Ice Din 12

“Katja Woywood”

Vice snapshot with Vice palette

Made with the GIMP from a KW photo
and converted to C64 320x200
Hires Mode Bitmap
by Stefano Tognon
in 2007

“Late is late”
...

Team Ice **Free Software Group**

GoDin 12
version 1.00
24 May 2008

SIDin Contents

General Index

Editorials

Stefano Tognon <ice00@libero.it>

Hi, again.

In this number of SIDin there are the final two analysis of the entry to Tiny Sid Compo II.

The second article is about ripping a tune. This time I not disassembly and reassembly it as I done in the past, but simple use the common way to isolate the music part and add the proper irq routine. As usual I describe all the steps involved, so you can learn a complete way for ripping and so you can try yourself with other tunes.

This is a very delayed issue and a very soft one. Sorry for that, but I have too many activity running. Lot of this activities are Sid related, for example I'm developing XSidplay2 at Sourceforge: this is the second version of the historical Sid Linux player that was no more developed for some years. Now it's time to make it growing (it already recognizes the player used by the song, supports SDL sound and is available for Windows system too).

HVMEC (High Voltage Music Engine Collection) is still active and with some contributions in the last year it growing very well- Lot of new material to release is being processed.

Finally I'm developing a Java Tracker for composing music for the Sid. For sure it will be very rastertime consuming, but I want to have the possibility to control all the sound in details (for example an instrument definition can reach up to 2KB of data). More about this will be available in the next number.

Bye
S.T.

News

Some various news of players, programs, and competitions:

- SID chip distortion simulation efforts
- Polly Tracker v1.2+
- X-SID v1.00
- Goattracker stereo 2.59
- Goattracker 2.6-2.67
- CRD-Tracker64 Alpha
- DMC Music Compo 2007
- HVSC update 47
- C64.sk SID Compo 7
- Java SID Player Music Library V2
- XSidplay2
- Rockbox
- C64.sk Cover Compo
- HVSC update 48
- HERMIT 3SID-TRACKER 2008
- HARDSID 4U

SID chip distortion simulation efforts

In February 2007 Antii Lankila released a patch against ReSID engine in libsidplay2-2.1.1 to make its sound closer to 6581R4 chip. Here it is reported lot of information about the project taken from the main page:

Patch status:

- **Applying the patch will break 8580 simulation** because it modifies it like it was a 6581. Solution: I need to extend resid internals + sidplay2 to bring all distortion tunables into some kind of configuration file. 8580 can just have settings that make no distortion, then.
- **The distortion term is not quite right**. Brutal distortion is well emulated on some songs (Mechanicus) but badly with some others (Filter). Lead sounds, as rule, seems to come out alright (Cybernoid and Spijkerhoek). On the other hand, something is wrong with Land of Illusions, the initial bass is maybe too distorted. Vendetta, Miami Vice and Gloria do not distort enough. *I predict the largest difficulties are with combined filters.*
- **The delicate filter interplay in David Dunn's songs is only partially emulated**. This is a difficult case. I believe that I need to allow the distortion terms to permute the filter state in order to emulate the effects and instruments on Dunn's songs.
- **CPU usage does not increase much**: the current situation is tolerable. The distortion now eats some 20% of CPU or so of an AMD64 3200+. Because resid synthesises audio at 1 MHz frequency, only very cheap algorithms ought to be used. We'll see if the system can be optimised as it nears completion.

Known nonlinearities in the SID chip:

- **Filter distorts high volume sounds a LOT.** I believe that this effect is caused by abrupt nonlinear behaviour by the SID chip op-amps. I have modelled this by approximating the voltage difference between the input and output of two main sid-chip op-amps, and distorting the lowpass, bandpass and highpass outputs if the difference goes above a certain threshold.
- **Filter resonance frequency shifts up if the intensity of sound grows, up to roughly 1 full octave, or 2x the frequency.** This is responsible for the sound of the initial "drums" in Jeff's Hard Track. Jeff keeps filter CF value (Center Frequency) as constant for the duration of each hit, but the audible effect is still a filter sweep down as these hits fade. A simi-

lar case is Elite, which simply will never be emulated correctly unless the filter is made to properly respond to these changes in sound intensity.
The linear filtering equation is as follows:

```
Vhp = Vbp / Q - Vlp - Vi
Vlp -= w0 * Vbp
Vbp -= w0 * Vhp
```

I suspect that the filtering may be simulated by correcting the w0 terms with a term derived from the absolute difference of the outputs, for instance the middle equation should be something like:

```
Vlp -= w0 * Vbp * distortion_function(Vlp, Vbp);
```

My current approximation of the distortion function calculates the absolute difference in output, and then scales that (with clipping) to to range 1 .. 3.

Known bugs in sidplay2 removed by this patch:

- **6581 CF-to-frequency mapping is rather wrong for 6581R4**. A replacement that matches closely with my 6581R4 is substituted instead. For instance, ReSID tables claim that CF value of 1024 selects approximately 4.6 kHz resonance frequency, but my measurements indicate it's approximately 1.1 kHz instead! This is a difference of roughly 2 octaves, and needs to be corrected for 6581R4-like sound. This CF-to-frequency mapping is not better than the one in resid in absolute terms (except that it has been sampled with higher accuracy), because there exists considerable variation between chips. The stock ReSID curve + distortion patch seems quite appropriate for Terra Cresta, for instance.
- **Filtered outputs are 4.5 to 6 dB quieter than unfiltered outputs.** This difference can be seen on any FFT plot from the chip when a low-volume sound is played without routing to filter, and then with any of the filters enabled. The output level difference may arise from the nonlinear properties of the filter itself.
- **Filter calculation was done in incorrect order, resulting in excess treble for highpass output.** This bug has been confirmed by Dag Lem and fixed in ReSID upstream.

Check and download the patch from: http://bel.fi/~alankila/c64-sw/

Polly Tracker v1.2+

Aleksi Eeben in March 2007 has released a plus version on Polly Tracker with new 3 disks of demo modules and an alternate Digimax version with 8-bit output:

- 4 sample channels
- 4-9 kHz sample rate on each channel (C-2 = 8000 Hz)
- 8-bit internal mixing
- 4-bit output on stock C-64 or 8-bit output with Digimax version
- Dynamic mixing based on polling the hardware timers, never skips a sample
- 48K reserved for sample data
- Loads 8-bit unsigned raw samples
- Edit options to adjust sample volume, trim sample end and octave upsample
- 6581/8580 ok, NTSC/PAL compatible and IDE64 friendly
- No SID voices used (except voice 3 output as sequencer sync)
- Standalone player, module-to-executable and module-to-SID tools included

Download at: http://noname.c64.org/csdb/release/?id=47686

X-SID v1.00

After a lamer had released some incomplete and not to spread Jeff's editors, Soeren Lund had decided to release in April a more usable version of X-SID, but that it still incomplete.

However, due to the action done by the lamer, no further develop will be done to this editor.

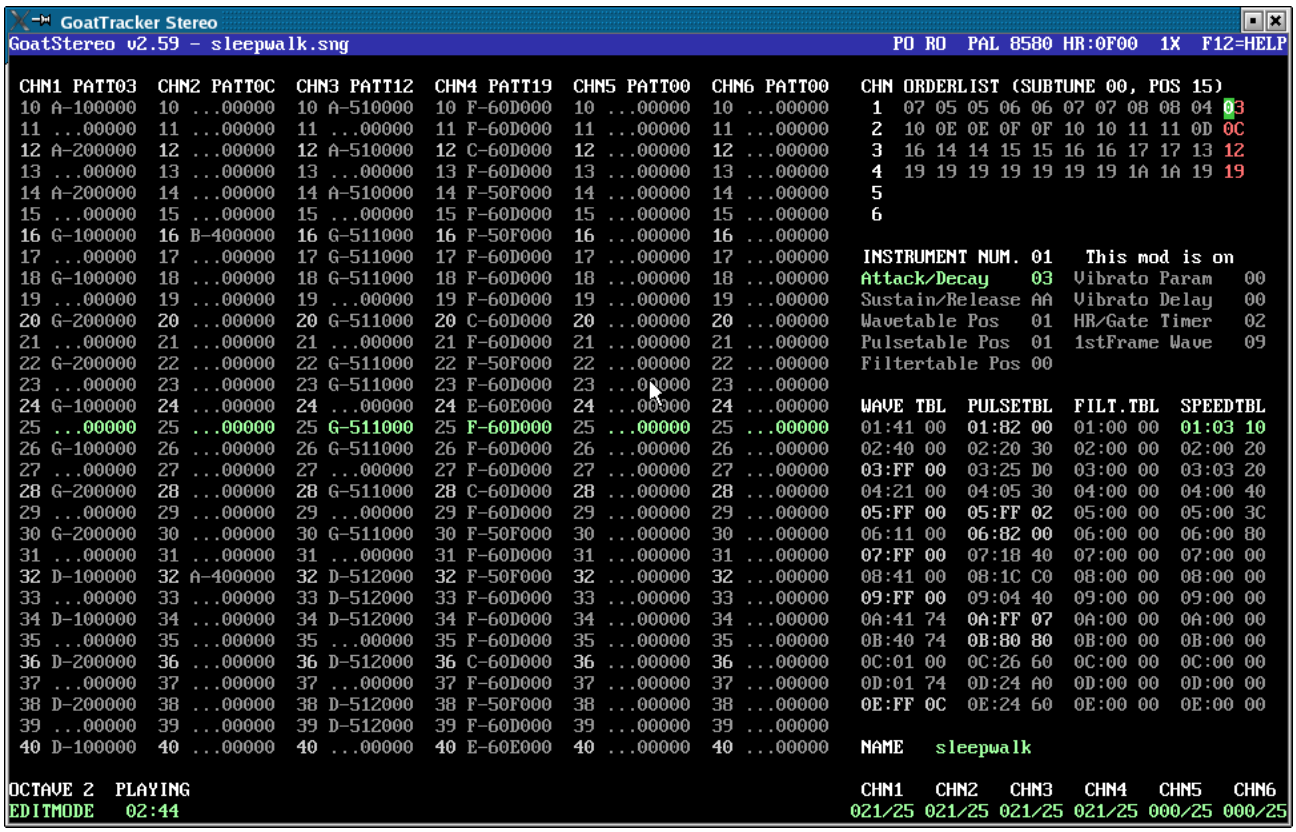
The editor looks very promising and it's very sad it will not improved anymore.

You will find the editor to download here http://noname.c64.org/csdb/release/?id=47985 , and a thirty part relocator here http://noname.c64.org/csdb/release/?id=48017.

Goattracker stereo 2.59

On 12 April Cadaver had released a stereo experimental version of Goattracker 2.59

Download from http://covertbitops.c64.org/tools/gt2stereo.zip



Goattracker 2.6-2.67

Still in April news versions of Goattracker come out (and in May 2008 the last version):

v2.6

- Fixed pattern default length selection display when decrementing from a length of 100 or higher.
- Fixed mouse selection of pattern when adjusting an adjacent channel.
- Fixed help screen instructions.
- Changed resolution to 800x600.
- Changed all songname rows to be displayed at the same time.
- Changed mouse control to allow pattern column selection even when left mouse button is held down.
- Changed speed of PageUp/PageDown scrolling to be faster.
- Optimized graphics output.
- BME library is no longer needed.

v2.61

- Added the backquote key (top-left on keyboard) to select channel in pattern edit mode, and to select table in table edit mode. Use with SHIFT to go backwards.
- Added SHIFT+channel number to mute channels in pattern edit mode.

v2.62

- Added possibility for realtime calculated note independent (hifi) portamento & vibrato. Warning: has potential for huge rastertime increase.

v2.63

- Fixed note independent portamento & vibrato to use the last note set in wavetable for calculations, instead of the last note in patterndata.

v2.64

- Fixed paste in table (SHIFT+V) working also without SHIFT pressed.

v2.65

- Fixed raw keycodes over 511 interpreted as some other keys in the 0-511 range.

v2.66beta

- Initial cycle-exact HardSID support (Win32 only)
- Permit running without sound.

v2.67

- Configurable cycle-exact HardSID buffer length (separate for inter-active and playback mode, see /T and /U command line options)

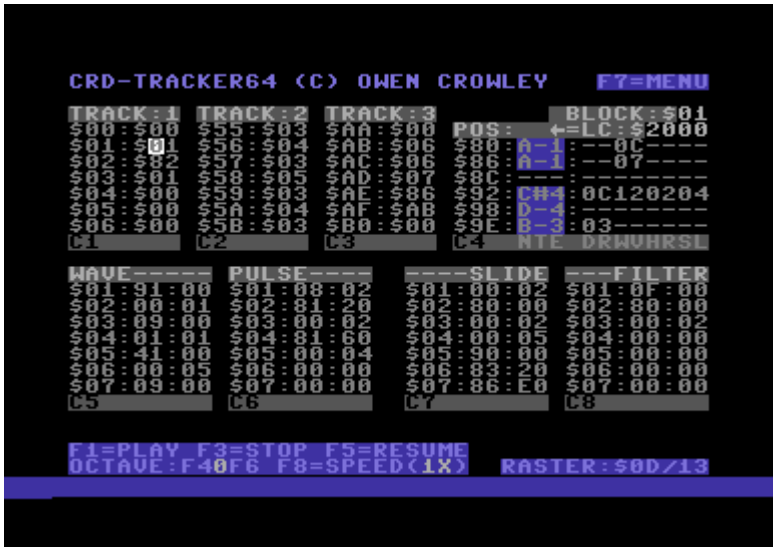
CRD-Tracker64 Alpha

On 9 May 2007, Owen Crowley had released his alpha version of a music tracker: CRD-Tracker64.

You can event read the history of this not finished editor by looking at the creator Readme file that it in the disk.

Download from cdsb:

http://noname.c64.org/csdb/get-internalfile.php/39695/CRDTRACK-ER64_ALPHA.D64



DMC Music Compo 2007

In 2007 Richard lanch a music compo for DMC editors (all the versions were allowed):

http://www.redesign.sk/tnd64/DMCcompo.html

Here the result:

1:

Name of song: Wot Da Funk
Author : Marcin Majdzik (Psycho)
Version : DMC V4 (Modified by Glover/Samar)
Duration : 2:15
SID Type :8580 (New)

2:

Name of song: Extreme - Part 2
Author : Surgeon/Vulture Design
Version : DMC V7.0 (Modified version of DMC V4.0 ;o))
Duration : 2:54
SID Type :8580 (New)

3:

Name of song: Tanker
Author : Rio/Rattenrudel
Version : DMC V7.0 (Modified version of DMC V4.0 IDE 64 Version)
Duration : 2:09
SID Type :8580 (New)

HVSC update 47

In July the new version of HVSC was released at http://www.hvsc.c64.org

After this update, the collection should contain 34,127 SID files! This update features (all approximates):

- 1127 new SIDs
- 19 fixed/better rips
- 8 fixes of PlaySID/Sidplay1 specific SIDs
- 8 repeats/bad rips eliminated
- 773 SID credit fixes
- 700 tunes assigned a sidmodel flag
- 15 UNKNOWN demo tunes identified
- 29 UNKNOWN game tunes identified

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)

A-Man - venturing into the Pollytracker domain now!
Bart
Richard Bayliss
Bernhard Burgstaller (NEW)
CRD - kindly donated his complete collection to us
Chantal Goret (NEW)
Dexter (NEW)
Eco
Fanta - make sure to have a listen to his Desert Dream conversion!
Fox
Gop
Greg
Gregfeel
Harlequin
Heinmuck

Hukka
MAC2
Merman
Nastiness Inc.
Pernet
Raze
Sax
Skam (NEW)
STP Sound System
Topaz
Zeta (NEW)

Image
Mac / Radical
Moogle Charm (NEW)
Omoroca (NEW)
Q-Man
Rio
Sharp
Slayer
Tonid (NEW)
Vintaque

HVSC News

- Motion joined the HVSC Crew
- Steppe retired from the HVSC admin post. Rambones will take over, good luck!
- The new directory structure requires a new update tool. It got updated for Linux and Windows, so we feel the majority of users won't have a problem. For the exotic platform users: The source code of update tool 2.8.4 is on the HVSC website in the Downloads section. If you manage to compile it on your specific platform, feel free to send it over! And by the way: You can still run update #47 with update tool 2.8.3. It will complain heavily that the */Hubbard_Rob/* directory is not where it expects it to be, thus assuming you did something fundamentally wrong. Just ignore the warning, nod away the next "y/n are you sure you're sure?" question and it will work anyway.

C64.sk SID Compo 7

The annual C64.sk compo was performed even this year, with lot of tunes:

Place	Release / Scener	Points
1	Darkening by Cadaver	(1314 PTS)
2	There was a Light That Went Out by Randall	(1167 PTS)
3	Autumn Symphony by Orcan	(1152 PTS)
4	Danger Zone by Conrad	(1150 PTS)
5	His Pearl by _V_	(1087.5 PTS)
6	Unsophisticated by Peace	(1011.5 PTS)
7	Unforgivable by Dane	(999 PTS)
8	Not A Jazz by Jammer	(995 PTS)
9	immoral coil by dalezy	(950 PTS)
10	y Little Daughter by PCH	(930 PTS)
11	6581 Destinations by Stainless Steel	(926 PTS)
12	November Echoes by The Syndrom	(910.5 PTS)
13	The Sexy Hardrestart by Hein	(887.5 PTS)
14	Girls'n Ghosts Ate Kleve by Linus	(863 PTS)
15	The Sweet Odour Of Jesus Christ by Intensity	(863 PTS)
16	Saturday (Laid Back) Weirdness by No-XS	(813.5 PTS)
17	I Died Defending the Mothership by Uneksija	(799 PTS)
18	Volcanoes of Passion by Sidder	(788.5 PTS)
19	Unstoppable by PsycHo	(732 PTS)
20	Mix 'Em Up 64 Style by Richard	(694 PTS)
21	Sorry For The Delay by Peter Bergstrand	(662 PTS)
22	Story About F Sharp by Jakim	(628.5 PTS)
23	Ready by Surgeon	(594 PTS)
24	Jammed With Arpeggios by Falciparum	(560.5 PTS)

25	Gold Lux by Xiny6581	(412 PTS)
26	Print All Your Crap by Ed	(409 PTS)

Visit the event here: http://www.c64.sk/files/sidcompo7/sidcompo7-results.html

Java SID Player Music Library V2

On December, kenchis released a port in Java of sidplay2 library and sidplay2 console player.
The library is very promising for all the Java fan. It improves day by day and it uses a even a graphical interfaces over the classical console.

```
+-----------------------------------------------------+
| Java SIDPLAY - Music Player & C64 SID Chip Emulator  |
|          Sidplay V2.0.8, libsidplay V2.1.1           |
+-----------------------------------------------------+
| Title        :                      Against My Enemy |
| Author       :                          Kjell Nordbø |
| Released     :                1996 SHAPE/Blues Muz' |
+-----------------------------------------------------+
| File format  :        PlaySID one-file format (PSID) |
| Filename(s)  :                  Against_My_Enemy.sid |
|              :                                       |
| Condition    :                             No errors |
| Playlist     :                       1/1 (tune 1/1[1]) |
| Song Length  :                               UNKNOWN |
+-----------------------------------------------------+
```

Download from http://sourceforge.net/projects/jsidplay2/

XSIDPLAY2

Developed at http://www.sourceforge.net/projects/xsidplay2, this is the second version of XSidlay (the historical Sid Linux player), that was no more developed by the author.

Version 2.0.0
Based onto xsidplay 1.6.5.2

- SidId v1.7 support
- Remove some memory related bug into TSID2 patch
- Remove a time related bug (about time resolution) when using libsidplay2

Version 2.0.1

- Clear list in SidId before load a configure file
- Allow to compile correctly if TSID2 is disable
- Add experimental support for SDL sound library

Version 2.0.2

- Add LCD display for songlength
- Allow to compile correctly with libsidplay1

Rockbox

Rockbox is an open source firmware for mp3 players, written from scratch. It runs on a wide range of players:

- **Apple**: 1st through 5.5th generation iPod, iPod Mini and 1st generation iPod Nano (*not the Shuffle, 2nd/3rd gen Nano, Classic or Touch*)
- **Archos**: Jukebox 5000, 6000, Studio, Recorder, FM Recorder, Recorder V2 and Ondio
- **Cowon**: iAudio X5, X5V, X5L, M5 and M5L
- **iriver**: H100, H300 and H10 series
- **SanDisk**: Sansa c200, e200 and e200R series (*not the v2 models*)
- **Toshiba**: Gigabeat X and F series (*not the S series*)

The interesting things about this project is that it play Sid file. Look at http://www.rockbox.org/

C64.sk Music Cover Compo

This was the first cover compo organized by www.64.sk

1 Elysion by Steven Diemer (A-Man/Xenon) (3:36) (738 PTS)
2 Cauldron II Sinus Milieu Studie by Linus (Sascha Zeidler) (4:05) (725.5 PTS)
3 Englishman In New York by Josep Barwick (Stainless Steel) (717.5 PTS)
4 Atlantic Reloaded Italo Disco by Randall (2:16) (650.5 PTS)
5 Galaxy Bounce (Tomb Raider Mix) by Kamil Wolnikowski (Jammer) (631 PTS)
6 Everybody Everybody by MSK Fanta Mitch (3:19) (626 PTS)
7 Dreamlights by Freedom (Marcello Marsetti) (2:43) (592.5 PTS)
8 Blitzzurueck Wahlmoeglichkeiten1 by Vincent Merken (_V_) (3:20) (584.5 PTS)
9 Mama by Stefan Uram (Orcan) (3:10) (520 PTS)
10 metroid_tune_5.sid by dalezy/triad (514 PTS)
11 The Great Destroyer by Stellan Andersson (Dane) (3:16) (502.5 PTS)
12 Burning Heat by Aegis501.5 PTS)
13 Could_You_Be_Loved by Rafal (Surgenon) (500.5 PTS)
14 Queen of Rain by Lars Hutzelmann (The Blue Ninja/DOS) (3:32) (488 PTS)
15 Nowhereland by Richard Bayliss (The New Dimension) (3:40) (470.5 PTS)
16 Daup of Pink Paint by Hein/Focus (Hein Holt) (0:55) (437.5 PTS)
17 Kate and Martin by Peter Bergstrand (3:40) (428 PTS)
18 Magnum Theme by Nico van der Zijden (Vai/Slash Design) (2:48) (379.5 PTS)
19 Chasing Cars by Andrew Fisher (Merman/POL/ROLE) (6.:10) (375.5 PTS)
20 Diamond in the Night by hukka (Joel Toivonen) (370.5 PTS)
21 Assault On Precint 13 by Andrew Lemon (Ne7/Triad) (350.5 PTS)
22 Intro A-team by Tim (bordeaux) (279.5 PTS)
23 Over The Rainbow by deizi (Tommi Lehtimäki) (1:32) (208.5 PTS)

More information can be found at http://www.c64.sk

HVSC update 48

Update 48 of HVSC (www.hvsc.c64.org) was released in March 2008.

This time we have music from:

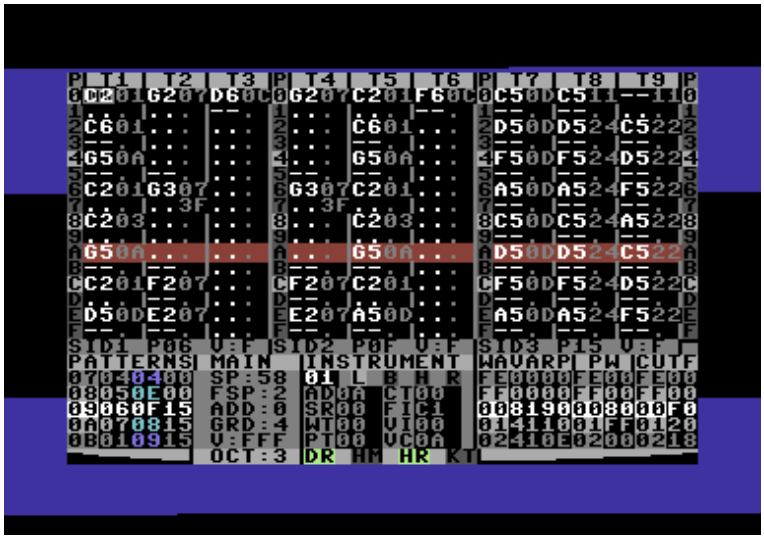
- Vandalism News 49
- Silesia party
- The 82 Ditties demo by Bluez Muz
- unreleased tunes by Adam Gilmore
- John Stormont (NEW)
- C64.sk SIDcompo 7
- Aegis (aged sweet 13 years old!) Jeroen Tel
- Linus
- Dwayne Bakewell
- Richard Bayliss
- Conrad (CRD)
- Froyd
- Adam Gilmore
- Goto80

After this update, the collection should contain 35,030 SID files!

This update features (all approximates):

```
 918 new SIDs
  42 fixed/better rips
   0 fixes of PlaySID/Sidplay1 specific SIDs
  12 repeats/bad rips eliminated
 370 SID credit fixes
 101 SID model/clock infos
  11 tunes from /DEMOS/UNKNOWN/ identified  :-)
  37 tunes moved out of /DEMOS/ to their composers' directories
  15 tunes moved out of /GAMES/ to their composers' directories
```

HERMIT 3SID-TRACKER 2008



Hermit Soft releases in May 2008 a new tracker that can handle 3 sid chips.

It can control 3 SIDs simultaneously on $d400,$de00,$df00 (yet). 9 polyphony, 3 filters. (no need arpeggio to have chords)

The tracker itself is a modern fast-tracker-like editor, which has additional functions (for example: the edit cursor and play-cursor,loop-play with F7,etc.).

You can get it from here:
http://noname.c64.org/csdb/release/?id=66065

HardSID 4U

The HardSID 4U is the most powerful SID synthesizer since the legendary C64!

- Two CPUs and main memory
- Updateable firmware over USB
- Full automation of all VSTi parameters
- USB connection (compatible with both 2.0 & 1.1)
- Isochronous USB endpoint for low-latency audio
- VSTi interface with 8000Hz update rate on all SID registers (free)
- Wave-in VSTi pin for routing 8000Hz signals to any registers (example: playing other VSTi's sound on the volume register)
- Superior sound quality (..it is a HardSID!)
- Support for up to four SID chips (6581/8580/6582 in any combination)
- Microsoft Vista compatible (drivers for Win2000/XP/Vista)

..and if you're a C64 fan

- Cycle-accurate playback of your favorite SID tunes
- Digitized sound + high-speed playback with low CPU utilization
- Seamless playback of .sid tunes while you work on your PC by providing a huge playback buffer for non-VSTi applications

The HardSID 4U Studio Edition

- Full physical separation of the SIDs from the USB driven circuits
- 100% elimination of EMI (Electromagnetic interference) noise that may come from your PC via USB

View all at: http://www.hardsid.com/hardsid_4u.php

Mihály Horváth (Hermit) Interview!

by Stefano Tognon

This time I go to interview Hermit, that has released a tracker that can control 3 sid chips. Something very interesting...

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

Hello, My real name is Mihály Horváth, I'm a 25 years old hungarian. I got electrical qualifications in secondary school, and I'm a communications-technician. But I was always interested in arts, music, painting, writing, and managed to get good competition results. Now the music is the first thing (after my Telecommunication Operator job) in my life, and I play live music with bassguitar and solo-singing in my Band. Any kind of music is preferred. In my free time I'm listening SID music and Seal,Sting,MJ,Jazz... music. I love nature, I have a girlfriend.

When and how did you discover the Commodore great machine called C64?

When I was 14, my primary school started computer education with 386 machines. I wanted a computer, and one of my friends sold me his C64 with a datasette. Lack of lot games made me to entertain myself with BASIC program-writing, even I wrote a GHOSTBUSTER-game and utils in BASIC in 1997. Then I started to learn the much more faster machine-code, and until 2000 I wrote many other user programmes, games, music collections (Mirage I,II,III).And the Mortal Kombat that is in the D64, wasn't completed... These "good times" ended, when I got my first 386 from a computer-service for my work. Later I sold my C64 with the new floppy-drive. I then wrote music with Fasttracker, then for some years I used IBM PC family to make music. On my last P4 is used SONAR and many VST and recorded live instruments, but had always problems wit Wi...ws. So after 5 years I bought back my first C64 to compose my music with it. Started to write the HMT-SYNTH,then the TRACKER.. Now I have 8 pieces of C64 already, and collect them ...

What is your actual way for creating programs for the commodore? Cross-develop into pc or directly coding in it?

I prefer the real machine with cartridge and floppy-drive for program-developing, especially in music editors, but if there isn't C64 near me I sometimes use the Vice to develop my programs.

Your 3-SID tracker is very amazing. When did your start to think to realize such a big challenge? It is very beautiful to have 2 sids into the C64, but 3 is absolutely fantastic.

Thanks. As I said before, I wanted to compose my live-musics on this safe-running C64 machine, but the first problem was the only 3 tracks, so I was searching 2SID composers, but there is only two yet: DMC 4.3 and Prophet64. DMC wasn't easy to make 2SID zaks, because of switching between SIDs. Prophet64 is a good thing, but I didn't manage to get it already, only the free demo without sequencer-part. And double framespeed was important too for me. So I decided to write a composer that's all in one for these desires. 3SID was the main new idea, and why not do it, if this great C64 can cope with it. I started it in Januray this year, finished this version in only 4 months.

Was you inspired by some existing tracker when you think of how to realize the your one?

I preferred the trackers because of my Fasttracker experiences. So Cybertracker, Odintracker, DMC and JCH's editor were the trackers that inspired me, and I downloaded many others to get ideas.. I invented some new things too: separate cursors for play and edit, all in one screen, advanced play-stop functions, pattern-play mode, loop-play mode...

What can you say about your future plan of releasing a 1-sid/ 2 sid version of the tracker? I think than many composers will be interesting into this.

Now it can be used as an 1SID or 2SID composer as in Cybertracker, but not specialized. Only leave the other SID-tracks empty. As the fixed new version will be ready with new functions, then 1SID and 2SID specialized versions will be ready too. I see it is worth it to spend more time with programming the tracker....

Have you some other interesting projects not jet released of just planned that you can talk to us?

Yes, I don't want to use IBM PC in the future as possible, so I'm developing Commodore 64 hardware and software to use it in my everydays.. Some other fanatic people does too. I now use for example 64HDD, want to use netcard-expansion later, to write e-mail from real c64, and so on.. The in USA-developed Commodore One is a good solution, it's all in one C64 expansion-motherboard with card-OS (Wings)... My actual work is the Hermit C64 Commander (HC), that will handle 64HDD,1541 and later pen drive as the Star Commander. My aim is to fanatically rise up again this good machine. What would be now, if its great line was continued? It only died because of wrong final leading of the firm... As I heard Tulip will manufacture C64 again with new logo.. I will seriously buy one....

Now some quick final (standard) questions:
Real machine vs emulator: what do you think about?

Vice is a very good emulator, but SID emulation isn't as fully good as the real. Analogue filter is important, the other important is the note-frequency dependent samplerate of the SID (44100 Hz is weak for pure analogue).

6581 vs 8580 chip: any (musical) preference?

I have a 3X8580 SID new machine and also an old 2X6581 SID machine. I tried both SID types, and I think that the 8580 is more accurate, but 6581 has good bass-filter behavior for older compositions. I prefer 8580.

What is the worst and the better sid you composed?

I didn't compose lots of SIDs yet, first I used Light Voices, and made very weakly composed musics (they are in my zak-collections called Mirage) 8 years ago. Today with my tracker I last composed the Earmind, which is my best SIDmusic now. But I have to train myself on SID a lot.

Who are your best sid authors?

The best for me is Jeroen Tel.
I think, he is one of the very-best composers in the world. I started to live-remix his composition and play them with my band in Hungary. But all Maniacs Of Noise members are nearly good too: Reyn Ouwehand, DRAX, Laxity, ... I also love Rob Hubbard music, and the best sounding and strange feelings of Shogoon zaks.
On C64 there are lots of extra-good authors too: Thomas Detert, Mark Cooksey, Martin Galway, Richard Joseph, David Whittaker, SLD, and so on.... SID (3 track) makes everyone to do his best to keep interest and quality..

What are the best sids ever in your opinion?

Jeroen Tel: Rubicon, Wiz, Aspar Grand Prix , Myth, Poseidon, Supre Macy, Golden Axe, Eliminator, Cybernoid I,II, Savage, ... Rob Hubbard: Zoids, Commando, Intern. Karate, Monty on Run, Star Paws, ...
Shogoon: Fun Factory, Zone of Darkness, Illmatic End, Muza do Dema, Love 2 ....

Other great SIDs: Barbarian, Last Ninja, BTTF3, Solitax end, Mysterious World, Stack Up, Krakout, Jackal, Dazzler, ...

And many-many very sophisticated compositions!!!!
(Todays pop-music is weak in good melodies, so the before mentioned zaks are much better and various for me...)

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

Everybody! Help us keep C64 alive, and make the feeling of a better world!!! READY.

Tiny Sid 2 (part 3)

by Stefano Tognon <ice00@libero.it>

With this article I finally ending to analysis of the last two entries of the Tiny Sid Compo II. All the engines present here are done by reverse engineering the code.

Plaster 512b

Plaster is the 512b entries of GRG and is the tune that take more points over all the others.

Here a description of the player:

- Autostarting at $0326 by setting the IRQ
- It disables IRQ and made a synchronization onto raster position
- It fills the zero page data all with 0, and copy some data to zero page while initializing
- It uses filter and cut frequency that varies from $3F to $FF up and down during the playing
- Song is based onto 2 tracks, one for voice 1 and one for voice 2 and 3
- One value in a track is a index to a pattern table of commands to execute, and if this value is negative, the absolute value is the transpose value to add for the next pattern.
- Each pattern command has this meanings:
    - value < 8 = set an instrument to use
    - 7 lower bit = note to play (if not an instrument)
    - negative = this is the last command (7 bit =command)
- One instrument is made by:
    - Control index value to use: an index to a series of control values to use for making timbre effect to the instrument
    - Attack/Decay value
    - Sustain/Release value
    - Pulse is varied by the player for making better effect
    - Hardrestart of note is performed by the player
- Note frequencies are built using the values of an octave
- Other effects are made inside the player for let voice 2 and 3 to be different (they play the same track)

```
   ; memory usage
; 84  note duration voice 1
; 86  pattern offset voice 1
; 87  freq low to add voice 1 (not used)
; 88  pattern index voice 1
; 89  track pattern index voice 1
; 8A  note to play voice 1
; 8B  note duration voice 2
; 8D  pattern offset voice 2
; 8E  freq low to add voice 2 (not used)
; 8F  pattern index voice 2
; 90  track pattern index voice 2
; 91  note to play voice 2
; 92  note duration voice 3
; 94  pattern offset voice 3
; 95  freq low to add voice 3 (used)
; 96  pattern index voice 3
; 97  track pattern index voice 3
; 98  note to play voice 3
; A0  pattern low address voice 1
; A2  control reg. index voice 1
; A3  instrument voice 1
; A4  note transpose value voice 1
; A5  note duration for control voice 1
; A6  control reg. value voice 1
; A7  pattern low address voice 2
; A9  control reg. index voice 2
; AA  instrument voice 2
; AB  note transpose value voice 2
; AC  note duration for control voice 2
; AD  control reg. value voice 2
```

```
; AE  pattern low address voice 3
; B1  control reg. index voice 3
; B2  instrument voice 3
; B3  note transpose value voice 3
; B4  note duration for control voice 3
; B5  control reg. value voice 3
; BE  actual wave low byte voice 1
; C0  direction 0=dec 1=inc
; C1  actual low freq.
; C2  actual high freq.
; C5  actual wave low byte voice 2
; CC  actual wave low byte voice 3
; D8  actual cut freq. hi byte


.org $0326
      .byte $2A, $03
      .byte $ED, $F6

.org  $032A
      sei
      tya
loop:
      sta  $0002,y              ; empty zero page variables
      iny
      bne  loop

      ldx  #$0E
      ldy  #$02

loopInit:
      lda  #$01
      sta  $84,x                ; note duration
      lda  track,y
      sta  $A0,x                ; pattern low address
      lda  wave_hi,y
      sta  $D403,x              ; Voice 1: Wave form pulsation amplitude (hi byte)
      dey

      lda  #$FF
      sbx  #$07                 ; undocument operation
      bpl  loopInit

      lda  #$03
      sta  $92                  ; note duration voice 3
      lda  #$1E
      sta  $95                  ; freq low to add voice 3

      lda  #$F1
      sta  $D417                ; Filter resonance control/voice input control
      lda  #$3F
      sta  $D418                ; Select volume and filter mode

playLoop:
                                ; made the cut freq goes from 3F to FF and back
      ldy  $D8                  ; actual cut freq. hi byte
      ldx  $C0                  ; direction 0=dec 1=inc
      beq  setDec
      iny
      cpy  #$FF
      bne  setCut
      dex
      beq  setCut
setDec
      dey
      cpy  #$3F
      bne  setCut
      inx
setCut:
      stx  $C0                  ; direction 0=dec 1=inc
      sty  $D8                  ; actual cut freq. hi byte
      sty  $D416                ; Filter cut frequency: hi byte
      ldx  #$0E                 ; voice 3

nextVoice:
      dec  $84,x                ; dec note duration
      bne  durNotZero

      ldy  $88,x                ; pattern index
      bne  patNotZero
      ldy  $86,x                ; pattern offset
      lda  $A0,x                ; pattern low address
      sta  trPos+1

notVoice1:
trPos:
      lda  track,y              ; load value of pattern
      bne  skipVEffect
```

```
      tay
      cpx  #$07                ; is voice 2?
      bne  notVoice2

                               ; transpose effect for voice 2
      lda  $AB                 ; note transpose value voice 2
      eor  #$F0
      sta  $AB                 ; note transpose value voice 2

notVoice2:
      cpx  #$00
      bne  notVoice1           ; is voice 1?

      lda  #$05                ; instrument 5
      sta  patIns              ; change pattern
      bne  notVoice1

skipVEffect:
      bpl  skipTranspose
      and  #$1F
      sta  $A4,x               ; note transpose value
      iny
      bne  notVoice1

skipTranspose:
      sta  $89,x               ; track pattern index
      iny
      sty  $86,x               ; pattern offset

patNotZero:
      ldy  $89,x               ; track pattern index
      lda  patTable-1,y
      sta  patPos+1
      ldy  $88,x               ; pattern index
      cpx  #$00                ; voice 1
      beq  skipVEffect2

      lda  $A3,x               ; instrument
      eor  #$07
      sta  $A3,x               ; instrument

skipVEffect2:
patPos:
      lda  $049E,y
      iny
      cmp  #$08                ; is an instrument?
      bcs  isNote
      sta  $A3,x               ; store instrument
      bne  skipVEffect2
isNote:
      pha
      and  #$7F                ; isolate the note
      clc
      adc  $A4,x               ; note transpose value
      sta  $8A,x               ; note to play
      pla
      bpl  notLast
      ldy  #$00                ; restart pattern as it is finished
notLast:
      sty  $88,x               ; pattern index
decNextVoice:
      lda  #$FF
      sbx  #$07                ; undocument operation
      bpl  nextVoice

loopSync:
      cmp  $D012               ; Reading/Writing IRQ balance value
      bne  loopSync
      jmp  playLoop

durNotZero:
      lda  $84,x               ; note duration
      pha
      cmp  #$01                ; test for hardrestart
      bne  skipHR

      sta  $D406,x             ; Generator 1: Sustain/Release
      lda  #$0F
      sta  $D405,x             ; Generator 1: Attack/Decay
      dec  $A5,x               ; note duration for control voice
skipHR:
      pla
      bpl  setCtrl

      sta  $A5,x               ; note duration for control
      lda  #$07
      sta  $84,x               ; note duration
      sta  $BE,x               ; actual wave low byte
```

```
      ldy  $A3,x                ; read instrument
      lda  controlIndex-1,y
      sta  $A2,x                ; control reg. index
      lda  AD_inst-1,y
      sta  $D405,x              ; Generator 1: Attack/Decay
      lda  SR_inst-1,y
      sta  $D406,x              ; Generator 1: Sustain/Release

setCtrl:
      ldy  $A2,x                ; control reg. index
      lda  control,y            ; read control reg
      bne  notZeroCtrl
      dey
      lda  $A6,x                ; control reg. value
      .byte $2C  ;  BIT  $A2F6

notZeroCtrl:
      inc  $A2,x                ; control reg. index

      sta  $A6,x                ; control reg. value
      and  $A5,x                ; note duration for control voice
      sta  $D404,x              ; Voice 1: Control registers
      lda  useFreq,y            ; use freq. for effect
      bpl  ignoreFreq

      and  #$7F                 ; isolate low 7 bits of high freq
      bpl  outHiFreq

ignoreFreq:
      lda  $8A,x                ; note to play
      pha
      and  #$0F                 ; take base note
      tay
      lda  hiFreq,y
      sta  $C2                  ; actual high freq.
      lda  loFreq,y
      sta  $C1                  ; actual low freq.
      pla                       ; take octave
      lsr
      lsr
      lsr
      lsr
      tay                       ; y=octave
      beq  skipDouble

doubleF:                        ; double the frequency
      lsr  $C2                  ; actual high freq.
      ror  $C1                  ; actual low freq.
      dey
      bne  doubleF

skipDouble:
      lda  $C1                  ; actual low freq.
      clc
      adc  $87,x                ; freq low to add
      sta  $D400,x              ; Voice 1: Frequency control (lo byte)
      lda  $C2                  ; actual high freq.
      adc  #$00

outHiFreq:
      sta  $D401,x               ; Voice 1: Frequency control (hi byte)

      lda  $BE,x                 ; actual wave low byte
      adc  #$18
      sta  $BE,x                 ; actual wave low byte

      sta  $D402,x               ; Voice 1: Wave form pulsation amplitude (lo byte)
      jmp  decNextVoice

track:
  .byte <track79, <track8F, <track8F

track79:
  .byte $83, $01, $01, $01
  .byte $01, $8A, $01, $01
  .byte $01, $01, $8B, $01
  .byte $01, $01, $01, $86
  .byte $01, $01, $85, $01
  .byte $06, $00

track8F:
  .byte $02, $02, $03, $03
  .byte $04, $04, $05, $05
  .byte $00

patTable:
  .byte <pat9E, <patAC, <patB6, <patC0, <patCA, <patA6
```

```
; value <8  = instrument
; negative  = last
; 7 low bit = note (if not instrument)
pat9E:
  .byte $02, $60, $01, $60
patIns:
  .byte $02, $60, $01, $E0

patA6:
  .byte $02, $60, $60, $05
  .byte $60, $E0

patAC:
  .byte $03, $3A, $33, $3A
  .byte $03, $36, $3A, $35
  .byte $36, $B3

patB6:
  .byte $03, $21, $35, $21
  .byte $03, $3A, $21, $38
  .byte $3A, $B5

patC0:
  .byte $03, $3B, $33, $3B
  .byte $03, $3A, $3B, $36
  .byte $3A, $B3

patCA:
  .byte $03, $3A, $4A, $3A
  .byte $03, $36, $3A, $33
  .byte $36, $CA

wave_hi:
  .byte $08, $08, $04            ; wave high value for each voice

AD_inst:
  .byte $00, $0F, $02, $E0, $08  ; attack/decay of instruments

SR_inst:
  .byte $A4, $FA, $8A, $8A, $E8  ; sustain/release of instruments

loFreq:
  .byte $1E, $18, $8B, $7E, $FA, $06, $AC, $F3, $E6, $8F, $F8, $2E

hiFreq:
  .byte $86, $8E, $96, $9F, $A8, $B3, $BD, $C8, $D4, $E1, $EE, $FD

control:
  .byte $09, $81, $41, $00
  .byte $09, $81, $11, $11, $11, $11, $11, $00
  .byte $09, $81, $41, $40, $40, $80

controlIndex:
  .byte $00, $04, $02, $02, $0C

; if <0, 7bits=high freq. to use for effect
useFreq:
  .byte $00, $FF, $00, $02, $00
  .byte $FF, $8B, $89, $86, $84
  .byte $00, $0A, $00, $FD, $8E
  .byte $8C, $8A, $FF, $00, $00
```

Back To Basics

This is the 256 bytes tune wrote by Jaymz Julian.

Here a description of the player:

- Autostarting at $0326 by setting the IRQ
- It synchronization onto raster position
- It copies some data to zero page while initializing for access them with short instructions
- High byte of filter cut of frequency and high byte of wave pulse (that will be used a little shifted for each voices) are changed every fixed time
- Voice 2 use tables of values for high frequencies and control register to use
- Pattern of values are used for Voice 1 and 3 frequencies low or high)
- Note duration are triggered between 2 values during the play

```
.org $0326
  .byte $2A, $03
  .byte $ED, $F6

.org $032A
      lda  #$1F
      sta  $D418              ; set volume
      ldx  #$46

loopCopy:                     ; copy data to zero pages
      lda  data,x
      sta  $01,x
      dex
      bne  loopCopy

mainLoop:
      ldx  #$00               ; voice 1
      jsr  play

      dec  $3A                ; dec delay for continue effect
      bne  skipInc

      lda  #$04
      sta  $3A                ; restore delay for continue effect
      inc  $47                ; inc Wave form pulsation amplitude (hi byte)/ Filter cut frequency: hi byte

skipInc:
      ldx  #$07               ; voice 2
      jsr  play
      ldx  #$0E               ; voice 3
      jsr  play

      lda  #$F4
      sta  $D417              ; Filter resonance control/voice input control
raster:
      cmp  $D012              ; Reading/Writing IRQ balance value
      bne  raster
      beq  mainLoop

play:
      txa                     ; differentiate the effect for the 3 voices
      adc  $47                ; Wave form pulsation amplitude (hi byte)/ Filter cut frequency: hi byte
      sta  $D403,x            ; Voice 1: Wave form pulsation amplitude (hi byte)
      sta  $D416              ; Filter cut frequency: hi byte

      lda  #$80
      sta  $D402,x            ; Voice 1: Wave form pulsation amplitude (lo byte)
      ldy  $36,x              ; read actual pattern index

      dec  $38,x              ; actual duration (delay)
      beq  readPat

      cpx  #$07               ; Voice 2
      bne  quit

      ldx  $3B                ; pattern value (=index starting from $1A)
      lda  $1A,x
      bne  notZeroVal

      lda  $1B,x              ; load restarting position
      tax
      lda  $1A,x              ; use value of new position

notZeroVal:
```

```
      sta  $D408                ; Voice 2: Frequency control (hi byte)
      lda  $28,x
      sta  $D40B                ; Voice 2: Control registers
      inx
      stx  $3B                  ; pattern value (=index starting from $1A)
quit:
      rts

readPat:
      lda  $0002,y              ; read pattern value
      bpl  notEndPattern

      lda  #$E9
      sta  $D406,x              ; Generator 1: Sustain/Release
      lda  #$41
      sta  $D404,x              ; Voice 1: Control registers

      inc  $2E                  ; counter
      lda  $2E
      and  #$0F                 ; test for every 16 ticks
      bne  not16

      sta  $D406,x              ; Generator 1: Sustain/Release
      sta  $D404,x              ; Voice 1: Control registers

      lda  $1C                  ; Voice 2 frequency high byte
      eor  #$0C                 ; make effect
      sta  $1C

not16:
      ldy  $37,x                ; restore pattern index
      sty  $36,x                ; actual pattern index
      bpl  readPat

notEndPattern:
      cpx  #$07                 ; voice 2
      bne  notV2

      sta  $3B                  ; pattern value  (=index starting from $1A)
      bne  notJump

notV2:
      bcc  isV1
      inc  $41
      bne  isV3
      inc  $42
isV3:
      sta  $D400,x              ; Voice 1: Frequency control (lo byte)
      and  #$0F
isV1:
      sta  $D401,x              ; Voice 1: Frequency control (hi byte)

notJump:.
      lda  $35,x                ; load duration (delay)
      sta  $38,x                ; store actual duration (delay)
      eor  $39                  ; change duration length
      sta  $35,x                ; store duration (delay)
      lda  ($33,x)
      and  #$07
      sta  $36                  ; actual pattern index position
      inc  $36,x
data:
      rts

; $02:
  .byte $00, $0F, $14, $22
  .byte $24, $28, $1E, $FF
; $0A
  .byte $01, $07, $0B, $0B
  .byte $07, $01, $01, $0B
  .byte $FF
; $13:
  .byte $05, $0A, $05, $09
  .byte $05, $68, $05
;$1A   Voice 2 frequency high byte
  .byte $77, $FF, $28, $2F, $3C, $00 $02
  .byte $FF
  .byte $0B, $0B
  .byte $B5, $FF
; $26
  .byte $0A
  .byte $00
; $28  Voice 2 control register
  .byte $0C, $81, $41, $41, $41, $00
; $2E
  .byte $00                     ; counter
  .byte $81, $41, $40, $80, $81
; $34
```

```
  .byte $15
; $35
  .byte $06                     ; duration (delay)
; $36
  .byte $00                     ; Voice 1 actual pattern index position
; $37
  .byte $00                     ; Voice 1 restore pattern index position
; $38
  .byte $01                     ; actual duration (delay)
; $39
  .byte $08                     ; for changing the duration length
; $3A
  .byte $04                     ; delay for continue effect
; $3B
  .byte $00                     ; pattern value (=index starting from $1A)
  .byte $06                     ; duration (delay)
  .byte $08                     ; Voice 2 actual pattern index position
  .byte $08                     ; Voice 2 restore pattern index position
  .byte $01                     ; actual duration (delay)
  .byte $08
; $41
  .byte $10
; $42
  .byte $A0
  .byte $06                     ; duration (delay)
  .byte $11                     ; Voice 3 actual pattern index position
  .byte $11                     ; Voice 3 restore pattern index position
  .byte $01
; $47       Wave form pulsation amplitude (hi byte)/ Filter cut frequency: hi byte
```

Conclusion

Finally we had seen all the Tiny Sid 2 entries.

Remember that writing tiny music is a sort of programming art: you have to customize your player according to the instruments to use, with the music effect you want to achieve and the melody of your tune, arranging all the code in a manner that must be the smallest one. The work behind this task could be very hard, and maybe with all the examples we showed during this and the other issues you now have a precise idea of what this means.

Ripping Trivia Arcade

by Stefano Tognon <ice00@libero.it>

In this article I go to describe all the steps involved in ripping the music of The Trivia Arcade game.

The program comes with 2 disks:

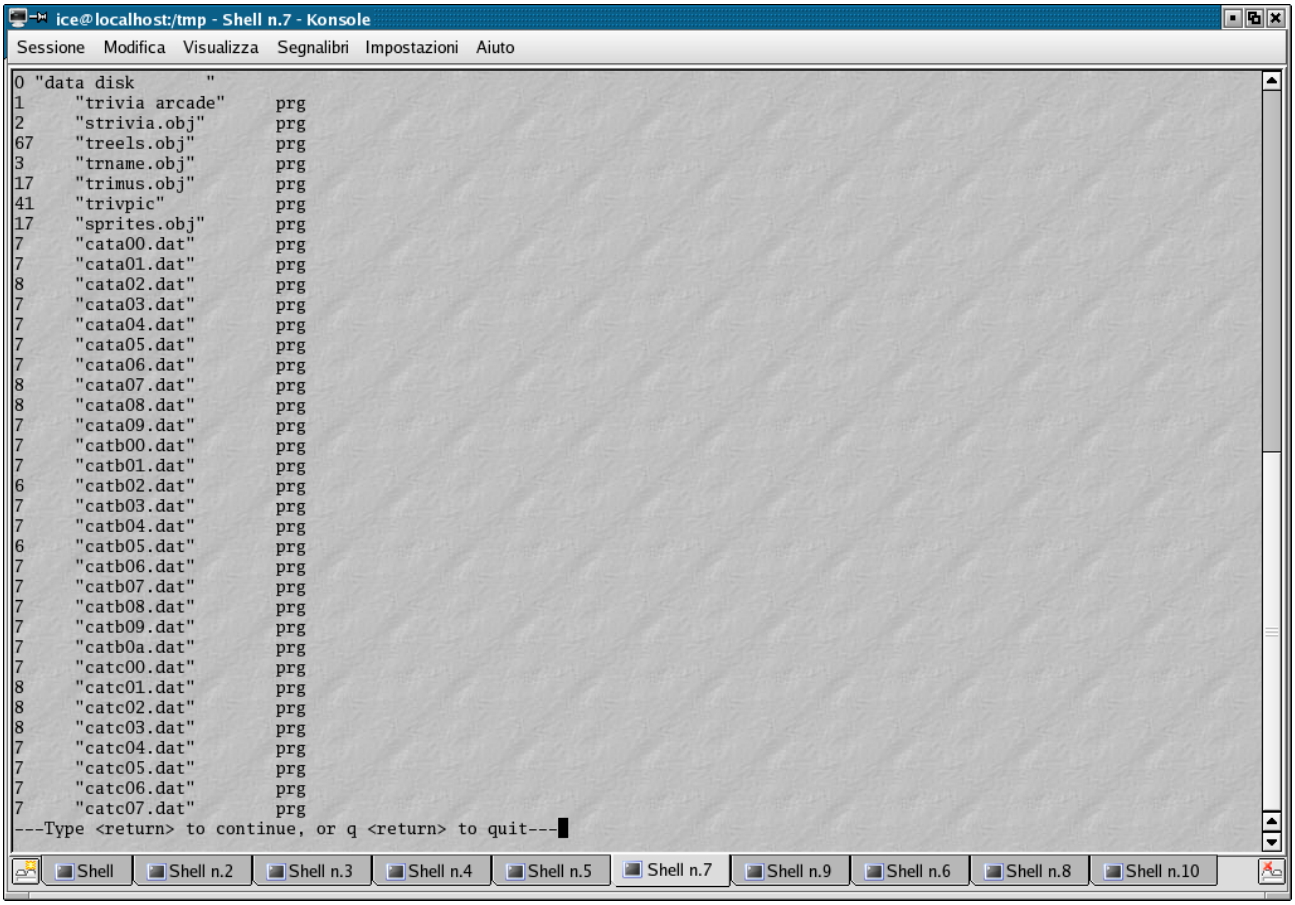
- TRIVIAA0.D64
- TRIVIAA1.D64

that are full of files used by the program.

But lets starts with the ripping action...

Starting

When you start the program, a presentation screen appears an some files from the disk are loaded during this process. Maybe the best things is to look at those files present in the disk, as they could give us some information about this program.

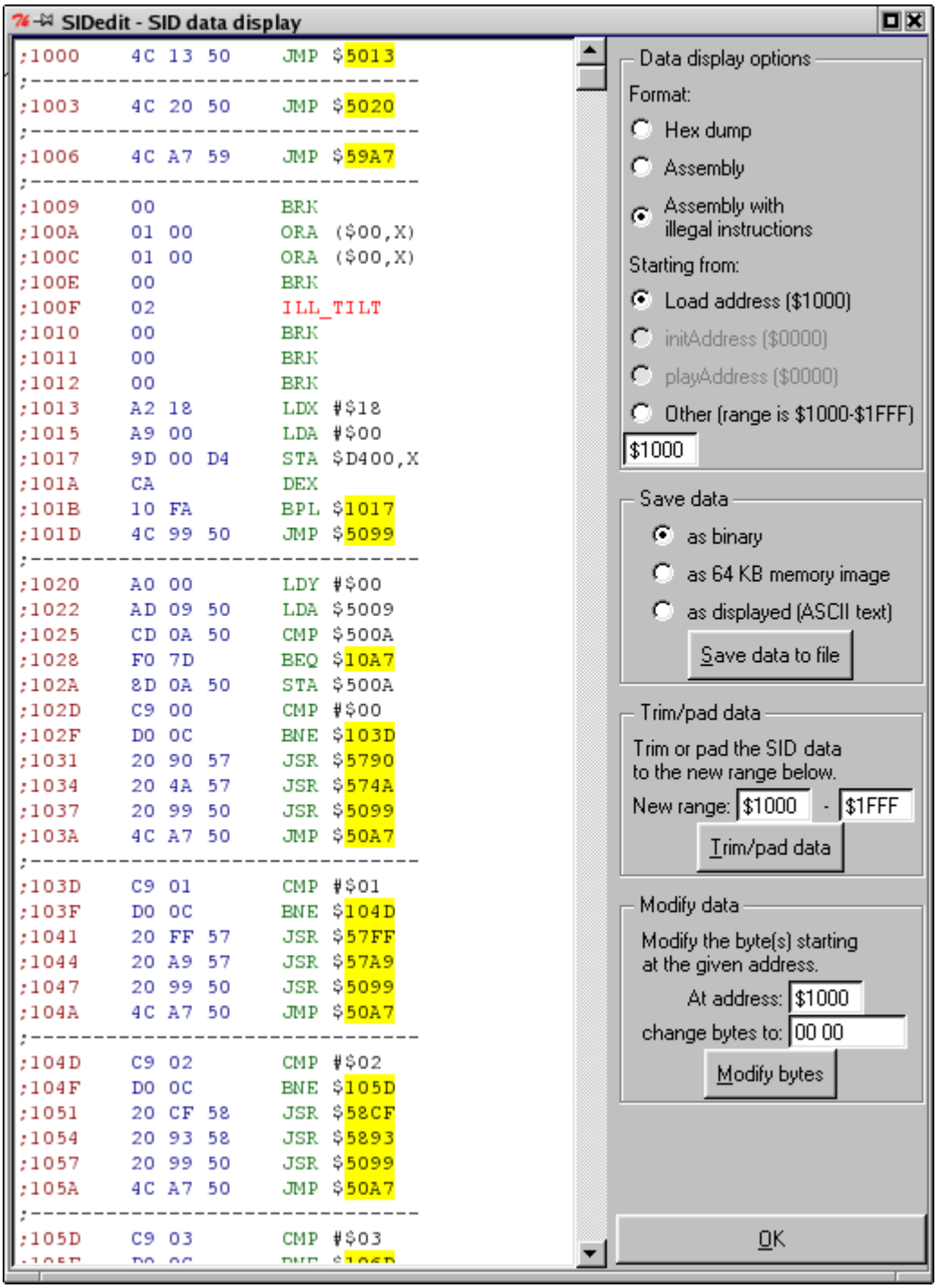


That's an easy task, from a console just type:

- c1541 TRIVIAA0.D64
- list
- extract
- quit

With the *list* command we see that there is a file called *trimus.obj* that can be a Trivia Music Object file, or in other word the music data non embedded into the program, but a separate file to load.

With *extract* we then have all the files extracted from the disk as PRG separated programs, so we can better manage them.



Now just run LaLa's SIDedit perl program to the *trimus* extracted files and go to data display:

- Load address of the files is $1000
- There are 3 jumps at the start of the code
- The code seems to be relocated at $5000 and not to the address the files comes with.

With this simple analysis we can hypothesize that the music code must be loaded to $5000 and that it not uses a standard IRQ calling (+0=init, +3=play), as there are 3 pointers in the beginning of music data. That also means that our task that with the presence of a obj file with music seemed easy, is now more difficult.

Now run vice and set some breakpoints to 5000, 5003, 5006 before loading the program.

```
(C:$eeac) break 5000
BREAK: 1 C:$5000   enabled
(C:$eeac) break 5003
BREAK: 2 C:$5003   enabled
(C:$eeac) break 5006
BREAK: 3 C:$5006   enabled
```

We will see these sequence of break:

```
#1 (Break) .C:5000  4C 13 50   JMP $5013
#2 (Break) .C:5003  4C 20 50   JMP $5020
#3 (Break) .C:5006  4C A7 59   JMP $59A7
#3 (Break) .C:5006  4C A7 59   JMP $59A7
#3 (Break) .C:5006  4C A7 59   JMP $59A7
#3 (Break) .C:5006  4C A7 59   JMP $59A7
#2 (Break) .C:5003  4C 20 50   JMP $5020
#3 (Break) .C:5006  4C A7 59   JMP $59A7
#3 (Break) .C:5006  4C A7 59   JMP $59A7
#3 (Break) .C:5006  4C A7 59   JMP $59A7
#3 (Break) .C:5006  4C A7 59   JMP $59A7
```

...

5000 is called only one time, so it must be a music player initialization.
5003 is called one time, then 5006 is called 4 times before 5003 is called again.

But how is the sequence regarding to the irq? Just see what there is at $314:

```
(C:$5003) m 314
>C:0314  3f 2e 66 fe  08 2f 4a f3  91 f2 0e f2  50 f2 33 f3   ?.f../J.....P.3.
```

So the irq call location is at $2e3f, and here we set a new breakpoint:

```
#4 (Break) .C:2e3f  78          SEI
#2 (Break) .C:5003  4C 20 50    JMP $5020
#3 (Break) .C:5006  4C A7 59    JMP $59A7
#4 (Break) .C:2e3f  78          SEI
#3 (Break) .C:5006  4C A7 59    JMP $59A7
#4 (Break) .C:2e3f  78          SEI
#3 (Break) .C:5006  4C A7 59    JMP $59A7
#4 (Break) .C:2e3f  78          SEI
#3 (Break) .C:5006  4C A7 59    JMP $59A7
...(restart)
```

Now we see that 5006 is called ad each tick, and 5003 is called one time over 4 before the 5006.

So, I try this simply player, but no one note were played:

```
; player for trivia

 processor 6502

INIT  = $5000
PLAY1 = $5003
PLAY2 = $5006


 .org 2049

  .byte $0b,$08,$e8,$03,$9e,"2061",0,0,0

  .org 2061


     lda  #0
     jsr  INIT

sei
     lda  #<raster
     sta  $0314
     lda  #>raster
     sta  $0315
     lda  #50                       ;Set low bits of raster
     sta  $d012                     ;position
     lda  $d011
     and  #$7f                      ;Set high bit of raster
     sta  $d011                     ;position (0)
     lda  #$7f                      ;Set timer interrupt off
     sta  $dc0d
     lda  #$01                      ;Set raster interrupt on
     sta  $d01a
     lda  $dc0d                     ;Acknowledge timer interrupt
     cli
aaa: jmp  aaa

 .org $2000

raster:
     inc  tick
     lda  tick
     and  #$03
     bne  skip
     jsr  PLAY1                     ; play first only one time over 4
skip
     jsr  PLAY2


     dec  $d019
     jmp  $ea31

tick:
  .byte $FF

 .org $4FFE
 .incbin trimus.dat
```

So it is the case to see some part of the music code to understand how it work:

```
5000    4C 13 50    JMP $5013


5013    A2 18       LDX #$18
5015    A9 00       LDA #$00
5017    9D 00 D4    STA $D400,X
501A    CA          DEX
501B    10 FA       BPL $5017
501D    4C 99 50    JMP $5099


5099    A9 01       LDA #$01
509B    A0 00       LDY #$00
509D    8D 10 50    STA $5010
50A0    8D 11 50    STA $5011
50A3    8D 12 50    STA $5012
50A6    60          RTS
```

It only cleans some memory area, no one passed value is taken, so this is an init that simple clear memory, and it is not related to sub-tune initialization.

So, we try to look at 5003 first, as it is called less time.

```
5003    4C 20 50    JMP $5020


5020    A0 00       LDY #$00
5022    AD 09 50    LDA $5009
5025    CD 0A 50    CMP $500A
5028    F0 7D       BEQ $50A7
502A    8D 0A 50    STA $500A
502D    C9 00       CMP #$00
502F    D0 0C       BNE $503D
```

Here we see that location 5009 is compared to location 500A and if different the value is stored. Maybe location 5009 is for sub-tune selection and 500A is the value of the current being played tune. So we can made a little test: run vice of the player we build, then enter the monitor and write:

- > 5009 2
- x

Now we heart something: the notes seems the right one, only the velocity is very low compared to the expected one. Maybe the IRQ frequency of call is not 1 per frame (multispeed tune), or it is done by CIA programming.

So now it is the case to see the IRQ routine more carefully:

```
2DE5  78         SEI
2DE6  A9 08      LDA  #$08
2DE8  8D 18 03   STA  $0318       Vector: Not maskerable Interrupt (NMI)
2DEB  A9 2F      LDA  #$2F
2DED  8D 19 03   STA  $0319       Vector: Not maskerable Interrupt (NMI)
2DF0  A9 7F      LDA  #$7F
2DF2  8D 0D DC   STA  $DC0D       Interrupt control register CIA #1
2DF5  A9 3F      LDA  #$3F
2DF7  8D 14 03   STA  $0314       Vector: Hardware Interrupt (IRQ)
2DFA  A9 2E      LDA  #$2E
2DFC  8D 15 03   STA  $0315       Vector: Hardware Interrupt (IRQ)
2DFF  A9 01      LDA  #$01
2E01  8D 12 D0   STA  $D012       Reading/Writing IRQ balance value
2E04  AD 11 D0   LDA  $D011       VIC control register
2E07  29 7F      AND  #$7F
2E09  8D 11 D0   STA  $D011       VIC control register
2E0C  A9 01      LDA  #$01
2E0E  8D 1A D0   STA  $D01A       IRQ mask register
2E11  A9 DF      LDA  #$DF
2E13  8D 76 0C   STA  $0C76       Normal space for BASIC programs
2E16  20 00 50   JSR  $5000       Normal space for BASIC programs
2E19  58         CLI
2E1A  60         RTS
```

It uses raster interrupt, so we must analyze the irq routine even more:

```
2E3F  78         SEI
2E40  D8         CLD
2E41  AD 12 D0   LDA  $D012       Reading/Writing IRQ balance value
2E44  C9 01      CMP  #$01
2E46  D0 03      BNE  $2E4B       Normal space for BASIC programs
2E48  4C 6A 2E   JMP  $2E6A       Normal space for BASIC programs
2E4B  C9 41      CMP  #$41
2E4D  D0 03      BNE  $2E52       Normal space for BASIC programs
2E4F  4C 6F 2E   JMP  $2E6F       Normal space for BASIC programs
2E52  C9 6E      CMP  #$6E
2E54  D0 03      BNE  $2E59       Normal space for BASIC programs
2E56  4C 74 2E   JMP  $2E74       Normal space for BASIC programs
2E59  C9 C3      CMP  #$C3
2E5B  D0 03      BNE  $2E60       Normal space for BASIC programs
2E5D  4C 79 2E   JMP  $2E79       Normal space for BASIC programs
2E60  C9 D2      CMP  #$D2
2E62  D0 03      BNE  $2E67       Normal space for BASIC programs
2E64  4C 7E 2E   JMP  $2E7E       Normal space for BASIC programs
2E67  4C A1 2E   JMP  $2EA1       Normal space for BASIC programs

2E6A  A9 41      LDA  #$41
2E6C  4C D2 2E   JMP  $2ED2       Normal space for BASIC programs
2E6F  A9 6E      LDA  #$6E
2E71  4C D2 2E   JMP  $2ED2       Normal space for BASIC programs
2E74  A9 C3      LDA  #$C3
2E76  4C D2 2E   JMP  $2ED2       Normal space for BASIC programs
2E79  A9 D2      LDA  #$D2
2E7B  4C D2 2E   JMP  $2ED2       Normal space for BASIC programs
```

```
2ED0  A9 01     LDA  #$01
2ED2  8D 12 D0  STA  $D012      Reading/Writing IRQ balance value
2ED5  AD 11 D0  LDA  $D011      VIC control register
2ED8  29 7F     AND  #$7F
2EDA  8D 11 D0  STA  $D011      VIC control register
2EDD  AD 19 D0  LDA  $D019      Interrupt indicator register
2EE0  8D 19 D0  STA  $D019      Interrupt indicator register
```

I do not report all the irq routine, but the point we need is that this is a multispeed tune and the vic raster position interested are:

$01, $41, $6E, $C3, $D2

As, however, we see before that there is 4 irq call for restarting the music cycle, so maybe one of this raster line is not actually used. In this case we just need to run vice again and set a break-point to 2E3F that is the irq. Looking at the raster line when there is the break, we see:

tune #1: 001 065 110 196
tune #2: 001 065 110 196
tune #3: 001 065 110 196
tune #4: 001 065 110 196
tune #5: 001 065 110 196

So $D2 is not actually used.

Combining the above information, we could make our player as:

```
; player for trivia

 processor 6502

INIT  = $5000
PLAY1 = $5003
PLAY2 = $5006

 .org 2049

  .byte $0b,$08,$e8,$03,$9e,"2061",0,0,0

  .org 2061
                          ; a=number of tune to paly
     tax
     inx
     stx  $5009           ; store subtune to play
     inx
     stx  $500a

     jsr  INIT

     sei
     lda  #<raster
     sta  $0314
     lda  #>raster
     sta  $0315
     LDA  #$7F
     STA  $DC0D       ; Interrupt control register CIA #1
     LDA  #$01
     STA  $D012       ; Reading/Writing IRQ balance value
     LDA  $D011       ; VIC control register
     AND  #$7F
     STA  $D011       ; VIC control register
     LDA  #$01
     STA  $D01A       ; IRQ mask register
     cli
aaa: jmp  aaa

 .org $2000

raster:
     lda  $d012
     cmp  #01
     bne  next1
     jsr  PLAY1                    ; play first only one time over 4
     lda  #$41
     sta  $d012
     jmp  skip
next1:
     cmp  #$41
```

```
     bne  next2
     lda  #$6E
     sta  $d012
     jmp  skip
next2:
     cmp  #$6E
     bne  next3
     lda  #$C3
     sta  $d012
     jmp  skip
next3:
     lda  #$01
     sta  $d012
skip
     jsr  PLAY2
     dec  $d019
     jmp  $ea31

 .org $4FFE
 .incbin trimus.dat
```

SID file

Now we want to create the sid file of the above sid tune that was in binary format.

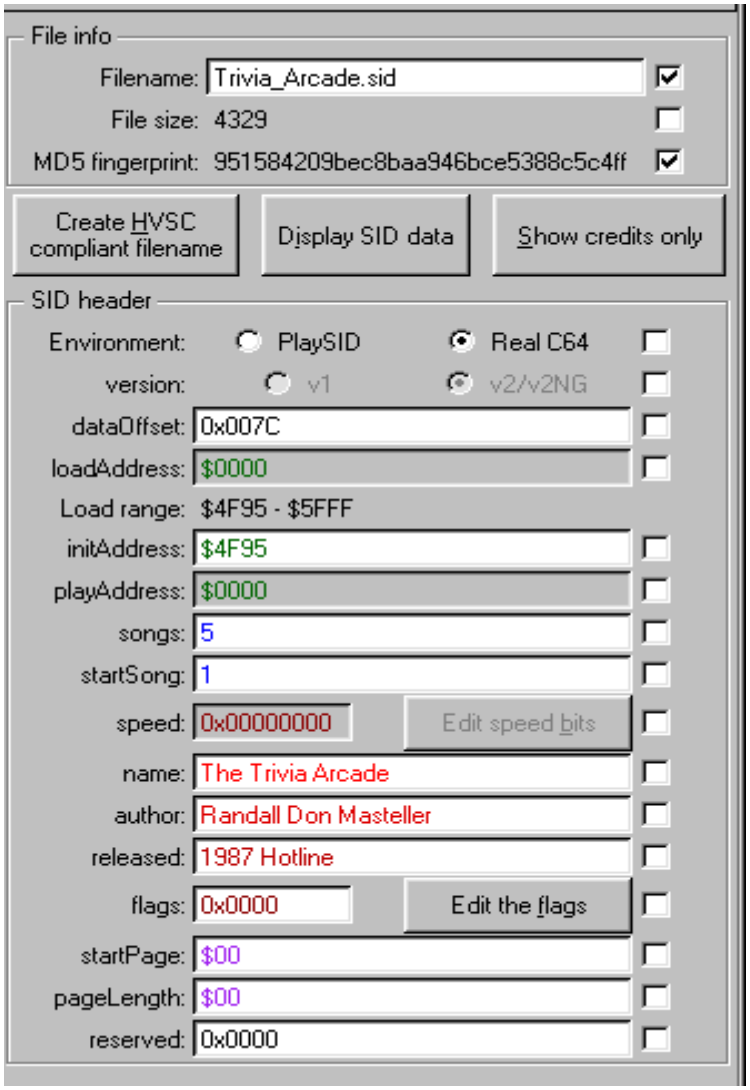
First of all we need to know the number of tunes that are present. Testing with the value of 5009, we see that:

- #0 it plays a tune with some strange sound at beginning and after it play good sound, only if this is not the first tune to be played, otherwise it is with no sound.
- #1..#4 they play good tune
- #5 a tune similar to previous
- >=#6 seems to play one of the above tune in every case

That is why I put *inx* into the code (skip tune 0).

So, now it simple: our player must be compacted for saving space in memory, by move the code near 5000 where there is the music data (e.g 4f95), and the SYS basic call can be removed too.

After that, just run SIDedit and let him create a standard sidplay header, then fill the options like in the image (Real C64 tune, init address, 4 subtunes). Save the sid file and the task is finished.



Conclusion

Ripping is alway a way to make you enjoy a sid tune into your sidplayer, so I hope that looking even at how this is done is a good way for new rippers to learn how to do that.

Sin 12 end