa=\$7x "repeat X steps from the top of the current arpeggio"-command.

<b>Driver 5.0x</b>		<b>Driver 6.0x</b>	
<i>Pulse table: aa bb cc</i>		<i>Pulse table: aa bb</i>	
if aa < \$10 (Add to pulse) <ul style="list-style-type: none"> <li>• aa = add pulse high</li> <li>• bb = add pulse low</li> <li>• cc = execution time (in frames/updates)</li> </ul>	if aa < \$80 <ul style="list-style-type: none"> <li>• aa = Execution time of current pulse program step</li> <li>• bb = Pulse sweep value (reversed nibble).</li> </ul> Pulse sweep is store in reversed nibble.	if aa = \$7f (Jump to index) <ul style="list-style-type: none"> <li>• aa = \$7f</li> <li>• bb = unused (could maybe be used for jump count?)</li> <li>• cc = new index</li> </ul>	if aa >= \$80 (Set pulse value) <ul style="list-style-type: none"> <li>• bb = New pulse value (if bb = xy, then new pulse width is: \$0yx0)</li> </ul>
if aa >= \$80 & aa < \$90 <ul style="list-style-type: none"> <li>• aa = \$8x where x is the new high pulse value</li> <li>• bb = new low pulse value</li> <li>• cc = execution time (in frames/update - effectively wait time)</li> </ul>	if aa = \$7f (Jump to index) <ul style="list-style-type: none"> <li>• aa = \$7f</li> <li>• bb = new index</li> </ul>		



<b>Driver 5.0x</b>	<b>Driver 6.0x</b>
<i>Filter table</i>	<i>Filter table: aa bb</i>
if aa < \$10 (Add to pulse) <ul style="list-style-type: none"> <li>• 0a = add filter low value (Beware, these are the opposite of the pulse table, where this one is the high part!)</li> <li>• bb = add filter high</li> <li>• cc = execution time (in frames/updates)</li> </ul>	if aa < \$80, <ul style="list-style-type: none"> <li>• aa = time to execute filter program step</li> <li>• bb = add to current filter value</li> </ul>
if aa = \$7f (Jump to index) <ul style="list-style-type: none"> <li>• aa = \$7f</li> <li>• bb = unused (could maybe be used for jump count?)</li> <li>• cc = new index</li> </ul>	if aa = \$80 (Set pulse value) <ul style="list-style-type: none"> <li>• aa = \$80</li> <li>• bb = new pulse value</li> </ul>
if aa >= \$80 & aa < \$90 <ul style="list-style-type: none"> <li>• aa = \$8x where x is insignificant</li> <li>• bb = new high filter value</li> <li>• cc = execution time (in frames/update - effectively wait time)</li> </ul>	if aa = \$7f (Jump to index) <ul style="list-style-type: none"> <li>• aa = \$7f</li> <li>• bb = new index</li> </ul>

<b>Driver 5.0x</b>		<b>Driver 6.0x</b>	
<i>Commands</i>		<i>Commands</i>	
<b>0x aa ?b</b> Set slide	aabb is the added value to the current frequency value (only additive)	<b>0X XX</b> Set slide up	XXX is the value added to the current frequency value
<b>1x aa bb</b> Set vibrato	aa = Frequency ?b = Amplitude	<b>1X XX</b> Set slide down	XXX is the value subtracted from the current frequency value
<b>2x ?? aa</b> Set arpeggio pointer	aa = Arpeggio index	<b>2X YY</b> Set vibrato	X = Frequency YY = Amplitude, absolute value added/subtracted to/from frequency value
<b>3x aa bb</b> Set filter and/or pulse table pointer	x - Bit 0 => AA is a filter table pointer Bit 1 => BB is a pulse table pointer Ex. x=0 =>AA and BB remain undefined x=1 =>AA is set as new filter table pointer x=2 =>BB is set as new pulse table pointer x=3 =>AA is set as new filter table pointer AND BB is set as new pulse table pointer	<b>3X YY</b> Set Filter parameters	X = Band width YY = Resonance and filter select bits

<b>Driver 5.0x</b>		<b>Driver 6.0x</b>	
<b>4x aa bb</b> Set wave and/or pulse table pointer	x - Bit 0 =>AA is a wave table pointer - Bit 1 =>BB is a pulse table pointer Ex. x=0 =>AA and BB remain undefined x=1 =>AA is set as new wave table pointer x=2 =>BB is set as new pulse table pointer x=3 =>AA is set as new wave table pointer AND BB is set as new pulse table pointer	<b>4x YY</b> Set Filter program pointer	YY = new filter program pointer
<b>8x AD SR</b>	ADSR is set for the scope of the current instrument, including the current note event	<b>5x YY</b> Set pulse program pointer	YY = new pulse program pointer
<b>9x AD SR</b>	(Direct)ADSR is set for the current note event only, and reset to the previous ADSR value on next event in the instrument scope	<b>6x YY</b> Set wave table pointer	YY = new wave table pointer
<b>c0 xx ??</b>	Set Volume	<b>8D SR</b> Set (A)DSR	Attack is always set to 0.. Bummer!
<b>f0 xx ??</b>	Set new tempo. xx = pointer into tempo table		
<b>f1 xx xx</b>	Set portamento. (Set until vibrato, slide or new instrument is set!) xx xx = Frequency variable add/subtract		

<b>Driver 6.0x</b>	
<b>Init table</b>	
<b>aa bb cc</b>	aa = Tempo table index ==> first comes filtersetting, then volume. bb = Volume / Filter bandpass setting (i.e. \$1f = \$f volume \$1 bandpass!) cc = Filter Resonance / Filter channel enabled (i.e. \$f1 = \$f resonance, \$01 = enable filter channel) Note that filter enable settings are bit: bit 0 / \$01 = channel 1 bit 1 / \$02 = channel 2 bit 2 / \$04 = channel 3

# Ninjatracker

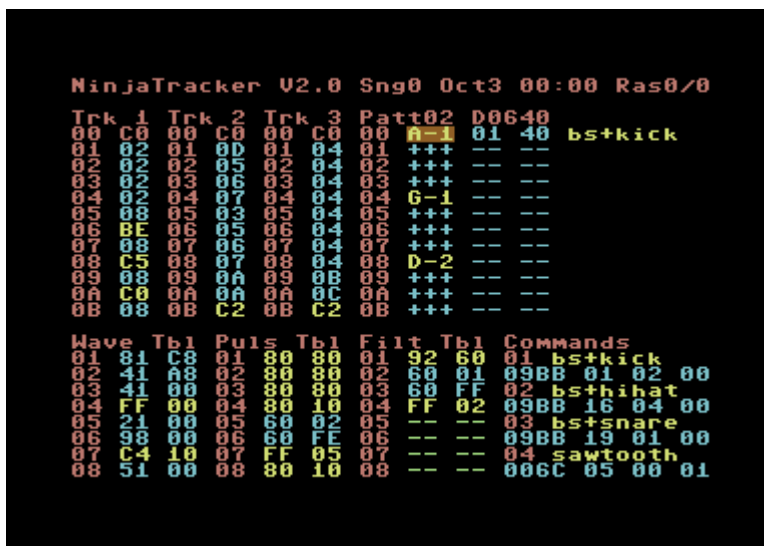
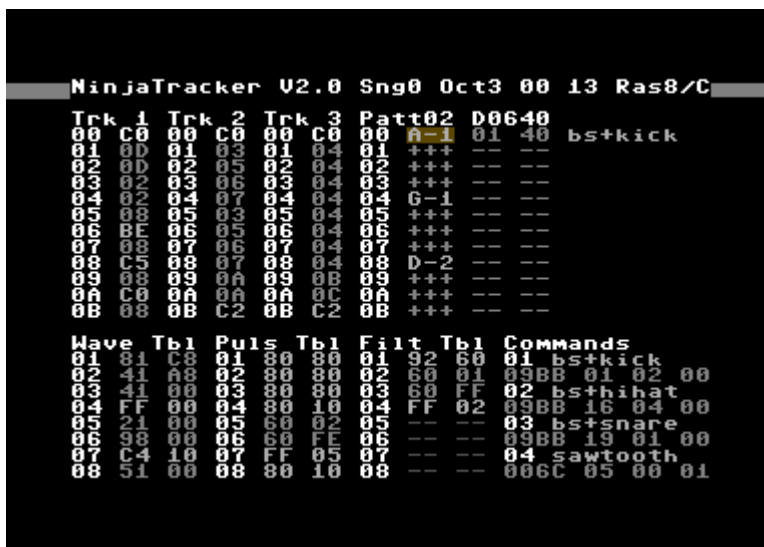
Ninjatracker V2.x is based onto V1.x that Cadaver created years ago for having an editor that had low rastertime and memory usage that can be used for the music of his games.

The editor was quite hard to use, but that were an utility that convert Goatracker tune to Ninjatracker format, so one could be concentrate only in instruments fixing upon the tune was imported into Ninjatracker.

The editor starts with his editor page that have all you need to compose into it.

The used colors in the editor are in gray scale, maybe a combination that look similar to the Goatracker editor, and so familiar for the ones that use GT. However, colors are configurable by user that can change them and so make a totally colored editor (look at the image in right part for an example).

The editor allow you to create up to 16 sub-tunes, and each sub-tune have 3 tracks (up left part of the editor) where you enter the pattern number to play. Note that you have reserved pattern numbers for transpose up of down one patten:



Track	Description
00	Loop
01-7F	Pattern 01-7F
80-BF	Transpose downwards
C0-FF	Transpose upwards

In the upper right part there is the pattern editor where you can insert note/duration/commands of the pattern. His syntax is:

Note / key on (+++) / key off (---)	Command number (01-7F) Legato (81-FF)	Note duration	Command name (just for best looking)
-------------------------------------	--	---------------	--------------------------------------

You can so indicate the note to play (C1-B7) and, in a notation that look likes tracker, if key stays on (+++) or off (---). Then there is the space for the command to use (a command is an instrument setting if one note is specified or effect to apply in the other cases) and the duration of the note. If no one is inserted, it uses the previous values. In the last part it is reported the name associated with the command. This is probably one of the best features as it is more simple to look at what you have done in the pattern.

Note that you could use the editor in a manner very similar to a tracker if you set a fixed notes duration and make use of key on/off.

Each command has this syntax:

ADSR	WAVE table	PULSE table	FILTER table
------	------------	-------------	--------------

You can so specify the Attack/Decay/Sustain/Release of instrument (not used if command is in legato mode), and the pointer (00 means use actual running tables) to use for wave, pulse or filter table commands.

In each table, there is the command and a parameter for the command:

<b>Wave table</b>	<b>Description</b>
00-8F	Set waveform, right side is arpeggio (00-7F relative, 8C-DF absolute notes)
90-BF	No waveform, delay arpeggio by 00-2F frames
C0-DF	Vibrato with speed 00-1F, right side is depth
E0-FE	Slide with speed highbyte 00-1E, right side is speed lowbyte
FF	Jump, right side is destination, not to be entered directly from a command
<b>Pulse table</b>	<b>Description</b>
01-7F	Modulate pulse for 01-7F frames, right side is signed mod.speed
80-FE	Set pulse to right side value
FF	Jump, right side is destination, can be entered from a command
<b>Filter table</b>	<b>Description</b>
01-7F	Modulate cutoff for 01-7F frames, right side is signed mod.speed
80-FE	Set passband (left nybble-8), channels to be filtered (right nybble) and cutoff (right side)
FF	Jump, right side is destination, can be entered from a command

Note that Cadaver released many versions of v2.x (you can see the history in the news section). However they differ in some points like hardrestart type, notes duration and other minus effects. This means that you have to choose the editor you need to use.

Maybe the best solution should be to have runtime loadable players in one common editor, like in SIDFactory or JCH.

The other things to say about the editor is that you have a function that pack and optimize the pat-

tern of the song when it is to release and a relocater that can even save the song without the player for use as game effects in a game.

The editor use very low rastertime as in version v1.x even if now it has more potential. What is missing is that you cannot use it for creating multispeed tune inside the editor.

## Comparison

Below there is a table that sum some of the features of the two editors, just to try a comparison.

You will see a + or – signal for the features that I like and not like (this is about my taste, don't take it too seriously. In fact, many features here are not related to quality of music you can create, but are simple improvements or nice features).

Else the first features (tracker vs duration based editor) is maybe something that let you choose one editor instead the other at the beginning without look at the other features. I myself prefer the tracker ones editor as it is more evident the flow of what happens in the tune.

<i>SID Factory</i>		<i>Ninjatracker</i>	
Tracker based editor	+	Duration based editor (but can be used even in a manner similar to tracker)	+
Loadable drivers for having different music features. A change to the editor itself is available to all the player	+	Different programs for having different music features. For a programming point of view this means to have many versions of the program to maintenance if you change something into the editor	-
Inside the editor, all (patterns, instruments, tables) are in one screen	+	Inside the editor, all are in one screen. This is always a point that for me help composing: switching to other screens could cause to lose temporary the vision of what you are being done	+
Files to load are selectable by a list. This is the best way, as you don't have to type the name of the file	+	The name of the files to load are to insert by hand. With some other editors that add an "extension" this is even more complicated	-
When loading a file does not show anything until the file is loaded	-	When loading show in the border an animated effect (this show you that the operation is going on)	+
Source code is not available. Maybe future additional thirty part improvements to the editor could not be realized in full manner	-	Source code is available. This means that in future thirty part programmer could improve/modify the editor.	+

<i>SID Factory</i>		<i>Ninjatracker</i>	
No online help available inside the editor. If you don't remember a key function you have to read the offline documentation. With today emulator onto pc, maybe offline documentation is just at one click distance, but if you compose onto the C64 it is not so simple	⊖	Online help inside the editor full of information. Maybe it is always more simple to look at it that going to search in offline documentation.	⊕
Offline documentation about the editor is available.	⊕	Offline documentation about the editor is available.	⊕
Only one kind of insertion mode for notes	⊖	ProTrack or DMC notes insertion mode (for sure this help if you come from other editors)	⊕
Instruments are based onto tables. Even if for beginners composers the use of table is hard, only with them you can get the best varieties of sound from the sid chip	⊕	Instruments are based onto tables of values.	⊕
There is a function that pack the tune	⊕	There is a function for optimize patterns	⊕
I not find a way to relocate the code to a given address (maybe a features not present into this alfa version)	⊖	Relocator is available (with Gamemusic option, so you can even create sound effects to use inside a game)	⊕
Example tunes are available. Examples are very important for looking at the power of the editor and how it works when you try it the first time	⊕	Example tunes are available	⊕
Multispeed in a natural way with a specific player. I can say that this is the first editor I try that has this features (in other, if you add multispeed to a tune, you have to adjust instruments to the new mode)	⊕	Not multispeed tunes can be created	⊖
No subtunes availables (this actually is a point that reduce the possibility to use it for creating music for game/demo unless you need only one tune to play)	⊖	Max 16 subtunes	⊕
Max 32 instruments (It is a good number) and max 64 commands	⊕	Max 127 commands (a command inside a note is an instrument)	⊕

<i>SID Factory</i>		<i>Ninjatracker</i>	
Fixed colors (even if the color looks good, maybe someone could be happy to customize them)	⊖	Customization of colors is allowed. If someone don't like the gray scale color used, he can change them	⊕
No fast forward option found	⊖	Fast forward when playing	⊕
Only raster time and time passed is showed when playing the tune (no note being play is showed)	⊖	Only rastertime usage is showed (no note being played is showed)	⊖
Instruments cannot be saved for use in other tunes. This means that you have to create the instruments form scratch each time you compose.	⊖	Instruments cannot be saved for use in other tunes	⊖

## Conclusion

Summing my comparison table, I have that Ninjatraker has 15+ and 4-, while SIDFactory has 10+ and 9-. That simple means that all the two editors have space for a little improvements and maybe SIDFactory is a little behind in my comparison by the fact that it is in alpha state and so no all the simple features are inserted. Think for example at the online help that it is a powerful tool for using the editor until you have learn all his keys/features.

In particular only 2 features are missing from the two editors the same time: the ability to show in real time what it is played for each voices (for a tracker that could be more natural to implement respect to a duration based editor), and the ability to load/save the instrument you have created.

However I hope that this little battle for the two editors give you the idea to test them and produce some quality sid music and maybe give some suggestion for the editors authors to improve them.





*QED in 11 end*