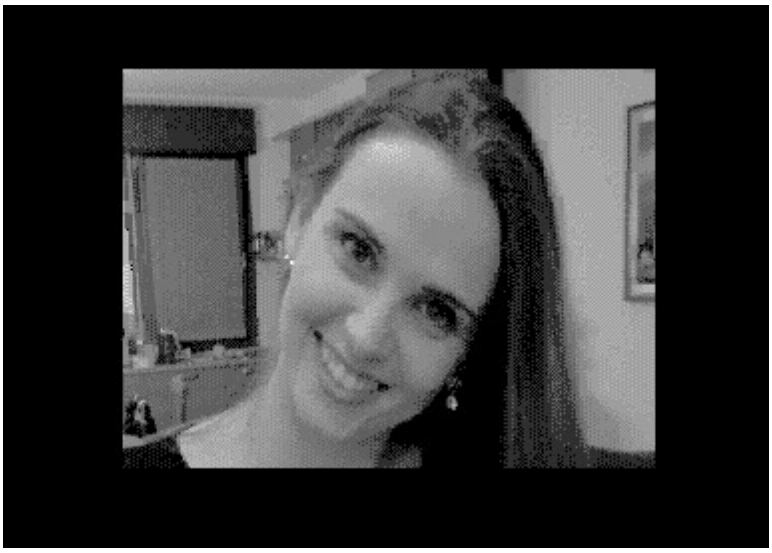


“Light”

Vice snapshot with Vice palette

Made with the GIMP from a ML photo
and converted to C64 320x200
MUIFLI
by Stefano Tognon
in 2015

“Always positive”
...

Free Software Group

Gwin 15
version 1.00
31 October 2015

SIDin Contents

General Index

Editorials

Stefano Tognon <ice00@libero.it>

Hi, again.

It is Halloween. The best day release something related to the Commodore 64.

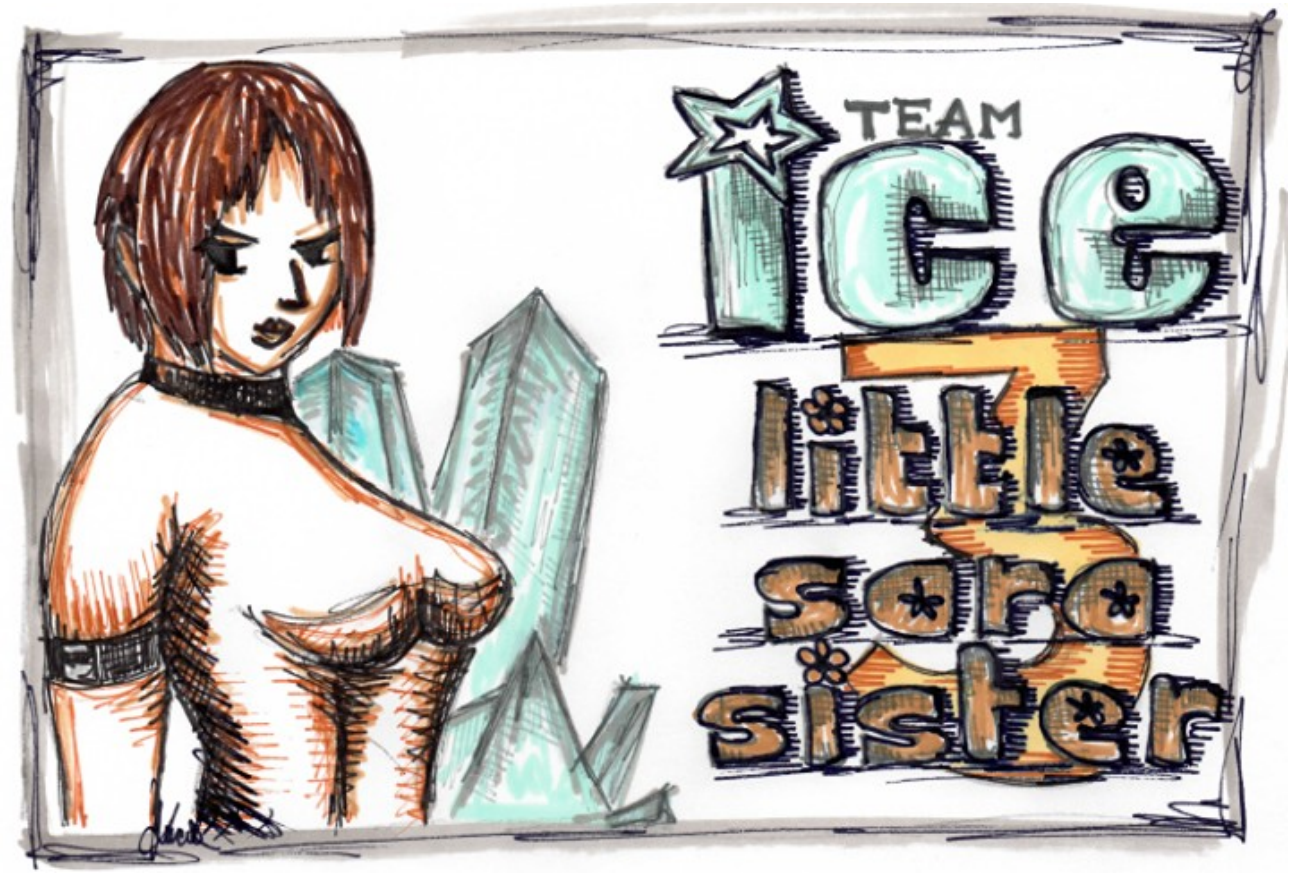
Well, this number were planned to be released at the end of May, but due too many activities it is slipped until today. However it is more early that you can expected, is it true??

First, go to read the Matt Gray interview, so you can see why we release the Serpent Demo source code! Then before going out for Halloween night, just read the JSidPlayer easy article.

On the SID side you may know that PSID file were extended to allow 3 SID chips to play together. For this XSidPlay2 now is able to play such files.

The IceTam site is not jet completed, but now you can find this issue in the new SIDin page: http://iceteam.altervista.org/sidin.php

The other new is that “Little Sara Sister” is growing and from Commodore 64 she is going to Java devices (PC, Tablet, Console), but we try to make the electronic music inside the game to be a little like SID sound :)
You can follow his develop on the FB page: https://www.facebook.com/Little-Sara-Sister-3-51109



Bye
S.T.

News

Some various news of players, programs, and competitions:

- CGSC v1.30
- XSidPlay2 2.1.7
- Tel Me More
- HVSC #63
- Project Sidologie
- Back in Time Symphonic Collection

CGSC v1.30

The Compute's Sid Collection has just been updated on 28 March 2015.

It now contains an extra 506 MUS, 4 STR & 163 WDS files.
The totals are now 14397 MUS files, 4139 STR files and 4948 WDS files.

Download from www.c64music.co.uk

HVSC #63

Released in June 2015 High Voltage Sid Collection update 63.

After this update, the collection should contain 46,851 SID files!

```
This update features (all approximates):
798 new SIDs
121 fixed/better rips
3 repeats/bad rips eliminated
412 SID credit fixes
67 SID model/clock infos
8 tunes from /DEMOS/UNKNOWN/ identified
19 tunes from /GAMES/ identified
24 tunes moved out of /DEMOS/ to their composers' directories
22 tunes moved out of /GAMES/ to their composers' directories
```

XSidPlay 2.1.7

Released on 20 September XSidPlay2:

- Use libsidplayfp 1.8.1 (support 3SID)
- Better QT5 porting
- Fix missing initialization in emu configuration
- Add compilation for win32

https://sourceforge.net/projects/xsidplay2/files/xsidplay2/2.1.7

Project Sidologie

Project Sidologie by Marcel Donné - this luxury box set features Commodore 64 music remakes with classic JARRE and Vangelis soundscapes.

https://www.kickstarter.com/projects/c64audio/project-sidologie-jarre-style-commodore-64-music-r/description

Tel Me More

Jeroen Tel (Maniacs of Noise) remakes of his most memorable Commodore 64 video game soundtracks.

The cd (and digital) album will include the following titles:

- Robocop 3
- Cybernoid II The Revenge
- Rubicon
- Hawkeye
- Myth
- Turbo Outrun
- Supremacy
- Stormlord
- Battle Valley
- Cybernoid
- Eliminator
- Iron Lord

https://www.indiegogo.com/projects/jeroen-tel-tel-me-more-c64-game-music-remakes#/

Back in Time Symphonic Collection

A C64 symphonic music campaign in 8 bits: 3 main albums, 5 stretch albums. The first step to the Albert Hall with the LSO? We hope so.

https://www.kickstarter.com/projects/c64audio/back-in-time-symphonic-collection-c64-symphonic-bo

Matt Gray interview!

by Stefano Tognon

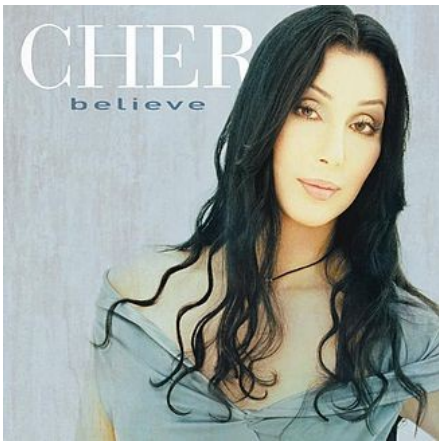
Even if Matt Gray is full working on Reformation Project, we took him some times for more or less technical questions about his Sid activities.

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

I'm a producer and composer for both games and pop. I've been doing this since pretty much full time since 1987. I live and work in the UK.

In the last 20 years there was only one question that SID fans ask in every place: where is Matt Gray?
So, what did you do in those years and when did you realize you are a SID legend with many and many fans in the world?

I stopped doing games music as the C64 market started to slow right down. I was working freelance and the work just wasn't there and for whatever reason I didn't bother to go over to the Amiga. I was more interested in the emerging sampler and affordable synth revolution and I wanted to do dance music. I was A DJ and put out plenty of white label dance tracks in 1990 until about 1993 which sold ok, a decent living at the time.



I then co-formed Motiv 8 which went on to great remix success in the mid 90's and then in '96 I joined the newly formed Xenomania and our first release which I co-produced was All I Wanna Do for Dannii Minougue which entered the charts at 4. Within a few years we went on to major success with tracks such as Believe for Cher and the Christmas number 1 Sound Of The Underground for Girls Aloud.
I worked as part of Xenomania on and off over an 18 year period but slowly over the last few years all the original members apart from Brian have left to do other things and I left last year to do my Reformation album once I realised what a following I seemed to have from SID days. I also wanted to pick up on the games music arena again. I'm already working on several games tracks for release later this year which is very exciting.

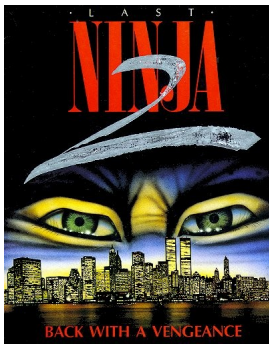
You had worked for many game company like Codemasters, System 3, Hewson, Thalamus, Silverbird, Incentive and others in the Commodore 64 era, so what did you remember about how was working for those company? I read recently that you not always were payed for the work you did, so there some anecdotal to tell about each company I think.

I spent a long time of my fairly short SID career with System 3 which was great to work with such a talented team there. I also had a good relationship with Codemasters for several years, David and Richard are such great guys and also Paul Cooper at Thalamus was great to work with.

SYSTEM·3

Memorable moments were Stavros Fasoulas and Paul coming to show me Quedex on my C64 in my bedroom at my parents house in 87. Getting the call up from Mark Cale at System 3 to not only do the Bangkok Knights loader instead of my hero Rob Hubbard, but also to get the in house job and do Last Ninja 2. Great days.

Last Ninja 2 soundtrack is considerate a true masterpeace (I have an hundred of different remix created by fans about the only LN2 musics). What make it so valuable and popular in your opinion?



I'm not sure I know the real reason.
But I do know that I just did what felt right to me and System 3 gave me a lot free reign to do so. I think it just struck the right balance between what they wanted and what I thought worked.
If there was a formula I'd have bottled it by now.

The Reformation Kickstarter campaign was very successfully with all stretch goals reached (so 75K and more than 1300 backers) with the last three days very emotional for the speed up rush. Can you describe your impression over those 30 intense days?

Well the opening day was one of the most surreal days I can remember, even more so than waiting to hear chart positions. My phone was just lighting up all day as the full impact of the support I was getting for the project from so many fans all over the world hit home. That was a super cool day and I was quite humbled by the sheer number of fans I still had. Other highlights was making the Sanxion loader remake in just a few days and unleashing that.
The last few days were also very exciting and as I had been warned by other Kickstarter project owners, there was a bit of a down period when the campaign finished. Then the really hard work started. But overall I'm just so pleased that so many fans wanted to see this happen.
It makes the production so much more exciting knowing it's wanted.

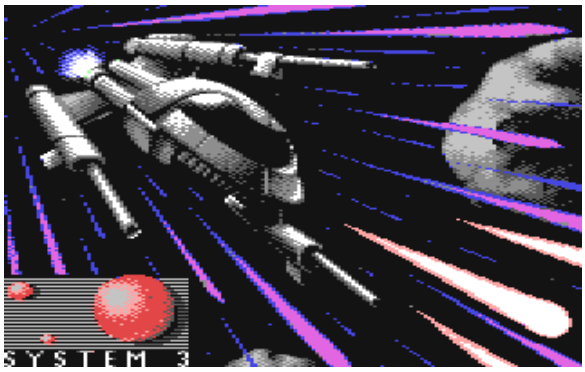
I'm sure some fans will discover your Reformation project in the next months and will be sad for missing it. So what they can still do now? What they can have at the release of the project respect the material that a backer had acquire?

Well anyone can still get exactly the same rewards at the same cost levels simply by following the links on the Kickstarter page which is still there. They just won't be listed on Kickstarter's site as a backer and they will receive updates via email rather than the site.

[Editorial note: here the link http://www.c64audio.com/reformation.html]

For the Reformation project you had released the source of your player used into Dominator game and make a music competition onto it. Can you give more details about this competition for our readers?

Well it's probably finished for entries by now, but I thought it would be interesting to see what people made of my routine rather than just myself. I think there are so many other players and trackers out there now that it didn't appeal to enough people in the end. There haven't really been many entries so far, but what there has been has been good listening.



Speaking about your own player code, Martin Galway many years ago told me that he was forced to rewrite his original engine to fit in some memory restriction imposed by the programmer of a game. What about you? Have you completely freedom about your player or did you have to make compromises of some way?

I always had to fit in with the games programmers. Luckily my routine didn't take much memory or raster time, no more than any of my rivals anyway. My first tracks were re coded which was a waste of time because they left out the mod routines so it sounded nothing like my original really. But that forced me to learn 6502 and code my own personal player. It took about 4-6 months to do that and then I refined it over the next 18 months as I went.

Did you remember if you base your player to others existent one, or it was created all from zero?

It was created from scratch, but there were fundamental things that mine and say Rob Hubbards also did. I did borrow his idea of drum tables but it was always coded from scratch not cut and pasted. My vibrato routine was very different to his and Martin Galway's.

This is just a curiosity. Did you remember where did you take (from other players, from books, handle calculated..) the frequencies value to put in SID registers for the notes? It seems you choosed a base A4 note as 424Hz, that becomes the standard 440Hz onto NTSC system instead of PAL system where the music was expected to be played. I always wanderer if that choice was a way to have a different characteristic sound or not.

That was just an accident. I was sure I got the frequencies from the back of the C64 Reference manual. I only realized this once I started to remake them. So I've had to tune them up by 30%. Some of Galway's tracks were the same, Green Beret for instance was at 424hz. But I prefer 440 because it's much easier for live over dubs. You have to be clever with the retuning though because using sound shifter or melodyne introduces some horrible frequencies to SID sounds. I convert the sounds to samples and then retune on the fine tune and then put them back at the original tempo. It seems to work about 90% of the time.

In your player you had not inserted a single line of code for use SID filters in a tune (unlike Martin Galways where his player uses lot of filter features). What was the motivation behind this?

I only started using them post Dominator on Vendetta and other tracks. Up until then I was well aware of the filter inconsistencies and wanted to avoid using them, but eventually I needed the extra palette of sounds.

Did you come up with the ADSR bug when composing with the SID? Did you apply some thick in the player to avoid it, or did you just going in trial an errors before getting the right values?

I was aware of it,but just kind of lived with it.

Serpent Demo is one of my best demo tune ever, as the drums are top quality. As it was made with your player, how did you add sample in it? [Serpent pictures is by Robin Levy].

Thanks, It was my first attempt at using samples and unfortunately the levels on the drums are too loud, but it was done using tables to change the volume setting very quickly to produce the sound. It was 4 bit sampling using just 16 values so real low level, but it sounded not bad as you say.

Your Dominator player had lot of features in instruments or in pattern commands like slide (and portamento), vibrato, variable pulse wave modulation, table based note/waveform possibility (for bass/drum like stuff) and arpeggio table. There were some other features you would like to have at that time but for any reason you had not coded?

The filter routines came later as did the sample routine, but I didn't get to use them too much in the end.

Have you looked at the actual (most used) programs for making SID music like Goatracker, SID wizard, JCH, DMC, SIDduzz it (just to name a few)? If so, what did you think about the tools you used for making SID music (like your player or SoundMonitor, Electro Sound, Rockmonitor and MusicMaster) compared to those?

The trackers out there now would have been mana from heaven to me back in the day. I started on Electro Sound and then moved on to Soundmonitor and Rockmonitor which were both excellent. But to get your own sound you need to write your own routines I feel.

What is the best features of the SID chip and the missing one in your opinion?

The SID's best feature is it's unique-ness. Nothing else sound exactly like it. Today's emulators are close but not exactly the same. The missing feature was another half dozen channels. That would have been awesome.

Now some quick final (standard) questions:

6581 vs 8580 chip: any (musical) preference?

Not really. It's what people do with what they have at their disposal.

What is the worst and the better sid you composed?

The worst is probably the Mean Streak or Yogi Bear tracks. The best is probably one of the Central Park tracks for Last Ninja 2. They were what I'd call "in the zone" tracks to produce and compose. They came so easily it didn't feel like work at all.

Who are your best sid authors?

Rob Hubbard and Martin Galway in almost equal amount. Martin's lead melodies were so flawless and Rob had a huge range of influence to draw from. He was very good and taking a theme or idea and developing it into something unique.

What are the best sids ever in your opinion?

Sanxion Loader Thalamusik is pretty much flawless and so innovative at the time and I love many of Rob's other tracks such as One Man & His Droid, Phantom Of The Asteroids, WAR, Knuckle-busters. Too many to mention really. Martin's best racks for me were Rambo First Blood Part 2 Loader and Green Beret title music. I also thought Wizball and Parallax were amazing as well.

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

Just that it's so great to see the level of interest in the SID still today and I wanted to say a huge thanks to all SID fans for perpetuating the interest with such genuine fondness. Looking back I was incredibly fortunate to go from a bedroom wannabe to a recognized C64 SID games musician within 18 months and doing a job that just a handful of people in Europe were actually getting paid for. That was very cool and I'm very grateful to leave that as a legacy. But I am working back in the games industry and hopefully there will be enough games work to make it viable long term. Time will tell.

[Matt Gray è su FB: https://www.facebook.com/MattGrayC64/]

Inside Matt Gray Serpent Demo player

by Stefano Tognon <ice00@libero.it>

In the interview you had read in this number with Matt Gray, we know that he uses samples in his engine in some tunes but the sound was considerate too lought. Have you wondered how this was achieved?

Well the answer is inside this analysis.

Sample

The Serpent Demo tune was spread as a demo with a cobra serpent images and in the music you can heart the samples mixed with Matt typical drums. Serpent Demo uses an engine that is younger of the one in Dominator, and it is about the same of Driller one.

So, the part that manages SID tracks, patterns, commands and instruments are the same of Driller engine. What is new is the sample management.

The first point for having the sample was to add a new *VOICE4* to the list of SID voices in the tune definitions.

So we have the table with *VOICE1*, *VOICE2*, *VOICE3* and *VOICE4* pointers for each song present into the program.

The contents of this tables is so all the (special) patterns to play as each commands is now different form the SID commands used in the other voices.

The possible values are:

Command	***Description***
NN	sample speed
$FD VV	sample length duration
$FA BI	Bank (B=0|1), Index of Sample (I)

For understanding those values we have to look at how the sample are played.

First, the logic still goes governed by the *IRQ* (called once a frame), then over a *CIA timer NMI* it is called the sample routine that manipulated the volume register of the SID.

The speed of how often the *NMI* is triggered is governed by the pattern command "*NN*". That values goes directly inside the *$DD04 Timer A Low-Byte* (the high part is fixed at 1). More little is this value and more often the NMI is triggered into a frame.

You can see this as the set of "*note frequency*" to play (or the transpose of all the sample notes being played).

Instead with *$FD* command you set the sample length duration (*VV*) that are to be played. You can so see this as the "*note duration*" to play.

The last command is *$FA* and it is used to set what sample to play. The *BI* value is a compacted

form of two characteristics: the low part is an index to a sample to play, while the high part is a bank switch (0|1) that selects from two possible tables.

Standing from this you have 16x2 max samples area to play, even if using bank 0 or bank 1 gives different behavior as during the play of bank 0, the volume is set to middle before the sample playback.

At this point what you need is to know how to insert the sample. The best approach is to put the sample data starting from memory boundary low address of 00 as the ending of area is performed by comparison to the given high value of memories.

For example: at address $1100 you put the sample and suppose it reaches $12FF as extension. Then you have that starting high area of sample is $11 and ending area is $13.

Those are put inside those tables:

```
BANKLO  .BYTE $00, $00, $00, $00, $FF, $00, $00                                 ; low bank mempoint
BANKHI0 .BYTE SM10\256, SM14\256, SM16\256, SM18\256, SM13\256, SM12\256, SM1C\256 ; high bank 0 mempoint
BANKEND .BYTE SM13\256, SM16\256, SM18\256, SM1A\256, SM14\256, SM13\256, SM1E\256 ; end sample high address
BANKHI1 .BYTE SM12\256, SM15\256, SM17\256, SM19\256, SM19\256, SM12\256, SM1D\256 ; high bank 1 mempoint
```

Each sample of 4 bit is compacted using one byte, so the high and low nibble of one byte have inside two samples to play in sequences.

Looking at the source you will see that there is a called *EMPTY* tunes other that the one played into the demo. This tune did not play sound in SID voices, instead it play only the pattern T16 as sample (maybe a way to test only the sample engine).

With further analysis, into the source there are some patterns not declared (*T2912* and *T28A9*) and 3 tracks not used: *T27DB*, *T2836* and *T285F*. Else, there are many patterns declared and not used. With a simple search *T27DB*, *T2836* and *T285F* are from Hunter's Moon tune (Serpent Demo was spread across the Hunter's Moon game).

As an example of sample pattern we look at this:

```
T29 .BYTE $FD, $00 ; sample length duration
    .BYTE $FA, $03 ; Bank (B=0|1), Index of Sample (I)
    .BYTE $01, $02, $03, $04, $05, $06, $07, $08
    .BYTE $09, $0A, $0B, $0C, $0D, $0E, $0F, $10
    .BYTE $11, $12, $13, $14, $15, $16, $17, $18
    .BYTE $19, $1A, $1B, $1C, $1D, $1E, $1F, $20
    .BYTE $21, $22, $23, $24, $25, $26, $27, $28
    .BYTE $29, $2A, $2B, $2C, $2D, $2E, $2F, $30
    .BYTE $31, $32, $33, $34, $35, $36, $37, $38
    .BYTE $39, $3A, $3B, $3C, $3D, $3E, $3F, $40
    .BYTE $FF
```

This pattern is the first played at the beginning. It uses a fast sample duration ($00) and use sample at index 3 of bank 1 (so the one that goes from $1800 to $19FF). The sample is played in sequences with a fasting increasing frequency (from $01 to $40) that you can easy heart in the drums.

Source

The source code presented here is Copyright by Matt Gray. It was obtained by reverse engineering the Serpent Demo tune and applying all the notations of Dominator engine when possible and all the other are inserted based on the minings of the peace of code.

```
;-------------------------------------------------------------------------------
; BASIC HEADER (WILL AUTOSTART FILE WHEN DROPPED INTO VICE)
; THE "SETIRQ" LINE REFERS TO A LABEL FURTHER DOWN THE CODE

        *= $0801
        .word (+), 2005
        .null $9e, ^STARTUP
+       .word 0

;-------------------------------------------------------------------------------

                ;PLAYER <V4.2
                ;(C)1987
                ;MATT GRAY
                ;This work is licensed
                ;under a Creative Commons
                ;Attribution-NonCommercial 4.0
                ;International License
STARTADD        =$1000          ; starting address of player
C0              =1
CS0             =2
D0              =3
DS0             =4
E0              =5
F0              =6
FS0             =7
G0              =8
GS0             =9
A0              =10
AS0             =11
B0              =12
C1              =13
CS1             =14
D1              =15
DS1             =16
E1              =17
F1              =18
FS1             =19
G1              =20
GS1             =21
A1              =22
AS1             =23
B1              =24
C2              =25
CS2             =26
D2              =27
DS2             =28
E2              =29
F2              =30
FS2             =31
G2              =32
GS2             =33
A2              =34
AS2             =35
B2              =36
C3              =37
CS3             =38
D3              =39
DS3             =40
E3              =41
F3              =42
FS3             =43
G3              =44
GS3             =45
A3              =46
AS3             =47
B3              =48
C4              =49
CS4             =50
D4              =51
DS4             =52
E4              =53
F4              =54
FS4             =55
G4              =56
GS4             =57
A4              =58
AS4             =59
B4              =60
C5              =61
CS5             =62
D5              =63
DS5             =64
E5              =65
F5              =66
FS5             =67
G5              =68
GS5             =69
```

```
A5              =70
AS5             =71
B5              =72
C6              =73
CS6             =74
D6              =75
DS6             =76
E6              =77
F6              =78
FS6             =79
G6              =80
GS6             =81
A6              =82
AS6             =83
B6              =84
C7              =85
CS7             =86
D7              =87
DS7             =88
E7              =89
F7              =90
FS7             =91
G7              =92
GS7             =93
A7              =94
AS7             =95
B7              =96
POINTS          =$FB                    ; track pattern pointer
BARS            =$FD                    ; pattern pointer
V2LO            =V1LO+7
V2HI            =V1HI+7
V3LO            =V1LO+14
V3HI            =V1HI+14
                *=STARTADD


SM10             .BYTE $DD, $DD, $DC, $CB, $BA, $A9, $98, $76
                 .BYTE $65, $43, $22, $10, $00, $00, $01, $12
                 .BYTE $23, $33, $44, $56, $78, $9A, $BB, $CC
                 .BYTE $CD, $DD, $DD, $DC, $DD, $DC, $CB, $BB
                 .BYTE $BB, $AA, $99, $99, $99, $87, $78, $88
                 .BYTE $87, $56, $68, $87, $65, $56, $78, $76
                 .BYTE $44, $67, $87, $64, $45, $79, $86, $44
                 .BYTE $68, $98, $64, $46, $8A, $97, $44, $68
                 .BYTE $BA, $84, $47, $9B, $A8, $54, $79, $BB
                 .BYTE $85, $46, $9B, $B8, $54, $69, $BB, $85
                 .BYTE $46, $9C, $C8, $54, $69, $CC, $85, $46
                 .BYTE $9C, $C8, $43, $68, $CC, $84, $36, $9C
                 .BYTE $C8, $43, $59, $DC, $84, $35, $9D, $D8
                 .BYTE $43, $59, $DD, $84, $25, $9D, $D8, $32
                 .BYTE $59, $DD, $84, $25, $9D, $D8, $32, $59
                 .BYTE $DD, $83, $25, $9D, $D8, $22, $59, $DD
                 .BYTE $72, $25, $9E, $D8, $22, $69, $ED, $72
                 .BYTE $26, $AE, $C6, $12, $69, $FC, $61, $37
                 .BYTE $AF, $C5, $24, $7A, $EB, $63, $56, $9C
                 .BYTE $A7, $88, $54, $78, $9A, $97, $79, $75
                 .BYTE $57, $89, $99, $89, $74, $36, $AB, $A8
                 .BYTE $66, $78, $57, $98, $78, $89, $A8, $23
                 .BYTE $8B, $BA, $86, $78, $65, $9B, $76, $78
                 .BYTE $AA, $61, $4A, $AA, $97, $67, $74, $7B
                 .BYTE $A6, $67, $8B, $B5, $26, $A9, $9A, $87
                 .BYTE $75, $38, $D9, $67, $78, $BA, $34, $A9
                 .BYTE $79, $A8, $86, $23, $BE, $86, $66, $8B
                 .BYTE $82, $6A, $87, $AA, $88, $40, $5E, $D8
                 .BYTE $66, $69, $A4, $4A, $A6, $7A, $A9, $72
                 .BYTE $09, $FB, $77, $66, $97, $38, $D8, $48
                 .BYTE $AB, $A5, $03, $DD, $98, $76, $77, $47
                 .BYTE $CC, $54, $8A, $C9, $10, $7D, $B9, $98
                 .BYTE $67, $33, $AE, $94, $58, $BB, $60, $3A
                 .BYTE $B9, $99, $87, $52, $4C, $E9, $56, $9A
                 .BYTE $A4, $27, $CA, $78, $9A, $A6, $03, $BC
                 .BYTE $87, $88, $A9, $22, $8C, $96, $79, $AA
                 .BYTE $50, $5C, $B7, $88, $9A, $60, $4C, $C7
                 .BYTE $78, $9B, $82, $29, $C9, $79, $9A, $92
                 .BYTE $18, $DA, $67, $89, $A4, $17, $C9, $79
                 .BYTE $99, $94, $16, $DB, $78, $78, $97, $25
                 .BYTE $BA, $78, $99, $96, $24, $AB, $89, $98
                 .BYTE $86, $46, $A9, $78, $99, $87, $45, $89
                 .BYTE $89, $A8, $65, $68, $98, $78, $98, $65
                 .BYTE $79, $87, $7A, $A8, $54, $8A, $87, $8A
                 .BYTE $85, $46, $AC, $86, $88, $75, $58, $A9
                 .BYTE $77, $99, $64, $69, $9A, $98, $75, $47
                 .BYTE $99, $9A, $97, $55, $79, $86, $7A, $A7
                 .BYTE $35, $99, $88, $9B, $83, $37, $BA, $88
                 .BYTE $88, $54, $7A, $98, $88, $86, $57, $88
                 .BYTE $99, $A9, $53, $6A, $A9, $99, $95, $14
                 .BYTE $BD, $97, $8A, $73, $27, $CB, $98, $B9
                 .BYTE $30, $49, $CC, $A9, $74, $14, $9C, $BA
```

```
                .BYTE $A8, $53, $37, $99, $AC, $C7, $21, $59
                .BYTE $AA, $BD, $A3, $03, $9B, $A9, $AB, $72
                .BYTE $27, $AA, $99, $B9, $41, $38, $CB, $AB
                .BYTE $85, $34, $7A, $B9, $98, $64, $57, $89
                .BYTE $AA, $A7, $43, $68, $AA, $AB, $84, $24
                .BYTE $9A, $A9, $A9, $63, $37, $9A, $9A, $B8
                .BYTE $43, $48, $BA, $9A, $94, $24, $8B, $B9
                .BYTE $99, $63, $36, $9B, $AA, $A8, $43, $48
                .BYTE $BA, $AA, $85, $34, $7A, $BA, $A9, $63
                .BYTE $36, $9A, $AA, $A8, $53, $58, $99, $AB
                .BYTE $95, $34, $79, $A9, $BA, $63, $47, $9A
                .BYTE $99, $A8, $54, $68, $88, $AA, $96, $45

SM12            .BYTE $89, $AA, $A9, $54, $58, $99, $99, $97
                .BYTE $55, $78, $88, $9A, $96, $45, $78, $9A
                .BYTE $B9, $64, $58, $A9, $99, $97, $55, $78
                .BYTE $88, $9A, $95, $56, $78, $9A, $A8, $55
                .BYTE $68, $98, $9A, $96, $56, $78, $89, $AA
                .BYTE $84, $46, $89, $9A, $A8, $54, $68, $98
                .BYTE $89, $87, $66, $78, $89, $AA, $75, $56
                .BYTE $79, $AA, $97, $45, $78, $99, $99, $85
                .BYTE $67, $88, $89, $AA, $74, $46, $89, $AA
                .BYTE $A6, $44, $69, $AA, $88, $75, $57, $78
                .BYTE $99, $99, $75, $56, $89, $AA, $96, $45
                .BYTE $79, $A9, $88, $65, $67, $88, $98, $99
                .BYTE $75, $56, $8A, $BA, $85, $45, $7A, $BA
                .BYTE $97, $55, $67, $89, $99, $98, $55, $57
                .BYTE $9A, $BA, $85, $35, $7A, $B9, $87, $65
                .BYTE $57, $89, $98, $98, $65, $57, $AB, $A9
                .BYTE $75, $45, $8A, $B9, $87, $55, $67, $9A
                .BYTE $98, $87, $55, $68, $AA, $A9, $64, $46
                .BYTE $9B, $B9, $87, $54, $57, $AB, $99, $85
                .BYTE $45, $69, $BA, $A9, $64, $46, $9A, $AA
                .BYTE $97, $44, $58, $AA, $A9, $85, $45, $79
                .BYTE $A9, $98, $65, $56, $89, $A9, $97, $54
                .BYTE $68, $AA, $99, $75, $56, $8A, $A9, $87
                .BYTE $65, $67, $9A, $99, $86, $55, $79, $BA
                .BYTE $98, $64, $56, $9A, $A9, $86, $55, $68
                .BYTE $9A, $A9, $75, $45, $79, $BB, $A8, $54
                .BYTE $57, $9A, $A9, $86, $54, $68, $AA, $9A
                .BYTE $85, $45, $79, $BA, $97, $65, $56, $9A
                .BYTE $A9, $86, $55, $68, $AA, $A9, $65, $56
                .BYTE $79, $AA, $97, $55, $67, $9A, $A9, $76
                .BYTE $56, $79, $AA, $98, $65, $56, $8A, $A9
                .BYTE $96, $55, $57, $AB, $A9, $65, $56, $79

SM13            .BYTE $AA, $97, $65, $66, $89, $BA, $96, $55
                .BYTE $68, $9A, $A9, $75, $56, $89, $A9, $98
                .BYTE $65, $57, $9A, $A9, $87, $65, $68, $AB
                .BYTE $A8, $65, $66, $89, $AA, $96, $56, $67
                .BYTE $8A, $AA, $75, $46, $78, $AA, $A8, $64
                .BYTE $57, $9A, $A9, $87, $55, $58, $AA, $99
                .BYTE $76, $55, $68, $BB, $97, $65, $67, $8A
                .BYTE $AA, $86, $56, $78, $99, $A9, $75, $56
                .BYTE $79, $AA, $97, $55, $67, $9A, $A9, $76
                .BYTE $56, $78, $99, $98, $76, $66, $79, $A9
                .BYTE $97, $55, $67, $9A, $A9, $76, $66, $78
                .BYTE $99, $98, $66, $66, $8A, $A9, $86, $56
                .BYTE $78, $9A, $98, $65, $67, $89, $9A, $97
                .BYTE $65, $67, $9A, $AA, $86, $45, $79, $A9
                .BYTE $98, $75, $56, $89, $A9, $98, $75, $57
                .BYTE $9A, $A9, $76, $55, $69, $A9, $98, $65
                .BYTE $09, $09, $09, $09, $09, $09, $09, $09
                .BYTE $09, $09, $09, $09, $09, $09, $09, $09
                .BYTE $09, $09, $09, $09, $09, $09, $09, $09
                .BYTE $09, $09, $09, $09, $09, $09, $09, $09
                .BYTE $09, $09, $09, $09, $09, $09, $09, $09
                .BYTE $09, $09, $09, $09, $09, $09, $09, $09
                .BYTE $09, $09, $09, $09, $09, $09, $09, $09
                .BYTE $09, $09, $09, $09, $09, $09, $09, $09
                .BYTE $09, $09, $09, $09, $09, $09, $09, $09
                .BYTE $09, $09, $09, $09, $09, $09, $09, $09
                .BYTE $09, $09, $09, $09, $09, $09, $09, $09
                .BYTE $09, $09, $09, $09, $09, $09, $09, $09
                .BYTE $09, $09, $09, $09, $09, $09, $09, $09
                .BYTE $09, $09, $09, $09, $09, $09, $09, $09
                .BYTE $09, $09, $09, $09, $09, $09, $09, $09
                .BYTE $09, $09, $09, $09, $09, $09, $09, $78

SM14            .BYTE $52, $81, $06, $B7, $BB, $73, $60, $25
                .BYTE $3A, $FF, $D7, $A9, $83, $64, $24, $7C
                .BYTE $9B, $BA, $EE, $A8, $69, $75, $11, $35
                .BYTE $67, $BE, $EC, $CB, $9B, $76, $32, $33
                .BYTE $35, $46, $78, $BD, $FF, $FD, $B9, $85
                .BYTE $44, $44, $43, $22, $24, $79, $AB, $CC
                .BYTE $DE, $ED, $CA, $86, $43, $32, $33, $44
                .BYTE $45, $56, $78, $AB, $DE, $ED, $CB, $A9
                .BYTE $98, $75, $43, $22, $23, $45, $67, $89
```

```
                .BYTE $9A, $AB, $BC, $CC, $CC, $B9, $86, $54
                .BYTE $44, $44, $44, $44, $45, $68, $9A, $BB
                .BYTE $CC, $CB, $BB, $AA, $98, $77, $65, $44
                .BYTE $33, $34, $45, $56, $67, $89, $9A, $BC
                .BYTE $CC, $CC, $BA, $A9, $87, $76, $65, $55
                .BYTE $44, $44, $45, $55, $66, $78, $9A, $AB
                .BYTE $BB, $BB, $BB, $BA, $A9, $87, $66, $55
                .BYTE $55, $55, $55, $55, $56, $66, $77, $89
                .BYTE $9A, $BB, $BB, $BB, $AA, $99, $88, $77
                .BYTE $66, $55, $55, $44, $55, $56, $67, $77
                .BYTE $78, $89, $9A, $AA, $AA, $BB, $BA, $AA
                .BYTE $98, $87, $76, $65, $55, $55, $55, $66
                .BYTE $66, $66, $77, $78, $89, $99, $AA, $AA
                .BYTE $AA, $AA, $99, $99, $88, $77, $66, $65
                .BYTE $55, $55, $66, $66, $67, $77, $77, $88
                .BYTE $89, $99, $9A, $AA, $AA, $A9, $99, $98
                .BYTE $88, $77, $76, $66, $66, $66, $66, $66
                .BYTE $66, $67, $77, $78, $88, $99, $99, $99
                .BYTE $99, $99, $99, $99, $98, $88, $77, $76
                .BYTE $66, $66, $66, $66, $66, $77, $77, $77
                .BYTE $78, $88, $88, $89, $99, $99, $99, $99
                .BYTE $99, $88, $88, $87, $77, $77, $66, $66
                .BYTE $66, $66, $67, $77, $77, $77, $88, $88

SM15            .BYTE $88, $88, $99, $99, $99, $99, $99, $88
                .BYTE $88, $88, $77, $77, $77, $66, $66, $66
                .BYTE $67, $77, $77, $77, $77, $88, $88, $88
                .BYTE $88, $99, $99, $99, $99, $98, $88, $88
                .BYTE $88, $77, $77, $77, $77, $66, $77, $77
                .BYTE $77, $77, $77, $77, $88, $88, $88, $88
                .BYTE $88, $89, $99, $99, $88, $88, $88, $88
                .BYTE $88, $77, $77, $77, $77, $77, $77, $77
                .BYTE $77, $77, $77, $77, $88, $88, $88, $88
                .BYTE $88, $88, $88, $88, $88, $88, $88, $88
                .BYTE $88, $77, $77, $77, $77, $77, $77, $77
                .BYTE $77, $77, $77, $77, $78, $88, $88, $88
                .BYTE $88, $88, $88, $88, $88, $88, $88, $88
                .BYTE $88, $88, $77, $77, $77, $77, $77, $77
                .BYTE $77, $77, $77, $77, $78, $88, $88, $88
                .BYTE $88, $88, $88, $88, $88, $88, $88, $88
                .BYTE $88, $88, $88, $87, $77, $77, $77, $77
                .BYTE $77, $77, $77, $77, $77, $77, $88, $88
                .BYTE $88, $88, $88, $88, $88, $88, $88, $88
                .BYTE $88, $88, $88, $88, $88, $88, $77, $77
                .BYTE $77, $77, $77, $77, $77, $77, $77, $88
                .BYTE $88, $88, $88, $88, $88, $88, $88, $88
                .BYTE $88, $88, $88, $88, $88, $88, $88, $77
                .BYTE $77, $77, $77, $77, $77, $77, $77, $77
                .BYTE $77, $88, $88, $88, $88, $88, $88, $88
                .BYTE $88, $88, $88, $88, $88, $88, $88, $88
                .BYTE $88, $77, $77, $77, $77, $77, $77, $77
                .BYTE $77, $77, $77, $78, $88, $88, $88, $88
                .BYTE $88, $88, $88, $88, $88, $88, $88, $88
                .BYTE $88, $88, $88, $88, $77, $77, $77, $77
                .BYTE $77, $77, $77, $77, $77, $78, $88, $88
                .BYTE $88, $88, $88, $88, $88, $88, $88, $88

SM16            .BYTE $80, $0A, $47, $14, $4C, $4F, $BA, $8E
                .BYTE $F8, $EF, $00, $2A, $91, $BA, $EF, $AF
                .BYTE $0F, $88, $DD, $2F, $D0, $08, $10, $DF
                .BYTE $F1, $46, $2A, $20, $88, $FE, $58, $FF
                .BYTE $D8, $FB, $AC, $FF, $35, $30, $D4, $53
                .BYTE $31, $4F, $5F, $03, $68, $21, $01, $BB
                .BYTE $10, $9B, $48, $98, $8F, $FD, $FE, $EC
                .BYTE $B2, $CA, $23, $30, $03, $09, $B1, $8C
                .BYTE $E5, $8E, $8F, $70, $10, $57, $77, $1B
                .BYTE $56, $B4, $CC, $F6, $36, $F7, $D7, $B4
                .BYTE $C4, $CC, $B7, $AC, $FD, $64, $59, $52
                .BYTE $30, $10, $44, $61, $01, $05, $66, $96
                .BYTE $87, $BD, $EC, $99, $C8, $69, $A5, $87
                .BYTE $68, $08, $C9, $98, $BB, $77, $64, $03
                .BYTE $34, $5B, $89, $58, $58, $EB, $7B, $36
                .BYTE $BB, $37, $75, $D9, $B8, $9B, $AF, $AA
                .BYTE $C9, $DC, $E9, $9C, $78, $A3, $68, $63
                .BYTE $66, $77, $55, $65, $22, $33, $B7, $45
                .BYTE $95, $85, $A8, $AB, $8C, $DD, $A8, $BA
                .BYTE $8B, $AC, $6A, $53, $65, $50, $33, $64
                .BYTE $14, $76, $46, $36, $58, $43, $47, $A8
                .BYTE $66, $A9, $8B, $BA, $98, $AA, $C7, $76
                .BYTE $69, $98, $68, $67, $96, $35, $44, $67
                .BYTE $55, $77, $79, $76, $87, $27, $55, $43
                .BYTE $66, $76, $88, $79, $AB, $CC, $CC, $CC
                .BYTE $FB, $C8, $98, $B9, $99, $89, $89, $95
                .BYTE $76, $63, $66, $64, $45, $37, $54, $43
                .BYTE $47, $35, $67, $7B, $A8, $89, $AB, $9A
                .BYTE $BC, $A9, $9B, $AA, $A7, $99, $A8, $8B
                .BYTE $99, $77, $78, $98, $56, $57, $54, $76
                .BYTE $54, $66, $57, $99, $97, $99, $98, $9A
```

```
                .BYTE $97, $78, $97, $A8, $78, $AA, $AB, $B9

SM17            .BYTE $B9, $8B, $96, $88, $88, $68, $46, $85
                .BYTE $55, $67, $34, $57, $66, $74, $68, $47
                .BYTE $86, $77, $77, $87, $99, $8B, $BA, $B9
                .BYTE $A9, $BB, $9A, $A9, $86, $67, $77, $77
                .BYTE $66, $57, $65, $77, $78, $67, $67, $68
                .BYTE $87, $97, $78, $78, $88, $78, $A8, $88
                .BYTE $79, $98, $99, $A9, $98, $9A, $97, $99
                .BYTE $89, $88, $76, $67, $77, $65, $78, $77
                .BYTE $77, $87, $67, $77, $76, $57, $67, $66
                .BYTE $77, $77, $89, $99, $98, $99, $8A, $88
                .BYTE $98, $77, $68, $78, $77, $78, $88, $87
                .BYTE $98, $77, $57, $66, $77, $76, $66, $55
                .BYTE $66, $77, $87, $77, $78, $98, $89, $A8
                .BYTE $8B, $99, $88, $87, $97, $89, $88, $88
                .BYTE $AA, $A9, $99, $78, $68, $78, $67, $77
                .BYTE $66, $66, $77, $78, $87, $87, $78, $99
                .BYTE $9A, $99, $99, $78, $88, $88, $78, $89
                .BYTE $98, $98, $88, $88, $87, $9A, $77, $77
                .BYTE $77, $66, $77, $75, $67, $77, $78, $87
                .BYTE $77, $77, $87, $87, $77, $78, $88, $88
                .BYTE $89, $89, $8A, $99, $88, $87, $99, $89
                .BYTE $98, $78, $88, $77, $78, $87, $77, $76
                .BYTE $67, $76, $77, $78, $88, $78, $87, $77
                .BYTE $66, $77, $76, $78, $78, $89, $78, $88
                .BYTE $87, $77, $77, $77, $77, $78, $89, $99
                .BYTE $98, $88, $77, $77, $76, $67, $77, $88
                .BYTE $89, $99, $98, $87, $77, $77, $76, $77
                .BYTE $78, $89, $99, $88, $88, $87, $77, $77
                .BYTE $77, $77, $78, $89, $99, $98, $88, $77
                .BYTE $77, $77, $77, $77, $88, $89, $99, $98
                .BYTE $88, $77, $77, $77, $77, $77, $88, $89
                .BYTE $98, $88, $88, $77, $77, $77, $77, $77

SM18            .BYTE $88, $88, $88, $88, $88, $88, $88, $88
                .BYTE $88, $88, $88, $88, $88, $88, $88, $88
                .BYTE $88, $88, $88, $88, $88, $88, $88, $88
                .BYTE $88, $88, $88, $88, $88, $88, $88, $88
                .BYTE $88, $88, $88, $88, $88, $88, $88, $88
                .BYTE $88, $88, $88, $88, $88, $88, $88, $88
                .BYTE $88, $88, $88, $88, $88, $88, $88, $88
                .BYTE $88, $88, $88, $88, $88, $88, $88, $88
                .BYTE $86, $A5, $49, $6B, $8B, $66, $8C, $79
                .BYTE $2A, $63, $38, $75, $8C, $38, $44, $9A
                .BYTE $66, $DD, $7B, $36, $27, $CA, $56, $54
                .BYTE $76, $78, $BC, $55, $98, $6B, $75, $A3
                .BYTE $89, $6A, $A6, $46, $86, $B8, $99, $54
                .BYTE $A8, $58, $96, $25, $69, $B8, $AC, $94
                .BYTE $69, $76, $9B, $88, $45, $89, $97, $AA
                .BYTE $65, $85, $88, $88, $A9, $65, $78, $77
                .BYTE $AA, $89, $64, $79, $86, $8B, $95, $68
                .BYTE $57, $98, $8A, $B8, $55, $77, $67, $AA
                .BYTE $89, $75, $69, $87, $8A, $B9, $66, $76
                .BYTE $68, $88, $AB, $96, $56, $76, $57, $AA
                .BYTE $89, $86, $57, $87, $79, $BA, $86, $67
                .BYTE $65, $78, $89, $BA, $86, $67, $76, $67
                .BYTE $AA, $99, $96, $56, $77, $77, $9B, $A8
                .BYTE $66, $77, $66, $88, $9A, $B9, $76, $67
                .BYTE $66, $57, $AA, $97, $88, $66, $77, $77
                .BYTE $9A, $B9, $76, $77, $65, $67, $99, $9A
                .BYTE $A8, $66, $77, $66, $68, $AA, $98, $88
                .BYTE $65, $67, $76, $89, $BA, $87, $66, $76
                .BYTE $55, $79, $99, $AA, $97, $67, $77, $76
                .BYTE $78, $AA, $98, $78, $76, $67, $77, $78
                .BYTE $AB, $A9, $77, $77, $66, $56, $89, $99
                .BYTE $A9, $87, $67, $77, $66, $78, $AA, $98

SM19            .BYTE $77, $87, $66, $77, $77, $9A, $A9, $87
                .BYTE $77, $77, $66, $67, $99, $99, $99, $76
                .BYTE $67, $76, $66, $78, $AA, $98, $77, $87
                .BYTE $65, $67, $77, $89, $AA, $98, $77, $77
                .BYTE $76, $66, $78, $99, $99, $98, $77, $77
                .BYTE $76, $66, $78, $AA, $99, $87, $78, $76
                .BYTE $66, $77, $78, $9A, $A9, $87, $77, $76
                .BYTE $65, $67, $89, $99, $99, $98, $77, $77
                .BYTE $76, $67, $79, $9A, $98, $88, $87, $76
                .BYTE $66, $77, $78, $9A, $A9, $87, $77, $77
                .BYTE $66, $66, $78, $99, $99, $98, $87, $77
                .BYTE $77, $66, $77, $89, $99, $98, $88, $87
                .BYTE $66, $67, $77, $78, $99, $99, $87, $77
                .BYTE $77, $66, $66, $78, $99, $99, $99, $87
                .BYTE $77, $77, $76, $67, $78, $99, $99, $88
                .BYTE $88, $77, $66, $77, $77, $88, $9A, $99
                .BYTE $87, $77, $77, $76, $66, $78, $89, $99
                .BYTE $99, $88, $77, $77, $77, $67, $78, $89
                .BYTE $99, $98, $88, $87, $76, $67, $77, $78
                .BYTE $89, $99, $98, $87, $77, $77, $66, $67
```

```
                .BYTE $78, $89, $99, $99, $87, $77, $77, $77
                .BYTE $77, $78, $89, $99, $88, $87, $77, $77
                .BYTE $66, $77, $78, $88, $99, $99, $88, $77
                .BYTE $77, $77, $67, $77, $88, $99, $99, $88
                .BYTE $87, $77, $77, $77, $77, $78, $89, $99
                .BYTE $98, $88, $77, $77, $76, $67, $77, $88
                .BYTE $89, $99, $98, $87, $77, $77, $76, $77
                .BYTE $78, $89, $99, $88, $88, $87, $77, $77
                .BYTE $77, $77, $78, $89, $99, $98, $88, $77
                .BYTE $77, $77, $77, $77, $88, $89, $99, $98
                .BYTE $88, $77, $77, $77, $77, $77, $88, $89
                .BYTE $98, $88, $88, $77, $77, $77, $77, $77

SM1A            .BYTE $32, $64, $98, $BB, $DC, $CD, $CC, $AB
                .BYTE $AA, $89, $78, $77, $76, $87, $A9, $CB
                .BYTE $CD, $BC, $9A, $78, $66, $66, $66, $77
                .BYTE $77, $77, $77, $66, $66, $56, $66, $65
                .BYTE $66, $66, $66, $66, $55, $44, $55, $76
                .BYTE $67, $66, $56, $65, $A8, $BB, $BB, $CC
                .BYTE $AC, $68, $13, $10, $42, $87, $BA, $CB
                .BYTE $CC, $BC, $AB, $AA, $AA, $9A, $78, $56
                .BYTE $55, $86, $B9, $DC, $DD, $BC, $89, $77
                .BYTE $77, $77, $88, $88, $67, $56, $55, $65
                .BYTE $66, $66, $66, $56, $65, $66, $77, $77
                .BYTE $67, $55, $44, $44, $44, $54, $66, $76
                .BYTE $A8, $CB, $CC, $CC, $BC, $7A, $24, $01
                .BYTE $21, $64, $A8, $CB, $CC, $CC, $CC, $BC
                .BYTE $CB, $CC, $9B, $67, $34, $43, $76, $B9
                .BYTE $ED, $EE, $BD, $9A, $78, $77, $87, $88
                .BYTE $78, $67, $66, $76, $77, $77, $56, $55
                .BYTE $55, $66, $76, $77, $57, $45, $33, $43
                .BYTE $44, $54, $66, $46, $64, $A8, $BB, $BB
                .BYTE $CB, $AC, $57, $02, $10, $42, $97, $BA
                .BYTE $CC, $CD, $BB, $BB, $BB, $BB, $AB, $89
                .BYTE $67, $66, $87, $B9, $CB, $CD, $AB, $89
                .BYTE $88, $98, $99, $99, $89, $78, $66, $55
                .BYTE $55, $66, $66, $55, $55, $65, $66, $77
                .BYTE $67, $45, $34, $23, $43, $65, $77, $66
                .BYTE $45, $43, $96, $BA, $CC, $DC, $BD, $69
                .BYTE $13, $21, $53, $97, $BA, $CC, $AB, $A9
                .BYTE $BB, $BB, $CB, $CD, $8A, $46, $54, $76
                .BYTE $BA, $ED, $DE, $BC, $89, $88, $88, $99
                .BYTE $99, $88, $67, $55, $66, $87, $88, $78
                .BYTE $56, $34, $54, $76, $88, $67, $45, $34
                .BYTE $23, $53, $66, $67, $56, $34, $32, $96

SM1B            .BYTE $BB, $CC, $DD, $9C, $46, $11, $31, $85
                .BYTE $B9, $CB, $AB, $88, $A9, $BB, $CB, $DD
                .BYTE $AC, $79, $55, $75, $B9, $DC, $DD, $AB
                .BYTE $89, $88, $A9, $BA, $9A, $78, $56, $55
                .BYTE $76, $87, $99, $78, $56, $34, $43, $65
                .BYTE $87, $78, $56, $44, $33, $23, $53, $87
                .BYTE $67, $55, $45, $43, $96, $BA, $CC, $CD
                .BYTE $9B, $58, $34, $43, $76, $88, $77, $88
                .BYTE $99, $BA, $BB, $CC, $CD, $AB, $89, $88
                .BYTE $98, $AA, $BB, $AA, $99, $99, $AA, $AA
                .BYTE $AA, $9A, $78, $77, $77, $88, $88, $77
                .BYTE $56, $45, $55, $65, $66, $66, $56, $45
                .BYTE $54, $65, $56, $44, $34, $54, $66, $55
                .BYTE $45, $54, $96, $CB, $DD, $CD, $9B, $68
                .BYTE $45, $54, $66, $77, $77, $77, $A9, $BB
                .BYTE $CB, $CC, $BC, $9A, $99, $99, $99, $AA
                .BYTE $AA, $9A, $99, $AA, $AA, $AA, $99, $89
                .BYTE $88, $88, $88, $78, $67, $56, $55, $66
                .BYTE $66, $55, $55, $55, $65, $66, $56, $45
                .BYTE $44, $44, $55, $34, $64, $77, $67, $45
                .BYTE $54, $97, $CB, $DC, $CD, $8A, $77, $77
                .BYTE $77, $67, $56, $55, $76, $A8, $BB, $BB
                .BYTE $BB, $BB, $CC, $BC, $BB, $9A, $88, $99
                .BYTE $AA, $AA, $99, $99, $99, $AA, $AA, $89
                .BYTE $78, $67, $66, $77, $67, $66, $55, $55
                .BYTE $55, $55, $55, $55, $55, $55, $55, $45
                .BYTE $34, $54, $56, $55, $76, $77, $77, $67
                .BYTE $55, $76, $B9, $DC, $CD, $AA, $89, $88
                .BYTE $88, $68, $55, $44, $75, $A9, $AA, $AA
                .BYTE $AA, $BA, $DC, $CD, $BC, $99, $98, $A9
                .BYTE $AA, $9A, $88, $98, $A9, $BA, $9A, $78
                .BYTE $77, $77, $77, $67, $56, $55, $55, $56

SM1C            .BYTE $55, $55, $55, $55, $55, $55, $55, $34
                .BYTE $43, $65, $66, $45, $54, $97, $9A, $78
                .BYTE $56, $66, $97, $B9, $CC, $AC, $A9, $BA
                .BYTE $AB, $79, $56, $55, $66, $87, $88, $88
                .BYTE $98, $BA, $CC, $BC, $AB, $BB, $BB, $9A
                .BYTE $99, $88, $88, $99, $99, $89, $88, $99
                .BYTE $99, $78, $67, $66, $66, $66, $66, $55
                .BYTE $55, $55, $55, $55, $54, $55, $55, $65
                .BYTE $55, $44, $54, $76, $67, $56, $64, $97
```

```
                .BYTE $A9, $89, $67, $87, $87, $87, $B9, $DC
                .BYTE $CD, $BB, $BB, $9A, $78, $67, $56, $55
                .BYTE $76, $88, $88, $99, $AA, $AA, $BB, $BB
                .BYTE $BB, $AA, $AA, $99, $99, $88, $88, $88
                .BYTE $88, $88, $99, $88, $77, $77, $67, $66
                .BYTE $56, $55, $55, $55, $55, $45, $44, $55
                .BYTE $55, $55, $55, $55, $66, $66, $55, $65
                .BYTE $66, $66, $98, $9A, $78, $76, $A9, $AB
                .BYTE $68, $75, $CA, $DD, $AB, $A9, $BA, $AB
                .BYTE $89, $66, $66, $87, $88, $67, $76, $A9
                .BYTE $AB, $89, $98, $BB, $BC, $9A, $99, $A9
                .BYTE $9A, $89, $88, $88, $88, $88, $78, $77
                .BYTE $88, $77, $66, $65, $66, $56, $45, $54
                .BYTE $55, $55, $55, $55, $65, $66, $55, $65
                .BYTE $66, $55, $65, $77, $66, $76, $87, $99
                .BYTE $AA, $89, $88, $AA, $9A, $68, $77, $B9
                .BYTE $BC, $AB, $A9, $BB, $BB, $79, $77, $87
                .BYTE $77, $56, $75, $87, $88, $88, $98, $A9
                .BYTE $BA, $AA, $AA, $AA, $AA, $99, $88, $89
                .BYTE $88, $78, $77, $88, $78, $77, $66, $67
                .BYTE $66, $55, $55, $55, $55, $55, $55, $55
                .BYTE $55, $55, $66, $66, $66, $66, $66, $66
                .BYTE $77, $77, $88, $78, $77, $A9, $AA, $89

SM1D            .BYTE $98, $AA, $8A, $77, $98, $99, $A9, $99
                .BYTE $AA, $BB, $AA, $89, $88, $88, $67, $65
                .BYTE $76, $77, $77, $87, $99, $99, $99, $A9
                .BYTE $AA, $AB, $99, $99, $99, $89, $78, $87
                .BYTE $78, $77, $77, $77, $77, $66, $66, $66
                .BYTE $66, $55, $55, $55, $55, $55, $65, $56
                .BYTE $65, $66, $66, $66, $66, $66, $77, $77
                .BYTE $87, $88, $77, $87, $A9, $99, $88, $A9
                .BYTE $99, $89, $98, $99, $89, $88, $A9, $AA
                .BYTE $AA, $99, $9A, $89, $88, $78, $77, $77
                .BYTE $77, $77, $77, $88, $88, $88, $99, $A9
                .BYTE $AA, $9A, $99, $99, $99, $88, $88, $78
                .BYTE $77, $77, $77, $77, $77, $66, $66, $66
                .BYTE $66, $66, $56, $55, $55, $66, $66, $66
                .BYTE $66, $66, $67, $66, $76, $77, $77, $77
                .BYTE $88, $88, $78, $88, $99, $89, $99, $99
                .BYTE $60, $6F, $7F, $B5, $D6, $ED, $FF, $FF
                .BYTE $FF, $FF, $FF, $F2, $E0, $AC, $A0, $7B
                .BYTE $89, $74, $7B, $6C, $40, $4A, $32, $08
                .BYTE $00, $00, $00, $00, $00, $03, $2F, $5F
                .BYTE $AB, $C5, $B8, $BF, $FF, $FC, $DC, $C9
                .BYTE $B4, $70, $41, $38, $00, $13, $2F, $7F
                .BYTE $9F, $D7, $FF, $FF, $FF, $F8, $B0, $B4
                .BYTE $70, $67, $8B, $72, $79, $50, $49, $54
                .BYTE $54, $6F, $85, $80, $55, $4C, $2F, $5F
                .BYTE $6F, $98, $83, $9F, $BF, $B8, $BF, $D1
                .BYTE $DB, $D8, $CB, $C4, $95, $A0, $9F, $B5
                .BYTE $92, $87, $BF, $DF, $FB, $FF, $E8, $A0
                .BYTE $70, $20, $00, $00, $00, $1F, $1C, $19
                .BYTE $1D, $1A, $18, $02, $00, $00, $07, $4B
                .BYTE $4F, $87, $80, $7F, $94, $93, $94, $82
                .BYTE $66, $64, $57, $60, $67, $58, $35, $77
SM1E
                .BYTE $73, $7E, $8A, $93, $9B, $A0, $94, $80
                .BYTE $6C, $68, $73, $7F, $8B, $92, $97, $9C
                .BYTE $95, $88, $72, $66, $6F, $7A, $7D, $7C
                .BYTE $7D, $7E, $80, $7D, $74, $73, $7D, $8B
                .BYTE $92, $91, $97, $9D, $9B, $90, $88, $81
                .BYTE $7C, $76, $6C, $60, $5A, $63, $6F, $75
                .BYTE $7A, $87, $9D, $AF, $B3, $B0, $AC, $A4
                .BYTE $98, $84, $72, $62, $54, $4C, $4A, $4D
                .BYTE $5B, $75, $93, $AF, $C3, $C8, $C4, $A8
                .BYTE $80, $50, $39, $3F, $53, $6B, $87, $A5
                .BYTE $B9, $B5, $A0, $84, $74, $71, $75, $7D
                .BYTE $84, $84, $7A, $6C, $5C, $54, $5A, $6B
                .BYTE $7F, $97, $AF, $BA, $B6, $A2, $80, $60
                .BYTE $50, $53, $65, $75, $7F, $8D, $94, $8C
                .BYTE $78, $68, $67, $7B, $8F, $99, $96, $94
                .BYTE $8D, $80, $6C, $61, $65, $73, $85, $8B
                .BYTE $8D, $8F, $8A, $7A, $66, $60, $6B, $7D
                .BYTE $89, $93, $9F, $A6, $A8, $A4, $96, $86
                .BYTE $7A, $6C, $5C, $54, $55, $5F, $6F, $7D
                .BYTE $8B, $99, $9F, $9A, $90, $87, $84, $81
                .BYTE $74, $6C, $6E, $7B, $80, $7A, $74, $77
                .BYTE $7F, $81, $7E, $82, $8B, $95, $97, $8E
                .BYTE $80, $78, $75, $70, $68, $67, $6E, $74
                .BYTE $7B, $7F, $8D, $96, $9A, $96, $92, $92
                .BYTE $8C, $7A, $64, $5B, $61, $73, $7F, $80
                .BYTE $7E, $85, $8D, $8D, $8B, $88, $87, $87
                .BYTE $84, $76, $69, $64, $6E, $7B, $87, $9A
                .BYTE $A9, $AC, $A0, $90, $80, $6C, $5E, $5C
                .BYTE $5F, $6F, $89, $A7, $B3, $B0, $A0, $92
                .BYTE $80, $75, $6E, $6D, $73, $7A, $78, $76
                .BYTE $7C, $87, $8F, $96, $9A, $96, $90, $86
```

```
                .BYTE $74, $64, $61, $6B, $7A, $8B, $99, $A0

SM1F            .BYTE $9B, $8C, $78, $60, $59, $65, $7B, $8D
                .BYTE $9F, $AB, $AE, $A6, $9A, $88, $70, $62
                .BYTE $62, $6D, $77, $7F, $84, $80, $76, $6E
                .BYTE $6A, $6D, $7B, $8B, $9A, $A9, $AE, $A8
                .BYTE $90, $74, $60, $57, $57, $5F, $6D, $79
                .BYTE $86, $8F, $93, $96, $9C, $9F, $A7, $AB
                .BYTE $A4, $94, $7A, $60, $46, $36, $3B, $47
                .BYTE $5E, $7F, $9F, $B9, $C3, $C2, $B8, $A6
                .BYTE $95, $84, $6A, $56, $54, $58, $54, $4D
                .BYTE $4A, $4B, $53, $65, $7F, $A7, $CB, $E2
                .BYTE $E2, $CC, $AC, $80, $58, $40, $37, $42
                .BYTE $57, $6D, $79, $7F, $80, $81, $89, $97
                .BYTE $A6, $A9, $A6, $A2, $98, $80, $5C, $44
                .BYTE $41, $4B, $5E, $77, $93, $A5, $B3, $B7
                .BYTE $AC, $9A, $8B, $80, $74, $69, $64, $69
                .BYTE $6F, $6D, $69, $6B, $75, $7F, $8D, $9B
                .BYTE $A9, $B3, $B3, $A6, $8A, $6C, $5A, $54
                .BYTE $57, $5F, $6D, $7F, $8F, $9C, $9F, $9D
                .BYTE $98, $8E, $7A, $64, $58, $5F, $6F, $7F
                .BYTE $86, $82, $82, $82, $7C, $78, $7F, $8F
                .BYTE $A2, $A4, $98, $86, $7A, $72, $65, $5B
                .BYTE $5D, $6E, $83, $8F, $90, $8A, $87, $85
                .BYTE $87, $91, $9F, $B2, $B8, $AC, $8C, $68
                .BYTE $50, $44, $41, $47, $5B, $77, $99, $B3
                .BYTE $BA, $B0, $A0, $92, $82, $70, $60, $5F
                .BYTE $67, $71, $75, $71, $6C, $71, $7B, $82
                .BYTE $8A, $9A, $A9, $AC, $A1, $88, $70, $60
                .BYTE $5D, $64, $72, $7F, $96, $A3, $A0, $88
                .BYTE $6C, $5C, $57, $5C, $67, $7B, $95, $AA
                .BYTE $B4, $B0, $A0, $8A, $80, $78, $73, $72
                .BYTE $79, $7F, $81, $7E, $7B, $76, $72, $6D
                .BYTE $75, $87, $9B, $A8, $AF, $AD, $A2, $00


MAIN            LDA TN                  ; song (track) number
                BNE PLAYMUSIC2
                STA $D418               ; Select Filter Mode and Volume (to 0)
                RTS
PLAYMUSIC2
                CMP #$AB                ; current songs ?
                BEQ MUSIC
                JMP SETPOINTS           ; init the tracks

SETCONT         LDA #$00
                STA $D404               ; Voice 1: Control Register
                STA $D40B               ; Voice 2: Control Register
                STA $D412               ; Voice 3: Control Register
                LDA #$0E
                STA $D418               ; Select Filter Mode and Volume

                LDY #$00
                STY SAMPLEBARCOUNT      ; set track position (offset - bar counter of sample) to the beginning
                STY BARCOUNT            ; set track 1 position to the beginning
                STY BARCOUNT+7          ; set track 2 position to the beginning
                STY BARCOUNT+14         ; set track 3 position to the beginning
                STY V1DUR               ; actual note length duration voice 1
                STY V1DUR+7             ; actual note length duration voice 2
                STY V1DUR+14            ; actual note length duration voice 3
                STY SAMPLEACTDUR        ; actual sample duration
                STY BEATCOUNT           ; set pattern index to the beginning
                STY BEATCOUNT+7
                STY BEATCOUNT+14
                STY SAMPLEBEATCOUNT     ; set pattern index of sample voice to the beginning

                INY
                STY SPEED               ; actual cycle timer (speed of song)
                JMP QUIT

MUSIC           LDA SAMPLEVOL           ; 0=volume to middle
                BNE OKMUSIC

                LDA #$09
                STA $D418               ; Select Filter Mode and Volume (to middle)

OKMUSIC         LDY SOUND,X             ; index of instrument data
                LDA VDATA+7,Y           ; instrument effect
                AND #$04                ; is effect implex (4)?
                BEQ NOIMPLEX

                LDA IMPLEX,X            ; read implex flag
                BEQ NORMAL

                DEC IMPLEX,X            ; dec implex (reset)
                LDA VDATA2+2,Y          ; control register
                STA $D404,X             ; Voice 1: Control Register
                BNE NOIMPLEX
```

```
NORMAL          LDA VDATA+1,Y           ; control register
                STA $D404,X             ; Voice 1: Control Register
NOIMPLEX
                LDA SPEED               ; actual cycle timer (speed of song)
                BNE GOCHECKFX

                DEC V1DUR,X             ; actual note length duration voice 1
                BMI MAINLOOP
GOCHECKFX
                JMP CHECKFX

SETPOINTS
                LDY TN                  ; song (track) number
                LDA VOICE1L,Y           ; get track 1 from song (low)
                STA V1LO                ; set current track 1 position (base low)
                LDA VOICE1H,Y           ; get track 1 from song (high)
                STA V1HI                ; set current track 1 position (base high)
                LDA VOICE2L,Y           ; get track 2 from song (low)
                STA V2LO                ; set current track 2 position (base low)
                LDA VOICE2H,Y           ; get track 2 from song (high)
                STA V2HI                ; set current track 2 position (base high)
                LDA VOICE3L,Y           ; get track 3 from song (low)
                STA V3LO                ; set current track 3 position (base low)
                LDA VOICE3H,Y           ; get track 3 from song (high)
                STA V3HI                ; set current track 3 position (base high)

                LDA VOICE4LO,Y          ; get track 4 (sample) from song (low)
                STA V4LO                ; set current track 4 (sample) position (base low)
                LDA VOICE4HI,Y          ; get track 4 (sample) from song (high)
                STA V4HI                ; set current track 4 (sample) position (base high)

                LDA TDATA,Y             ; read the song speed
                STA TEMPOBYTE           ; set cycle timer (speed of song - tempo)
                JMP SETCONT             ; init music

QUIT2           DEC SPEED               ; actual cycle timer (speed of song)
                BPL QUIT
                LDA TEMPOBYTE           ; get cycle timer (speed of song - tempo)
                STA SPEED               ; actual cycle timer (speed of song)

QUIT            LDA #$AB
                STA TN                  ; song (track) number
                RTS

MAINLOOP        LDA V1LO,X              ; current track 1 position (base low)
                STA POINTS              ; track pattern pointer (low)
                LDA V1HI,X              ; current track 1 position (base high)
                STA POINTS+1            ; track pattern pointer (high)

AGAIN4          LDY BARCOUNT,X          ; read actual track position (offset - bar counter)
                LDA (POINTS),Y          ; read actual track pattern pointer value

NOTEND2         TAY
                LDA BARLO,Y             ; read pattern pointer low
                STA BARS                ; pattern pointer low
                LDA BARHI,Y             ; read pattern pointer high
                STA BARS+1              ; pattern pointer high

                LDA #$FF
                STA GATEBYTE            ; mask gate byte to let all as is

                LDA #$00
                STA V1SLIDE,X           ; slide flag
                STA V1PLEX,X            ; no plex (arpeggio)
                STA V1VIB,X             ; no vibrato
                                        ; read the pattern value of note to play
AGAIN           LDY BEATCOUNT,X         ; load pattern index of this voice
                LDA (BARS),Y            ; read a pattern value

AGAIN3          CMP #$FD                ; plex?
                BCC SLIDE               ; jump if <$FD

                INY                     ; next pattern value index
                INC BEATCOUNT,X         ; next (saved) pattern value index
                LDA (BARS),Y
                STA NEWDUR,X            ; new note duration

REGET           INC BEATCOUNT,X
                BNE AGAIN

SLIDE           CMP #$FB                ; slide down ?
                BCC NEWVOICE            ; jump if <$FB

                CMP #$FB                ; slide down ?
                BNE SLIDEUP             ; jump if <>$FB

                LDA #$01
```

```
SLIDECONT       STA V1SLIDE,X          ; store in slide flag
                INY
                INC BEATCOUNT,X
                LDA (BARS),Y
                STA SLIDELO,X          ; store slide value

                LDA #$00
                STA V1PLEX,X           ; no plex (arpeggio)
                STA V1VIB,X            ; no vibrato
                BEQ REGET

SLIDEUP         LDA #$02               ; positive slide (up - portamento)
                BNE SLIDECONT          ; store portamento value

NEWVOICE        CMP #$FA               ; is new instrument?
                BCC NOBV           ; jump if <$FA

;=======================
; New Instrument
;=======================
                                       ; select new instruments (voice)
                INY
                INC BEATCOUNT,X        ; next (saved) pattern value index
                LDA (BARS),Y           ; read a pattern value (instrument index)

                ASL A
                ASL A
                ASL A                  ; index * 8
                STA SOUND,X            ; index of instrument data
                TAY

                LDA VDATA,Y            ; read Hi/Lo of pulsation amplitude
                PHA
                AND #$0F
                STA V1PULSEHI,X        ; Actual Wave form pulsation amplitude (hi byte)
                STA PWH,X              ; Wave form pulsation amplitude (hi byte)
                PLA

                AND #$F0
                STA V1PULSELO,X        ; actual wave form pulsation amplitude (lo byte)
                STA PWL,X              ; Wave form pulsation amplitude (lo byte)
                JMP REGET

NOBV            STA BARVALUE,X
                LDA NEWDUR,X           ; new note duration
                STA V1DUR,X            ; actual note length duration voice 1

                LDA #$00
                STA CYCLEINT,X
                STA CYCLEEST,X

                LDA #$02
                STA IMPLEX,X

                LDY SOUND,X            ; index of instrument data
                LDA VDATA+7,Y          ; instrument effect
                AND #$02               ; is effect wave (2)?
                BEQ PLAYCONT
                                       ; reload wave modulation for the new note
                LDA PWL,X              ; Wave form pulsation amplitude (lo byte)
                STA V1PULSELO,X        ; actual wave form pulsation amplitude (lo byte)

                LDA PWH,X              ; Wave form pulsation amplitude (hi byte)
                STA V1PULSEHI,X        ; Actual Wave form pulsation amplitude (hi byte)

PLAYCONT        LDA BARVALUE,X         ; value of bar (pattern) - read note to play
                BNE NOREST

;===============
; A rest (no note)
;===============
PLAYCONT2       LDA TEMP3,X            ; note to play
                STA BARVALUE,X         ; value of bar (pattern) - read note to play
                LDA #$00
                STA TEMP3,X            ; note to play
                LDY SOUND,X            ; index of instrument data
                DEC GATEBYTE           ; release gate
                BNE NOPITCH

;===============
; Out note
;===============
NOREST          STA TEMP3,X            ; note to play
                TAY
                LDA NTH,Y              ; get high frequency from table
                STA $D401,X            ; Voice 1: Frequency Control - High-Byte
                STA V1HIFREQ,X         ; Voice 1: Frequency control (hi byte) for effect 1
                STA C1NHIGH,X          ; high frequency for vibrato/slide/drum
```

```
                LDA NTL,Y               ; get low frequency from table
                STA $D400,X             ; Voice 1: Frequency Control - Low-Byte
                STA V1LOFREQ,X          ; Voice 1: Frequency control (lo byte)
                STA C1NLOW,X            ; low frequency for vibrato/slide/drum

                LDY SOUND,X             ; index of instrument data
                LDA VDATA+6,Y           ; control register 2
                STA $D404,X             ; Voice 1: Control Register

NOPITCH         LDA VDATA+1,Y           ; control register
                AND GATEBYTE            ; manipulate with gate byte
                STA $D404,X             ; Voice 1: Control Register

                LDA VDATA+2,Y           ; read A/D value
                STA $D405,X             ; Voice 1: Attack / Decay Cycle Control

                LDA VDATA+3,Y           ; read S/R value
                STA $D406,X             ; Voice 1: Sustain / Release Cycle Control

                LDA V1PULSELO,X         ; actual wave form pulsation amplitude (lo byte)
                STA $D402,X             ; Voice 1: Pulse Waveform Width - Low-Byte

                LDA V1PULSEHI,X         ; Actual Wave form pulsation amplitude (hi byte)
                STA $D403,X             ; Voice 1: Pulse Waveform Width - High-Nybble

                INC BEATCOUNT,X         ; next pattern index
                LDY BEATCOUNT,X         ; read pattern index
                LDA (BARS),Y
                CMP #$FF                ; end of pattern?
                BNE FXSETUP

                LDA #$00
                STA BEATCOUNT,X         ; reset pattern index
                INC BARCOUNT,X          ; inc actual track position (offset - bar counter)
                LDY BARCOUNT,X          ; read actual track position (offset - bar counter)
                LDA (POINTS),Y
                CMP #$FF                ; repeat the song (track)?
                BNE NOTEND

                LDA #$00                ; start at beginning
                STA BARCOUNT,X          ; actual track position (offset - bar counter)
                BEQ FXSETUP

NOTEND          CMP #$FE                ; end of song (track)?
                BNE FXSETUP
                LDA #$00
                STA TN                  ; song (track) number
                RTS

FXSETUP         LDA TEMP3,X             ; temp pattern value (note to play)
                BEQ CHECKFX

                LDY SOUND,X             ; index of instrument data
                LDA V1SLIDE,X           ; slide flag
                BNE ALREADY

                LDA VDATA2+4,Y          ; instrument slide flag
                BEQ NOBEND

                STA V1SLIDE,X           ; slide flag
                LDA VDATA2+3,Y          ; instrument slide value
                STA SLIDELO,X           ; store slide value
ALREADY         JMP SLIDECHECK

NOBEND          LDA VDATA+5,Y           ; hi/lo value for plex (arpeggio)
                BEQ NOPLEX
                JMP PLEXSETUP
NOPLEX
                STA V1PLEX,X            ; plex (arpeggio)
                LDA VDATA2,Y            ; oscillating frequency value (for vibrato)
                BEQ EXITVIB
                JMP VIBSETUP

EXITVIB         STA V1VIB,X             ; vibrato flag
                JMP QUIT

;=================================
; pulse modulation timbre routine
;=================================
CHECKFX         LDA VDATA+4,Y           ; Wave amplitude inc/dec value
                STA PTEMP               ; store for late use
                BEQ PLEXCHECK

                LDA PMODDIR,X           ; direction of pulse modulation
                BNE PDOWN

                CLC
```

```
                LDA V1PULSELO,X         ; actual wave form pulsation amplitude (lo byte)
                ADC PTEMP               ; add incremental value
                STA V1PULSELO,X         ; actual wave form pulsation amplitude (lo byte)
                STA $D402,X             ; Voice 1: Pulse Waveform Width - Low-Byte

                LDA V1PULSEHI,X         ; Actual Wave form pulsation amplitude (hi byte)
                ADC #$00
                STA V1PULSEHI,X         ; Actual Wave form pulsation amplitude (hi byte)
                STA $D403,X             ; Voice 1: Pulse Waveform Width - High-Nybble

                CLC
                CMP #$0E
                BCC PLEXCHECK
                INC PMODDIR,X           ; change direction of pulse modulation
                BNE PLEXCHECK

PDOWN           LDA V1PULSELO,X         ; actual wave form pulsation amplitude (lo byte)
                SEC
                SBC PTEMP
                STA V1PULSELO,X         ; actual wave form pulsation amplitude (lo byte)
                STA $D402,X             ; Voice 1: Pulse Waveform Width - Low-Byte
                LDA V1PULSEHI,X         ; Actual Wave form pulsation amplitude (hi byte)
                SBC #$00
                STA V1PULSEHI,X         ; Actual Wave form pulsation amplitude (hi byte)
                STA $D403,X             ; Voice 1: Pulse Waveform Width - High-Nybble
                CLC
                CMP #$08
                BCS PLEXCHECK
                DEC PMODDIR,X           ; change direction of pulse modulation

PLEXCHECK       LDA V1PLEX,X            ;  plex (arpeggio)
                BEQ VIBUPDATE

;===============
; plex timbre routine
;===============
                LDA PLEXTEMP,X          ; plex index in table
                ASL A
                TAY
                LDA PLEXLH,Y            ; plex table index low
                STA PLEXADD+1
                LDA PLEXLH+1,Y          ; plex table index high
                STA PLEXADD+2

                LDA PLEXC,X             ; read actual index
                CMP PLEXCOUNT,X         ; plex counter (table dimension) reached?
                BNE PLEXCONT

                LDA #$00
                STA PLEXC,X             ; actual index

PLEXCONT        TAY
                LDA BARVALUE,X          ; value of bar (pattern) - read note to play
                CLC
PLEXADD         ADC P0,Y
                TAY
                LDA NTL,Y               ; get low frequency from table
                STA $D400,X             ; Voice 1: Frequency Control - Low-Byte
                LDA NTH,Y               ; get high frequency from table
                STA $D401,X             ; Voice 1: Frequency Control - High-Byte
                INC PLEXC,X             ; inc actual plex index
                JMP QUIT

VIBUPDATE       LDA V1VIB,X             ; vibrato flag
                BNE OKVIB
                JMP SLIDECHECK

;==================================
; make the vibrato
; Vibrato direction:
;  0 = down (first time)
;  1 = up
;  2 = up
;  3 = down
;  4 = down
;==================================
OKVIB           LDA VIBDIR,X            ; vibrato direction flag
                BEQ VIBDOWN

                CMP #$03
                BCC VIBUP               ; jump if <03
                                        ; vibrato down
                SEC
                LDA C1NLOW,X            ; low frequency for vibrato/slide/drum
                SBC VIBSTEP,X           ; sub vibrato step
                STA C1NLOW,X            ; low frequency for vibrato/slide/drum
                STA $D400,X             ; Voice 1: Frequency Control - Low-Byte
```

```
                LDA C1NHIGH,X           ; high frequency for vibrato/slide/drum
                SBC #$00
                STA C1NHIGH,X           ; high frequency for vibrato/slide/drum
                STA $D401,X             ; Voice 1: Frequency Control - High-Byte

                DEC VIBTEMP,X           ; dec actual temporary vibrato
                BNE VIBEND

                LDA VIBTIME,X           ; read stored value of vibrato time (counter)
                STA VIBTEMP,X           ; set to actual temporary vibrato

                INC VIBDIR,X            ; change vibrato direction flag
                LDA VIBDIR,X            ; vibrato direction flag
                CMP #$05
                BCC VIBEND              ; jump if <05

                LDA #$01                ; direction up
                STA VIBDIR,X            ; change vibrato direction flag

VIBEND          JMP QUIT
                                        ; vibrato down
VIBDOWN
                SEC
                LDA C1NLOW,X            ; low frequency for vibrato/slide/drum
                SBC VIBSTEP,X           ; sub vibrato step
                STA C1NLOW,X            ; low frequency for vibrato/slide/drum
                STA $D400,X             ; Voice 1: Frequency Control - Low-Byte

                LDA C1NHIGH,X           ; high frequency for vibrato/slide/drum
                SBC #$00
                STA C1NHIGH,X           ; high frequency for vibrato/slide/drum
                STA $D401,X             ; Voice 1: Frequency Control - High-Byte

                DEC VIBTEMP,X           ; dec actual temporary vibrato
                BNE VIBEND2

                LDA VIBTIME,X           ; read stored value of vibrato time (counter)
                STA VIBTEMP,X           ; set to actual temporary vibrato
                INC VIBDIR,X            ; change vibrato direction flag
VIBEND2         JMP QUIT

VIBUP           CLC
                LDA C1NLOW,X            ; low frequency for vibrato/slide/drum
                ADC VIBSTEP,X           ; add vibrato step
                STA C1NLOW,X            ; low frequency for vibrato/slide/drum
                STA $D400,X             ; Voice 1: Frequency Control - Low-Byte
                LDA C1NHIGH,X           ; high frequency for vibrato/slide/drum
                ADC #$00
                STA C1NHIGH,X           ; high frequency for vibrato/slide/drum
                STA $D401,X             ; Voice 1: Frequency Control - High-Byte
                DEC VIBTEMP,X           ; dec actual temporary vibrato
                BNE NODRUMS
                LDA VIBTIME,X           ; read stored value of vibrato time (counter)
                STA VIBTEMP,X           ; set to actual temporary vibrato
                INC VIBDIR,X            ; change vibrato direction flag
                BNE NODRUMS
                JMP QUIT
;=================================
; Slide timbre routine
;=================================
; slide flag:
;  0= none
;  1= down
;  2= up
;  3= down high
;  4= up high
SLIDECHECK      LDA V1SLIDE,X           ; slide flag
                BEQ NOMOREFX
                CMP #$01                ; negative slide (down)
                BEQ SLIDEDOWN2
                CMP #$02                ; positive slide (up)
                BEQ SLIDEUP2
                CMP #$03
                BEQ HIGHDOWN            ; negative only high slide (down)

                CLC
                LDA C1NHIGH,X           ; Voice 1: Frequency control (hi byte) for slide
                ADC SLIDELO,X           ; add slide value
                STA C1NHIGH,X           ; Voice 1: Frequency control (hi byte) for slide
                STA $D401,X             ; Voice 1: Frequency Control - High-Byte
                JMP NOMOREFX

SLIDEDOWN2      CLC
                LDA C1NLOW,X            ; low frequency for vibrato/slide/drum
                SBC SLIDELO,X           ; sub slide value
                STA C1NLOW,X            ; low frequency for vibrato/slide/drum
                STA $D400,X             ; Voice 1: Frequency Control - Low-Byte
```

```
                LDA C1NHIGH,X          ; high frequency for vibrato/slide/drum
                SBC #$00
                STA C1NHIGH,X          ; high frequency for vibrato/slide/drum
                STA $D401,X            ; Voice 1: Frequency Control - High-Byte
                JMP NOMOREFX

HIGHDOWN        SEC
                LDA C1NHIGH,X          ; high frequency for vibrato/slide/drum
                SBC SLIDELO,X          ; sub slide value
                STA C1NHIGH,X          ; high frequency for vibrato/slide/drum
                STA $D401,X            ; Voice 1: Frequency Control - High-Byte
                JMP NOMOREFX

SLIDEUP2        CLC
                LDA C1NLOW,X           ; low frequency for vibrato/slide/drum
                ADC SLIDELO,X          ; add slide value
                STA C1NLOW,X           ; low frequency for vibrato/slide/drum
                STA $D400,X            ; Voice 1: Frequency Control - Low-Byte
                LDA C1NHIGH,X          ; high frequency for vibrato/slide/drum
                ADC #$00
                STA C1NHIGH,X          ; high frequency for vibrato/slide/drum
                STA $D401,X            ; Voice 1: Frequency Control - High-Byte

NOMOREFX        LDY SOUND,X            ; index of instrument data
                LDA VDATA+7,Y          ; instrument effect
                AND #$01               ; is effect drum (1)?
                BEQ NODRUMS
                JMP EFFECT1
NODRUMS
                JMP QUIT

V1VIB           .BYTE $00              ; vibrato flag
V1PLEX          .BYTE $01              ; plex flag
V1SLIDE         .BYTE $00              ; slide flag (0= none, 1= down, 2= up, 3= down high, 4= up high)
CYCLEINT        .BYTE $00
CYCLEEST        .BYTE $00
BEATCOUNT       .BYTE $00
PMODDIR         .BYTE $01              ; direction of pulse modulation
                .BYTE $00, $00, $00, $00, $00, $00, $00
                .BYTE $00, $00, $00, $00, $00, $00, $00

SLIDELO         .BYTE $1F              ; slide value
FADEFLAG        .BYTE $00              ; not used
NEWDUR          .BYTE $01              ; new note duration
SOUND           .BYTE $18              ; index of instrument data
V1PULSELO       .BYTE $40              ; Actual Wave form pulsation amplitude (lo byte)
PWL             .BYTE $40              ; Wave form pulsation amplitude (lo byte)
V1PULSEHI       .BYTE $0B              ; Actual Wave form pulsation amplitude (hi byte)
                .BYTE $10, $00, $01, $08, $80, $40, $02
                .BYTE $00, $00, $01, $08, $80, $40, $02

PWH             .BYTE $05               ; Wave form pulsation amplitude (hi byte)
PLEXTEMP        .BYTE $00               ; plex index in table
V1LO            .BYTE $C9               ; current track 1 position (base low)
V1HI            .BYTE $27               ; current track 1 position (base high)
BARCOUNT        .BYTE $0A               ; bar counter (track position)
                .BYTE $00
V1DUR           .BYTE $00              ; actual note length duration voice
                .BYTE $01, $00, $1E, $28, $10, $00, $00
                .BYTE $01, $00, $47, $28, $10, $00, $00

                .BYTE $00
                .BYTE $00
TN              .BYTE $AB              ; song (track) number
TEMPOBYTE       .BYTE $04              ; cycle timer (speed of song - tempo)
PTEMP           .BYTE $40              ; Wave form pulsation amplitude step
SPEED           .BYTE $04              ; actual cycle timer (speed of song)
GATEBYTE        .BYTE $FF              ; set ON/OFF the gate (ADS phase) - mask gate byte

C1NLOW          .BYTE $C3              ; low frequency for vibrato/slide
V1LOFREQ        .BYTE $C3              ; Frequency control (low byte)
V1HIFREQ        .BYTE $11              ; Frequency control (high byte)
BARVALUE        .BYTE $31              ; value of bar (pattern)
C1NHIGH         .BYTE $11              ; high frequency for vibrato/slide/drum
PLEXCOUNT       .BYTE $04              ; plex counter (table dimension)
PLEXC           .BYTE $01              ; actual plex counter
                .BYTE $61, $61, $08, $24, $08, $04, $01
                .BYTE $30, $30, $04, $18, $04, $00, $00

VIBDIR          .BYTE $00              ; vibrato direction
VIBSTEP         .BYTE $00              ; vibrato step
VIBTIME         .BYTE $00              ; vibrato time (counter)
VIBTEMP         .BYTE $00              ; temporary vibrato value
VIBH            .BYTE $00              ; not used
VIBL            .BYTE $00              ; not used
TEMP3           .BYTE $31              ; temp pattern value (note to play)
                .BYTE $00, $00, $00, $00, $00, $00, $24
                .BYTE $00, $00, $00, $00, $00, $00, $18
```

```
IMPLEX          .BYTE $00
                .BYTE $00
                .BYTE $00
                .BYTE $00
                .BYTE $00
                .BYTE $00
                .BYTE $00
                .BYTE $02, $00, $00, $00, $00, $00, $00
                .BYTE $02, $00, $00, $00, $00, $00, $00

;=======================
; Note frequency table
;=======================
NTL             .BYTE $0C, $1C, $2D, $3E, $51, $66, $7B, $91
                .BYTE $A9, $C3, $DD, $FA, $18, $38, $5A, $7D
                .BYTE $A3, $CC, $F6, $23, $53, $86, $BB, $F4
                .BYTE $30, $70, $B4, $FB, $47, $98, $ED, $47
                .BYTE $A7, $0C, $77, $E9, $61, $E1, $68, $F7
                .BYTE $8F, $30, $DA, $8F, $4E, $18, $EF, $D2
                .BYTE $C3, $C3, $D1, $EF, $1F, $60, $B5, $1E
                .BYTE $9C, $31, $DF, $A5, $87, $86, $A2, $DF
                .BYTE $3E, $C1, $6B, $3C, $39, $63, $BE, $4B
                .BYTE $0F, $0C, $45, $BF, $7D, $83, $D6, $79
                .BYTE $73, $C7, $7C, $97, $1E, $18, $8B, $7E
                .BYTE $FA, $06, $AC, $F3, $E6, $8F, $F8, $2E

NTH             .BYTE $01, $01, $01, $01, $01, $01, $01, $01
                .BYTE $01, $01, $01, $01, $02, $02, $02, $02
                .BYTE $02, $02, $02, $03, $03, $03, $03, $03
                .BYTE $04, $04, $04, $04, $05, $05, $05, $06
                .BYTE $06, $07, $07, $07, $08, $08, $09, $09
                .BYTE $0A, $0B, $0B, $0C, $0D, $0E, $0E, $0F
                .BYTE $10, $11, $12, $13, $15, $16, $17, $19
                .BYTE $1A, $1C, $1D, $1F, $21, $23, $25, $27
                .BYTE $2A, $2C, $2F, $32, $35, $38, $3B, $3F
                .BYTE $43, $47, $4B, $4F, $54, $59, $5E, $64
                .BYTE $6A, $70, $77, $7E, $86, $8E, $96, $9F
                .BYTE $A8, $B3, $BD, $C8, $D4, $E1, $EE, $FD

;=======================
; Plex table index
;=======================
PLEXLH          .WORD P0,P1,P2,P3,P4,P5,P6,P7
;=======================
; Plex definitions
;=======================
P0              .BYTE $13, $0C, $07, $00
P1              .BYTE $00, $07, $0A, $0C
P2              .BYTE $00, $03, $07, $0C
P3              .BYTE $00, $04, $07, $0C
P4              .BYTE $00, $05, $09, $0C
P5              .BYTE $00, $05, $07, $0C
P6              .BYTE $00, $04, $09, $0C
P7              .BYTE $00, $03, $08, $0C

SETIRQ          SEI
                LDA #INTER&255
                STA $0314
                LDA #INTER/256
                STA $0315
                LDX #$00
                STX $DC0E               ; Control Register A
                INX
                STX $D01A               ; Interrupt Mask Register (IMR)
                CLI
                RTS

INTER           LDA #$01
                STA $D019               ; Interrupt Request Register (IRR)
                LDA #$82
                STA $D012               ; Raster Position
                LDA #$1B
                STA $D011               ; Control Register 1
                NOP
                NOP
                NOP
                NOP
                NOP
                JSR MUSICROUTINE
                NOP
                NOP
                NOP
                JMP $EA31               ; Main IRQ Entry Point

MUSICROUTINE    LDX #$00               ; voice 1
                JSR MAIN
                LDX #$07               ; voice 2
```

```
                JSR MAIN
                LDX #$0E                ; voice 3
                JSR MAIN
                JSR MAINEXTRA
                RTS

;-------------------------------------------------------------------------
                .TEXT   '(C)1987 MATT GRAY'             ; CHANGED FROM .BYTE TO .TEXT
;-------------------------------------------------------------------------

;================
; Set up the plex
;================
PLEXSETUP
                PHA
                AND #$0F
                STA PLEXTEMP,X          ; plex index in table
                PLA
                AND #$F0
                LSR A
                LSR A
                LSR A
                LSR A
                STA PLEXCOUNT,X         ; plex counter (table dimension)

                LDA #$00
                STA PLEXC,X             ; reset actual plex counter

                LDA #$01
                STA V1PLEX,X            ; on plex

                LDA #$00
                STA V1VIB,X             ; no vibrato
                JMP QUIT

;===============
; Set up the vibrato
;===============
VIBSETUP        STA VIBSTEP,X           ; vibrato step
                LDA VDATA2+1,Y          ; length of vibrato
                STA VIBTIME,X           ; set vibrato time (counter)
                STA VIBTEMP,X           ; set to actual temporary vibrato
                LDA #$00
                STA V1PLEX,X            ; reset plex
                STA VIBDIR,X            ; reset actual vibrato delay
                LDA #$01
                STA V1VIB,X             ; on vibrato
                JMP QUIT

;=================================
; instruments part 1
; 0: wave form pulsation amplitude LO/HI -> 00HI/LO00
; 1: Control register
; 2: A/D value
; 3: S/R value
; 4: Wave amplitude inc/dec value
; 5: not used
; 6: Control register 2 (at new instrument and new note start)
; 7: instrument effect
;    1: a frequency effect
;    2: a pulse wave effect
;    4: implex (switch between waveform)
;
;   16: hat effect
;=================================
VDATA           .BYTE $00, $81, $0A, $00, $00, $00, $80, $01
                .BYTE $41, $41, $0B, $00, $40, $00, $40, $02
                .BYTE $94, $43, $00, $EC, $15, $00, $42, $00
                .BYTE $45, $41, $09, $00, $60, $40, $40, $04
                .BYTE $6A, $41, $00, $20, $20, $41, $40, $00
                .BYTE $00, $00, $00, $00, $00, $00, $00, $02
                .BYTE $90, $43, $0F, $00, $07, $00, $42, $02
                .BYTE $00, $81, $08, $00, $00, $00, $80, $01
                .BYTE $60, $41, $0D, $00, $30, $00, $40, $02
                .BYTE $41, $43, $0F, $00, $30, $00, $42, $02
                .BYTE $60, $41, $00, $90, $30, $43, $40, $02
                .BYTE $60, $41, $00, $90, $30, $44, $40, $02
                .BYTE $90, $41, $00, $90, $2A, $00, $40, $00
                .BYTE $90, $41, $00, $90, $2A, $00, $40, $02
                .BYTE $60, $41, $00, $90, $30, $45, $30, $02
                .BYTE $98, $41, $09, $00, $00, $00, $40, $01
                .BYTE $50, $41, $00, $90, $40, $47, $40, $02
                .BYTE $50, $41, $00, $90, $40, $43, $40, $02
                .BYTE $50, $41, $00, $90, $40, $46, $40, $02

;=================================
; instruments part 2
; 0: oscillating frequency value (for vibrato)
```

```
; 1: length of vibrato intensity (for vibrato)
; 2: Control register for effect implex (4)
; 3: slide value
; 4: slide flag (0= none, 1= down, 2= up, 3= down high, 4= up high)
; 5: duration cycle for effect 1
; 6: not used
; 7: not used
;================================
VDATA2          .BYTE $00, $00, $11, $00, $00, $03, $00, $00
                .BYTE $00, $00, $81, $00, $00, $00, $00, $00
                .BYTE $00, $00, $81, $00, $00, $00, $00, $00
                .BYTE $00, $00, $81, $00, $00, $00, $00, $00
                .BYTE $00, $00, $00, $00, $00, $00, $00, $00
                .BYTE $00, $00, $81, $00, $00, $00, $00, $00
                .BYTE $00, $00, $81, $00, $00, $00, $00, $00
                .BYTE $00, $00, $11, $41, $01, $01, $00, $00
                .BYTE $00, $00, $00, $00, $00, $00, $00, $00
                .BYTE $00, $00, $81, $10, $03, $00, $00, $00
                .BYTE $30, $02, $00, $00, $00, $00, $00, $00
                .BYTE $00, $00, $00, $00, $00, $00, $00, $00
                .BYTE $30, $02, $00, $00, $00, $00, $00, $00
                .BYTE $30, $02, $00, $00, $00, $00, $00, $00
                .BYTE $30, $02, $00, $00, $00, $00, $00, $00
                .BYTE $00, $00, $41, $F0, $01, $01, $00, $00
                .BYTE $30, $02, $00, $00, $00, $00, $00, $00
                .BYTE $30, $02, $00, $00, $00, $00, $00, $00
                .BYTE $30, $02, $00, $00, $00, $00, $00, $00

;=====================================
; pointer to bars (patterns) low address
;=====================================
BARLO           .BYTE   T0&255,T1&255,T2&255,T3&255,T4&255,T5&255
                .BYTE   T6&255,T7&255,T8&255,T9&255,T10&255
                .BYTE   T11&255,T12&255,T13&255,T14&255,T15&255
                .BYTE   T16&255,T17&255,T18&255,T19&255,T20&255,T21&255
                .BYTE   T22&255,T23&255,T24&255,T25&255,T26&255
                .BYTE   T27&255,T28&255,T29&255

;=====================================
; pointer to bars (patterns) high address
;=====================================
BARHI           .BYTE   T0/256,T1/256,T2/256,T3/256,T4/256,T5/256
                .BYTE   T6/256,T7/256,T8/256,T9/256,T10/256
                .BYTE   T11/256,T12/256,T13/256,T14/256,T15/256
                .BYTE   T16/256,T17/256,T18/256,T19/256,T20/256,T21/256
                .BYTE   T22/256,T23/256,T24/256,T25/256,T26/256
                .BYTE   T27/256,T28/256,T29/256

;================================
; Songs (tunes) pointers
;================================
VOICE1L         .BYTE $00, TUNE1&255, EMPTY1&255
VOICE1H         .BYTE $00, TUNE1/256, EMPTY1/256

VOICE2L         .BYTE $00, TUNE2&255, EMPTY2&255
VOICE2H                 .BYTE $00, TUNE2/256, EMPTY2/256

VOICE3L                 .BYTE $00, TUNE3&255, EMPTY3&255
VOICE3H         .BYTE $00, TUNE3/256, EMPTY3/256

VOICE4LO                .BYTE $00, TUNE4&255, EMPTY4&255
VOICE4HI        .BYTE $00, TUNE4/256, EMPTY4/256

EFFECT1         LDA V1HIFREQ,X          ; Voice 1: Frequency control (hi byte) for effect 1
                BEQ NODEC
                DEC V1HIFREQ,X          ; dec Frequency control (hi byte) for effect 1

NODEC           LDA CYCLEINT,X
                BEQ TESTENDCYCLE1

                DEC CYCLEINT,X
                LDA #$81
                STA $D404,X             ; Voice 1: Control Register

                LDA V1HIFREQ,X          ; Voice 1: Frequency control (hi byte) for effect 1
                EOR #$23
                STA $D401,X             ; Voice 1: Frequency Control - High-Byte
                JMP QUIT

TESTENDCYCLE1   JMP TESTENDCYCLE

CHANGEFREQ      LDA C1NHIGH,X           ; high frequency for vibrato/slide/drum
                STA $D401,X             ; Voice 1: Frequency Control - High-Byte
                STA V1HIFREQ,X          ; Voice 1: Frequency control (hi byte) for effect 1
                LDA VDATA2+2,Y          ; control register
                STA $D404,X             ; Voice 1: Control Register
                JMP QUIT
```

```
TESTENDCYCLE    LDA CYCLEEST,X
                CMP VDATA2+5,Y          ; duration cycle for effect 1
                BEQ RESETCYCLE

                INC CYCLEINT,X
                INC CYCLEEST,X
                BNE CHANGEFREQ

RESETCYCLE      LDA #$00
                STA CYCLEEST,X
                STA CYCLEINT,X
                BEQ CHANGEFREQ

;========================
; Song speed
;========================
TDATA           .BYTE   $00,$04,$04

;=======================
; song patterns
;=======================
;  XX:  pattern XX
; $FF:  repeat the track
; $FE:  end of music
TUNE1           .BYTE $04, $00, $00, $0F, $0F, $0F, $0F, $0D
                .BYTE $0D, $0F, $0F, $0F, $0F, $0F, $0F, $0F
                .BYTE $0F, $FF

T27DB           .BYTE $0B, $0B, $0B, $0B, $0B, $0B, $0B, $0B
                .BYTE $0B, $0B, $0B, $0B, $0B, $0B, $0B, $0B
                .BYTE $0B, $0B, $0B, $0B, $0B, $0B, $0B, $0B
                .BYTE $0B, $0B, $0B, $0B, $0B, $0B, $0B, $0B
                .BYTE $0B, $0B, $0B, $0B, $0B, $0B, $0B, $0B
                .BYTE $0B, $0B, $0B, $0B, $0B, $0B, $0B, $0B
                .BYTE $0B, $0B, $0B, $0B, $0B, $0B, $0B, $0B
                .BYTE $0B, $0B, $0B, $0B, $0B, $0B, $0B, $0B
                .BYTE $0A, $0A
                .BYTE $FF

TUNE2           .BYTE $03, $03, $03, $0C, $0C, $0C, $0C, $0F
                .BYTE $0F, $0F, $0F, $0F, $0F, $0F, $0F, $0C
                .BYTE $0C, $0C, $0C, $0C, $0C, $0C, $0C
                .BYTE $FF

T2836           .BYTE $10, $07, $03, $03, $17, $17, $17, $17
                .BYTE $18, $18, $03, $03, $19, $19, $1B, $1B
                .BYTE $FF

TUNE3           .BYTE $00, $00, $00, $01, $01, $01, $01, $01
                .BYTE $01, $01, $01, $01, $01, $01, $01, $01
                .BYTE $01, $01, $01, $01, $01, $01, $01
                .BYTE $FF


T285F           .BYTE $0C, $12, $12, $14, $14, $0C, $0C, $00
                .BYTE $00, $00, $16, $16, $16, $08, $18, $18
                .BYTE $18, $18, $18, $18, $04, $04, $06, $1A
                .BYTE $1A, $1C, $1C
                .BYTE $FF

TUNE4           .BYTE $1D, $09, $0A, $0A, $0A, $0B, $0A, $0A
                .BYTE $0A, $0E, $06, $06, $06, $07, $06, $06
                .BYTE $06, $07, $06, $06, $06, $07, $06, $06
                .BYTE $06, $07
                .BYTE $FF

;=================================
; pattern data
;=================================
; format:
; $00           : rest
; xx            : note xx
; $FA nn        : select instrument nn
; $FB mm        : negative portamento (mm)
; $FC kk        : positive portamento (kk)
; $FD kk        : duration kk
; $FF           : end of pattern
;=================================
; sample pattern data
;=================================
; format:
; NN            : sample speed
; $FD  VV       : sample length duration
; $FA  BI       : Bank (B=0|1), Index of Sample (I)

;***USED**NT****
T0              .BYTE $FA, $04                              ; select instrument
                .BYTE $FD, $3F                              ; select duration
```

```
                .BYTE $00
                .BYTE $FF
;***USED**NT****
T1              .BYTE $FA, $01                          ; select instrument
                .BYTE $FD, $01                          ; select duration
                .BYTE C2, C2, C3, C2, C2, C2, AS1, B1
                .BYTE $FF
;**NOT DECLARED**
T28A9           .BYTE $19, $25, $19, $19, $25, $25, $19, $25
                .BYTE $FF
;***USED**NT****
T3              .BYTE $FA, $09                          ; select instrument
                .BYTE $FD, $3F                          ; select duration
                .BYTE C4
                .BYTE $FF

T2              .BYTE $FD, $0F
                .BYTE $FA, $04
                .BYTE $00
                .BYTE $FF
;***USED**NT****
T4              .BYTE $FA, $08                          ; select instrument
                .BYTE $FD, $00                          ; select duration
                .BYTE $00
                .BYTE $FD, $3E                          ; select duration
                .BYTE C2
                .BYTE $FF

T5              .BYTE $FD, $07
                .BYTE $FA, $13
                .BYTE $49, $36, $16
                .BYTE $FD, $03
                .BYTE $09, $16
                .BYTE $FF
;***USED**ST****
T6              .BYTE $FD, $01                           ; sample length duration
                .BYTE $FA, $01                           ; Bank (B=0|1), Index of Sample (I)
                .BYTE $0A, $0A
                .BYTE $FA, $02                           ; Bank (B=0|1), Index of Sample (I)
                .BYTE $FD, $02                           ; sample length duration
                .BYTE $40, $40
                .BYTE $FD, $01                           ; sample length duration
                .BYTE $FA, $01                           ; Bank (B=0|1), Index of Sample (I)
                .BYTE $0A
                .BYTE $FD, $02                           ; sample length duration
                .BYTE $FA, $02                           ; Bank (B=0|1), Index of Sample (I)
                .BYTE $40
                .BYTE $FD, $00                           ; sample length duration
                .BYTE $FA, $01                           ; Bank (B=0|1), Index of Sample (I)
                .BYTE $0A
                .BYTE $FF
;***USED**ST****
T7              .BYTE $FD, $01                           ; sample length duration
                .BYTE $FA, $01                           ; Bank (B=0|1), Index of Sample (I)
                .BYTE $0A, $0A
                .BYTE $FA, $02                           ; Bank (B=0|1), Index of Sample (I)
                .BYTE $FD, $02                           ; sample length duration
                .BYTE $40
                .BYTE $FD, $01                           ; sample length duration
                .BYTE $40
                .BYTE $FD, $00                           ; sample length duration
                .BYTE $40
                .BYTE $FD, $01                           ; sample length duration
                .BYTE $40, $40
                .BYTE $FD, $00                           ; sample length duration
                .BYTE $40, $40
                .BYTE $FF

;T8              .BYTE $FD, $03, $FA, $01, $19, $19, $19, $19
;               .BYTE $FF
T8              .BYTE $FD, $03
                .BYTE $FA, $01
                .BYTE C2, C2, C2, C2
                .BYTE $FF
;**NOT DECLARED**
T2912           .BYTE $73, $73, $73, $73, $73, $73, $FA, $00
                .BYTE $16, $16, $49, $49
                .BYTE $FF
;***USED**ST****
T9              .BYTE $FD, $3F                           ; sample length duration
                .BYTE $FA, $10                           ; Bank (B=0|1), Index of Sample (I)
                .BYTE $0A
                .BYTE $FF
;***USED**ST****
T10             .BYTE $FA, $01                           ; Bank (B=0|1), Index of Sample (I)
                .BYTE $FD, $03                           ; sample length duration
                .BYTE $0A, $0A, $0A
                .BYTE $FD, $00                           ; sample length duration
```

```
                .BYTE $0A, $0A, $0A, $0A
                .BYTE $FF
;***USED**ST****
T11             .BYTE $FA, $01                          ; Bank (B=0|1), Index of Sample (I)
                .BYTE $FD, $03                          ; sample length duration
                .BYTE $0A, $0A, $0A
                .BYTE $FD, $00                          ; sample length duration
                .BYTE $FA, $02                          ; Bank (B=0|1), Index of Sample (I)
                .BYTE $40, $40, $40, $40
                .BYTE $FF
;***USED**NT****
T12             .BYTE $FA, $01                          ; select instrument
                .BYTE $FD, $01                          ; select duration
                .BYTE C3, C3, C4, C3, C3, C3, AS2, B2
                .BYTE $FF
;***USED**NT****
T13             .BYTE $FA, $02                          ; select instrument
                .BYTE $FD, $07                          ; select duration
                .BYTE AS4
                .BYTE $FC, $1F                          ; positive portamento
                .BYTE AS4
                .BYTE $FD, $2F                          ; select duration
                .BYTE C5
                .BYTE $FF
;***USED**ST****
T14             .BYTE $FA, $02                          ; Bank (B=0|1), Index of Sample (I)
                .BYTE $FD, $00                          ; sample length duration
                .BYTE $40, $40, $40, $40
                .BYTE $40, $40, $40, $40, $35, $30, $25, $20
                .BYTE $15, $10, $05, $01
                .BYTE $FF
;***USED**NT****
T15             .BYTE $FA, $03                         ; select instrument
                .BYTE $FD, $01                         ; select duration
                .BYTE C4, C4, C4, C4, C4, C4, C4, C4
                .BYTE $FF
;**~USED**ST****
T16             .BYTE $FD, $01                          ; sample length duration
                .BYTE $FA, $01                          ; Bank (B=0|1), Index of Sample (I)
                .BYTE $0A
                .BYTE $FA, $03                          ; Bank (B=0|1), Index of Sample (I)
                .BYTE $FD, $00                          ; sample length duration
                .BYTE $03, $03
                .BYTE $FA, $02                          ; Bank (B=0|1), Index of Sample (I)
                .BYTE $FD, $00                          ; sample length duration
                .BYTE $40
                .BYTE $FA, $03                          ; Bank (B=0|1), Index of Sample (I)
                .BYTE $05, $05
                .BYTE $FA, $02                          ; Bank (B=0|1), Index of Sample (I)
                .BYTE $40
                .BYTE $FA, $03                          ; Bank (B=0|1), Index of Sample (I)
                .BYTE $05
                .BYTE $FA, $02                          ; Bank (B=0|1), Index of Sample (I)
                .BYTE $40
                .BYTE $FA, $03                          ; Bank (B=0|1), Index of Sample (I)
                .BYTE $05
                .BYTE $FA, $01                          ; Bank (B=0|1), Index of Sample (I)
                .BYTE $0A
                .BYTE $FA, $03                          ; Bank (B=0|1), Index of Sample (I)
                .BYTE $05
                .BYTE $FA, $02                          ; Bank (B=0|1), Index of Sample (I)
                .BYTE $40
                .BYTE $FA, $03                          ; Bank (B=0|1), Index of Sample (I)
                .BYTE $05
                .BYTE $FA, $01                          ; Bank (B=0|1), Index of Sample (I)
                .BYTE $FD, $01                          ; sample length duration
                .BYTE $0A
                .BYTE $FF
;***USED**NT****
T17             .BYTE $FA, $0D                          ; select instrument
                .BYTE $FD, $1B                          ; select duration
                .BYTE $38
                .BYTE $FD, $03                          ; select duration
                .BYTE $3A
                .BYTE $FD, $05                          ; select duration
                .BYTE $3B, $3A
                .BYTE $FD, $03                          ; select duration
                .BYTE $38
                .BYTE $FD, $07                          ; select duration
                .BYTE $36
                .BYTE $FD, $03                          ; select duration
                .BYTE $35, $36
                .BYTE $FF

EMPTY1          .BYTE $00,$FF
EMPTY2          .BYTE $00,$FF
EMPTY3          .BYTE $00,$FF
EMPTY4          .BYTE $10,$FF
```

```
T18             .BYTE $FA, $01
                .BYTE $FD, $01
                .BYTE $19, $25, $19, $19, $25, $19, $19, $18
                .BYTE $14, $20, $14, $14, $20, $14, $14, $13
                .BYTE $17, $23, $17, $17, $23, $17, $17, $16
                .BYTE $12, $1E, $12, $12, $1E, $12, $12, $11
                .BYTE $FF
;***UNUSED**NT**
T19             .BYTE $FA, $0D                          ; select instrument
                .BYTE $FD, $03                          ; select duration
                .BYTE C5, G4, E4, C4, D4, D4, D4, C5
                .BYTE AS4, AS4, AS4, A4, A4, A4, AS4, AS4
                .BYTE $FF

T20             .BYTE $FA, $01
                .BYTE $FD, $01
                .BYTE $19, $25, $19, $19, $25, $19, $19, $1B
                .BYTE $14, $20, $14, $14, $20, $14, $14, $16
                .BYTE $17, $23, $17, $17, $23, $17, $17, $16
                .BYTE $12, $1E, $12, $12, $1E, $12, $15, $16
                .BYTE $FF
;***UNUSED**NT**
T21             .BYTE $FA, $0A                          ; select instrument
                .BYTE $FD, $0F                          ; select duration
                .BYTE C4
                .BYTE $FA, $0B                          ; select instrument
                .BYTE D4
                .BYTE $FA, $0E                          ; select instrument
                .BYTE F4
                .BYTE $FA, $0A                          ; select instrument
                .BYTE F4
                .BYTE $FF
;***UNUSED**NT**
T22             .BYTE $FA, $0F                          ; select instrument
                .BYTE $FD, $01                          ; select duration
                .BYTE C4, C4, C4, C4, A3, C4, A3, C4
                .BYTE $FA, $05                          ; select instrument
                .BYTE $FD, $2B                          ; select duration
                .BYTE $00
                .BYTE $FA, $0F                          ; select instrument
                .BYTE $FD, $01                          ; select duration
                .BYTE D4, D4
                .BYTE $FF

T23             .BYTE $FA, $10
                .BYTE $FD, $0F
                .BYTE $36,
                .BYTE $FA, $11
                .BYTE $36
                .BYTE $FA, $12
                .BYTE $FD, $1F
                .BYTE $34
                .BYTE $FF

T24             .BYTE $FA, $01
                .BYTE $FD, $03
                .BYTE $1A, $1A, $1A, $1A, $1E, $1E, $1E, $1E
                .BYTE $19, $19, $19, $19, $19, $19, $19, $19
                .BYTE $FF
;***UNUSED**NT**
T25             .BYTE $FA, $01                          ; select instrument
                .BYTE $FD, $0B                          ; select duration
                .BYTE $25
                .BYTE $FD, $03                          ; select duration
                .BYTE $FB, $29                          ; negative portamento
                .BYTE $25
                .BYTE $FD, $0B                          ; select duration
                .BYTE $1F
                .BYTE $FD, $03                          ; select duration
                .BYTE $FC, $29                          ; positive portamento
                .BYTE $1F
                .BYTE $FD, $0B                          ; select duration
                .BYTE $25
                .BYTE $FD, $03                          ; select duration
                .BYTE $FB, $29                          ; negative portamento
                .BYTE $25
                .BYTE $FD, $0B                          ; select duration
                .BYTE $1F
                .BYTE $FD, $03                          ; select duration
                .BYTE $FC, $29                          ; positive portamento
                .BYTE $1F
                .BYTE $FF
;***UNUSED**NT**
T26             .BYTE $FA, $01                          ; select instrument
                .BYTE $FD, $0B                          ; select duration
                .BYTE $19
                .BYTE $FD, $03                          ; select duration
```

```
                .BYTE $FB, $14                          ; negative portamento
                .BYTE $19
                .BYTE $FD, $0B                          ; select duration
                .BYTE $13
                .BYTE $FD, $03                          ; select duration
                .BYTE $FC, $14                          ; positive portamento
                .BYTE $13
                .BYTE $FD, $0B                          ; select duration
                .BYTE $19
                .BYTE $FD, $03                          ; select duration
                .BYTE $FB, $14                          ; negative portamento
                .BYTE $19
                .BYTE $FD, $0B                          ; select duration
                .BYTE $13
                .BYTE $FD, $03                          ; select duration
                .BYTE $FC, $14                          ; positive portamento
                .BYTE $13
                .BYTE $FF
;***UNUSED**NT**
T27             .BYTE $FA, $0D                          ; select instrument
                .BYTE $FD, $03                          ; select duration
                .BYTE C5, C5, C5
                .BYTE $FD, $01                          ; select duration
                .BYTE AS4, C5
                .BYTE $FD, $05                          ; select duration
                .BYTE CS5
                .BYTE $FD, $03                          ; select duration
                .BYTE DS5
                .BYTE $FD, $05                          ; select duration
                .BYTE CS5
                .BYTE $FD, $03                          ; select duration
                .BYTE FS4, FS4, FS4
                .BYTE $FD, $01                          ; select duration
                .BYTE F4, FS4
                .BYTE $FD, $05                          ; select duration
                .BYTE F4
                .BYTE $FD, $03                          ; select duration
                .BYTE C4
                .BYTE $FD, $05                          ; select duration
                .BYTE F4
                .BYTE $FF

T28             .BYTE $FA, $08
                .BYTE $FD, $0F
                .BYTE $19, $1A, $13, $12
                .BYTE $FF

;-------------------------------------------------------------------------------
                .TEXT   '(C)1987 MATT GRAY'             ; CHANGED FROM .BYTE TO .TEXT
;-------------------------------------------------------------------------------

SETNMI
                JSR SETIRQ
                LDA #$00
                STA $DD0E               ; Control Register A
                LDA #NMI&255
                STA $0318               ; set NMI low
                LDA #NMI/256
                STA $0319               ; set NMI high

                LDA #$01
                STA $DD04               ; Timer A Low-Byte  (RS232)
                LDA #$01
                STA $DD05               ; Timer A High-Byte (RS232)
                LDA #$11
                STA $DD0E               ; Control Register A
                LDA #$81
                STA $DD0D               ; Interrupt (NMI) Control Register
                LDA $DD0D               ; Interrupt (NMI) Control Register
                RTS

NMI             PHA
                TXA
                PHA
                TYA
                PHA
                LDA #$7F
                STA $DD0D               ; Interrupt (NMI) Control Register
                JSR SAMPLEROUTINE
                LDA #$81
                STA $DD0D               ; Interrupt (NMI) Control Register
                LDA $DD0D               ; Interrupt (NMI) Control Register
                JMP $EA81               ; Restore A/X/Y and End IRQ


SAMPLEROUTINE   LDA SAMPLEVOL           ; 0=volume to middle
                BNE GETSAMPLE
                RTS
```

```
LOWNIBBLE
                LDA #$00
                STA NIBBLE              ; set for high nibble

                PLA
                AND #$0F                ; get low nibble
                JMP OUTSAMPLE

GETSAMPLE
MEMPOINT        LDA $1412               ; read two samples
                PHA
                LDA NIBBLE
                BNE LOWNIBBLE

                LDA #$01
                STA NIBBLE              ; set for low nibble

                PLA
                AND #$F0
                LSR A
                LSR A
                LSR A
                LSR A                   ; get high nibble

OUTSAMPLE       STA $D418               ; Select Filter Mode and Volume
                LDA NIBBLE
                BNE EXITSAMPLE

                INC MEMPOINT+1          ; inc low address pointer
                BNE EXITSAMPLE
                INC MEMPOINT+2          ; inc high address pointer
                LDA MEMPOINT+2          ; load high address pointer
ENDAREA         CMP #$16                ; end of sample area?
                BCC EXITSAMPLE

SETSAMPLEVOL    LDA #$00
                STA SAMPLEVOL           ; 0=volume to middle

                LDA #$00
                STA MEMPOINT+1          ; set low address pointer
MEMHIGH         LDA #$14
                STA MEMPOINT+2          ; set high address pointer

EXITSAMPLE      RTS


SAMPLEVOL       .BYTE $14               ; 0=volume to middle
NIBBLE          .BYTE $01               ; select Nibble flag (1=low, 0=high)
                .BYTE $00
SAMPLESPEED     .BYTE $0A                ; sample speed (low byte)
                .BYTE $00
V4LO            .BYTE $7B               ; current track 4 (sample) position (base low)
V4HI            .BYTE $28               ; current track 4 (sample) position (base high)
SAMPLEBEATCOUNT .BYTE $00               ; pattern index of sample voice
SAMPLEBARCOUNT  .BYTE $13               ; actual track position (offset - bar counter of sample)
SAMPLEACTDUR    .BYTE $00               ; actual sample duration
SAMPLEDURATION  .BYTE $00               ; sample length duration
SAMPLEBANK      .BYTE $00               ; sample bank (0/1)
SAMPLEINDEX     .BYTE $01               ; index of sample in table

BANKLO          .BYTE $00,      $00,      $00,      $00,      $FF,      $00,      $00       ; low bank mempoint
BANKHI0         .BYTE SM10/256, SM14/256, SM16/256, SM18/256, SM13/256, SM12/256, SM1C/256  ; high bank 0 mempoint
BANKEND         .BYTE SM13/256, SM16/256, SM18/256, SM1A/256, SM14/256, SM13/256, SM1E/256  ; end sample high address
BANKHI1         .BYTE SM12/256, SM15/256, SM17/256, SM19/256, SM19/256, SM12/256, SM1D/256  ; high bank 1 mempoint

MAINEXTRA       LDA TN                  ; song (track) number
                BNE PLAYSAMPLE

                STA $D418               ; Select Filter Mode and Volume (to 0)
                STA SAMPLEVOL           ; 0=volume to middle
                RTS

PLAYSAMPLE      LDA SPEED               ; actual cycle timer (speed of song)
                BNE GOEXIT

                DEC SAMPLEACTDUR        ; actual sample duration
                BMI GETSAMPLETRK

GOEXIT          JMP QUIT2

GETSAMPLETRK    LDA V4LO                ; get current track 4 (sample) position (base low)
                STA POINTS              ; track pattern pointer (low)
                LDA V4HI                ; get current track 4 (sample) position (base high)
                STA POINTS+1            ; track pattern pointer (high)

                LDY SAMPLEBARCOUNT      ; read actual track position (offset - bar counter of sample)
                LDA (POINTS),Y          ; read sample pattern index to use
```

```
                TAY

                LDA BARLO,Y             ; read pattern pointer low
                STA BARS                ; pattern pointer low
                LDA BARHI,Y             ; read pattern pointer high
                STA BARS+1              ; pattern pointer high

GETSAMPLECMD    LDY SAMPLEBEATCOUNT      ; load pattern index of sample voice
                LDA (BARS),Y
                CMP #$FD                 ; set sample length duration?
                BCC TESTSAMPLEFA

                INY
                INC SAMPLEBEATCOUNT     ; next pattern index of sample voice
                LDA (BARS),Y            ; sample length duration
                STA SAMPLEDURATION      ; store sample length duration

INCSAMPLE       INC SAMPLEBEATCOUNT     ; next pattern index of sample voice
                BNE GETSAMPLECMD

TESTSAMPLEFA    CMP #$FA                ; set Bank (B=0|1), Index of Sample (I) ?
                BCC SETSAMPLESPEED

                INY
                INC SAMPLEBEATCOUNT     ; next pattern index of sample voice
                LDA (BARS),Y
                PHA
                AND #$0F                ; takes low part (index of sample)
                STA SAMPLEINDEX         ; index of sample in table
                PLA
                AND #$F0                ; takes high part (bank 0|1)
                STA SAMPLEBANK
                JMP INCSAMPLE

SETSAMPLESPEED  STA SAMPLESPEED         ; sample speed (low byte)
                LDY SAMPLEINDEX         ; index of sample in table
                LDA BANKLO,Y            ; mempoint low bank address
                STA MEMPOINT+1
                LDA BANKHI0,Y           ; mempoint high bank 0 address
                STA MEMPOINT+2

                LDA BANKEND,Y           ; mempoint high bank end address
                STA ENDAREA+1           ; ending area of sample

                LDA SAMPLEBANK
                BEQ ISBANK0

                LDA BANKHI1,Y           ; mempoint high bank 1 address
                STA MEMHIGH+1
                STA SETSAMPLEVOL+1
                BNE SKIPBANK0

ISBANK0         STA SETSAMPLEVOL+1

                LDA BANKHI0,Y           ; mempoint high bank 0 address
                STA MEMHIGH+1

SKIPBANK0       STA SAMPLEVOL           ; 0=volume to middle

                LDA SAMPLESPEED         ; sample speed (low byte)
                STA $DD04               ; Timer A Low-Byte  (RS232)

                LDA SAMPLEDURATION      ; read sample length duration
                STA SAMPLEACTDUR        ; actual sample duration

                INC SAMPLEBEATCOUNT     ; next pattern index of sample voice
                LDY SAMPLEBEATCOUNT     ; load pattern index of sample voice
                LDA (BARS),Y

                CMP #$FF                ; end?
                BNE QUIT3

                LDA #$00
                STA SAMPLEBEATCOUNT     ; reset pattern index of sample voice

                INC SAMPLEBARCOUNT      ; inc actual track position (offset - bar counter of sample)
                LDY SAMPLEBARCOUNT      ; read actual track position (offset - bar counter of sample)
                LDA (POINTS),Y
                CMP #$FF                ; restart
                BNE TESTEND

                LDA #$00
                STA SAMPLEBARCOUNT      ; set track position (offset - bar counter of sample) to the beginning
                BEQ QUIT3

TESTEND         CMP #$FE                ; end
                BNE QUIT3
```

```
                LDA #$00
                STA TN                  ; song (track) number
                RTS

QUIT3           JMP QUIT2

;***USED**ST****
T29             .BYTE $FD, $00                                  ; sample length duration
                .BYTE $FA, $03                                  ; Bank (B=0|1), Index of Sample (I)
                .BYTE $01, $02, $03, $04, $05, $06, $07, $08
                .BYTE $09, $0A, $0B, $0C, $0D, $0E, $0F, $10
                .BYTE $11, $12, $13, $14, $15, $16, $17, $18
                .BYTE $19, $1A, $1B, $1C, $1D, $1E, $1F, $20
                .BYTE $21, $22, $23, $24, $25, $26, $27, $28
                .BYTE $29, $2A, $2B, $2C, $2D, $2E, $2F, $30
                .BYTE $31, $32, $33, $34, $35, $36, $37, $38
                .BYTE $39, $3A, $3B, $3C, $3D, $3E, $3F, $40
                .BYTE $FF

; Startup
STARTUP
                LDA #$01            ; start with first song
                STA TN              ; song (track) number
                JMP SETNMI
```

Conclusion

If the analysis we have seen that sample logic where added like and extension of the three normal patterns used for the 3 sid voices.

The only secret is that you have to use a NMI routine over the logic governed by the IRQ to performs the sample generation.

The other trick is to store the 4 bit value of each sample in one byte (8 bits), so you have to read the low and then high nibble to have the two samples.

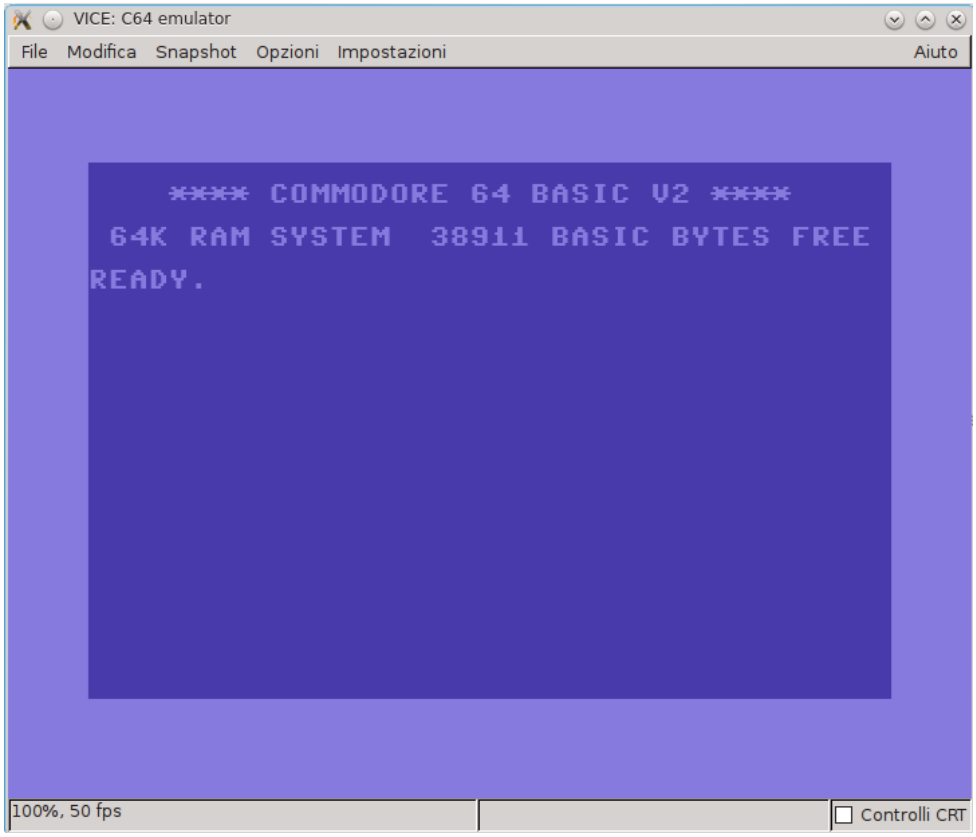
At this point we have seen 3 Matt Gray engines, that has to be read in this sequences (as features maturated from one to another):

- Driller
- Serpent Demo
- Dominator

So....why not see the last one???
The one use in Vendetta that use filter???

Well....not now, …. maybe in another issue ;)

At the beginning (well, many years ago) we had two kind of distinct products: a C64 emulator and a SID player.



With the C64 emulator you can play a game or run a demo, while with the SID player you can listen only to PSID/RSID music (that is a convenient way to store the code and data that can be used for reproducing the music).

But, if you look carefully all the two products have to emulate the C64 internal chips for being able to execute a C64 program. Even the emulation of the internal memories (like Kernal and Basic) could be achieved by the two products.

For a SID tune you did not need to emulate the VIC II chip (as the PSID format takes some information that allow to bypass for example the Vic rasterline IRQ), unless you want to be able to play some tunes that cannot be packed inside a PSID file.

Else you need to emulate the Basic rom if you want to be able to listen to Basic based music in the sid player.

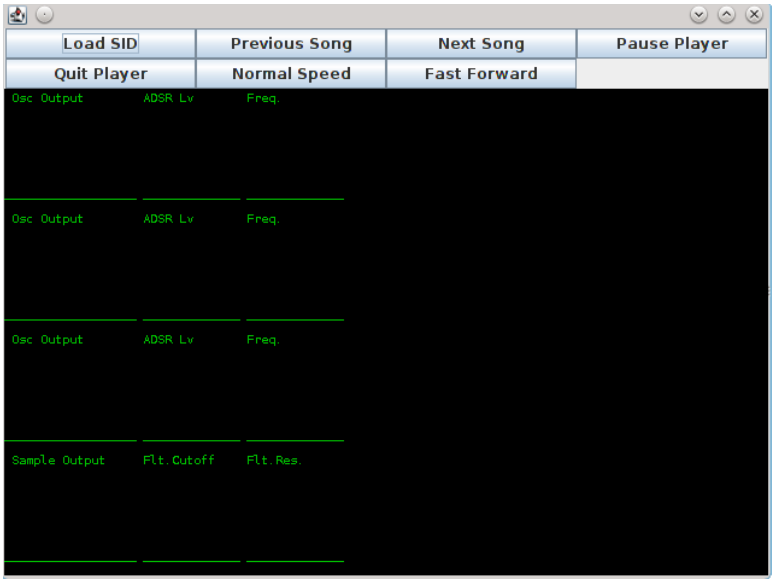
Instead the C64 emulator can even play PSID music if some hooks are used for using the information inside the PSID file that bypass some C64 internal process.

So we have emulator like Vice that can be used as sidplayer by using the VSID program, or Sidplayer that can be used as a C64 emulator like JSIDPlay2.

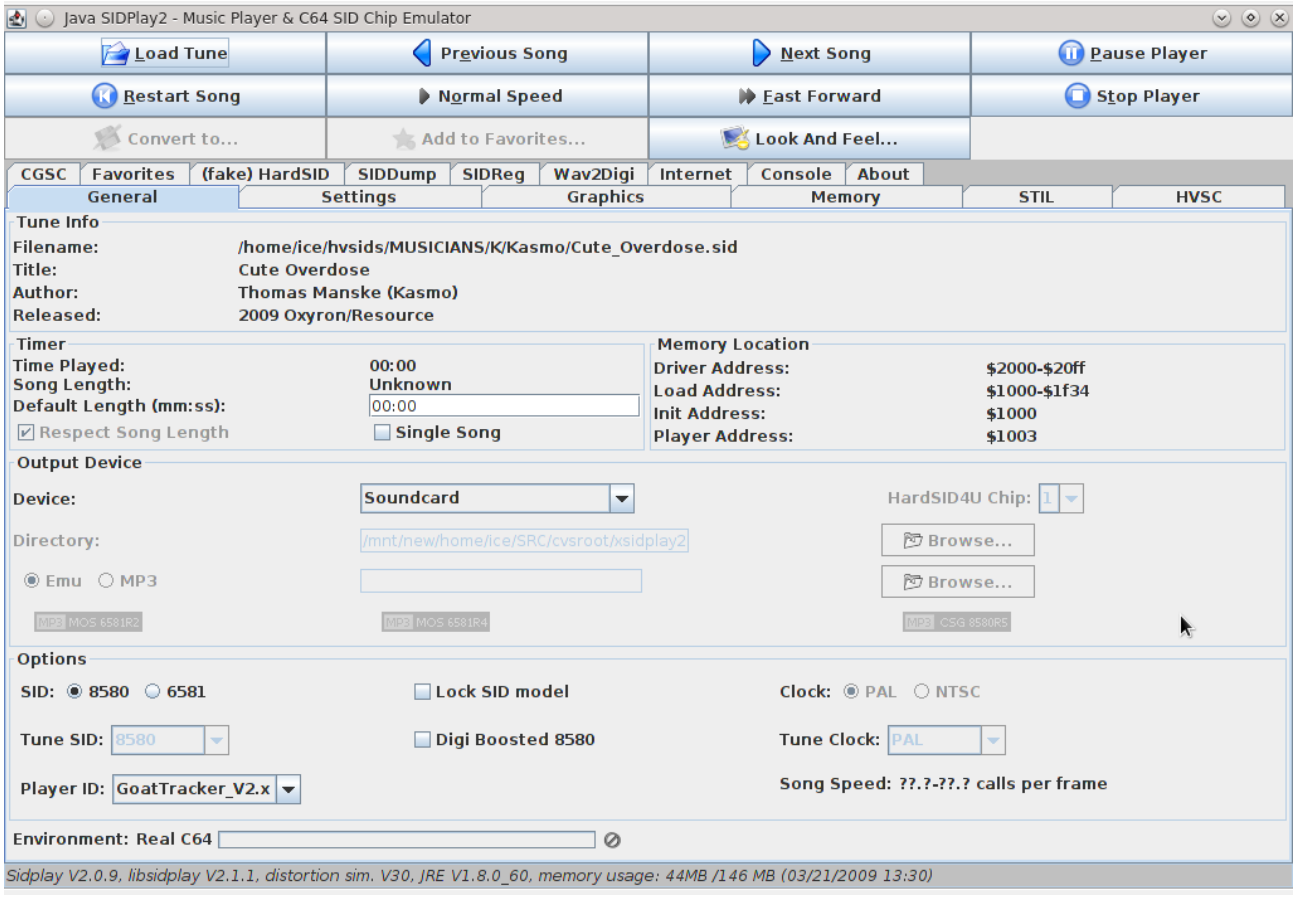
Beginning

At the beginning JSIDPlay2 was only a sid player made in Java that base his code in the porting of libsidplay2 C++ sound library.

The below screenshot is of an initial version of it (it did not show his version):



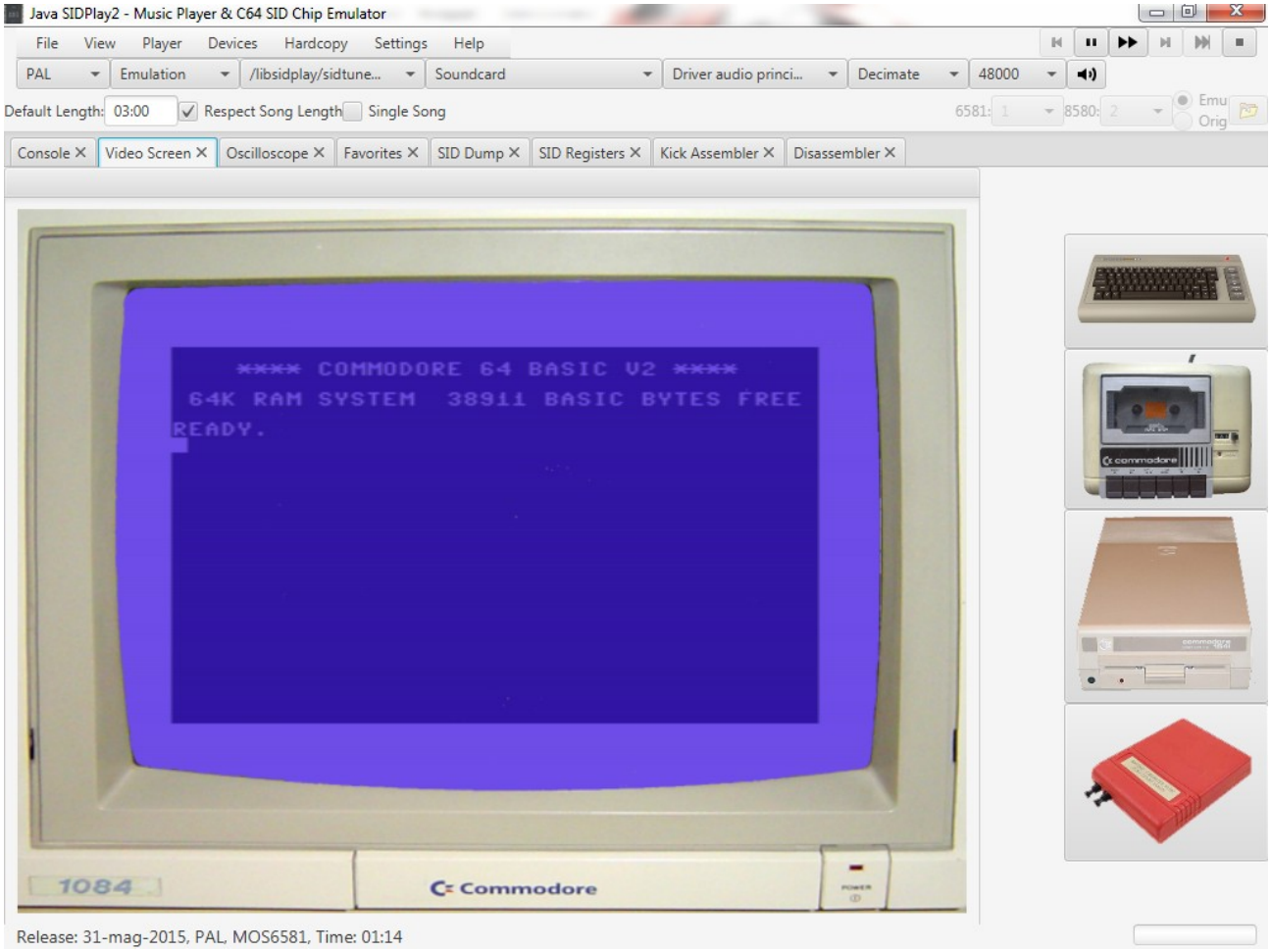
And this is from version 2.0.9 where it is more advanced but no like now:



Actually

Now, JSidplay2 3.6 is near to be a C64 complete emulator, that let you listen to SID music inside game or demo too.

Lets look at is when opened:



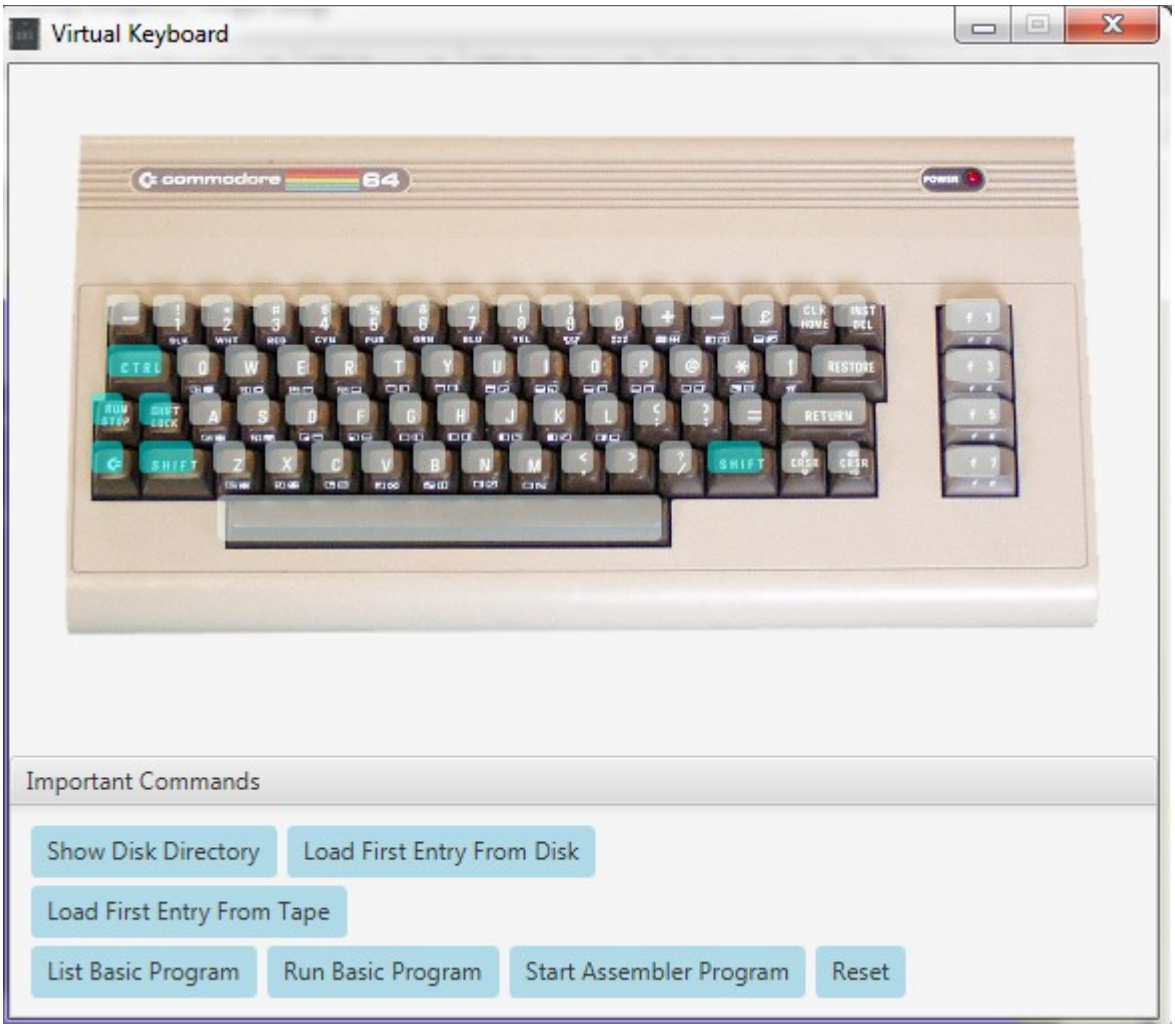
Amazing!

You are in front a C64 with all his peripherals:

- Old model C64
- Datassette
- Floppy drive
- Cartridges
- 1084 C= Monitor

In the monitor you can see what is actually emulated by the C64.

In fact you are inside the **"Video Screen"** tabs and you can interact with the peripherals you see in that tabs, for example you can take up the C64 keyboard directly:



Here you have even some commands in buttons that executes some common instructions to the emulated C64.

But lets as continue with the analysis of this program.

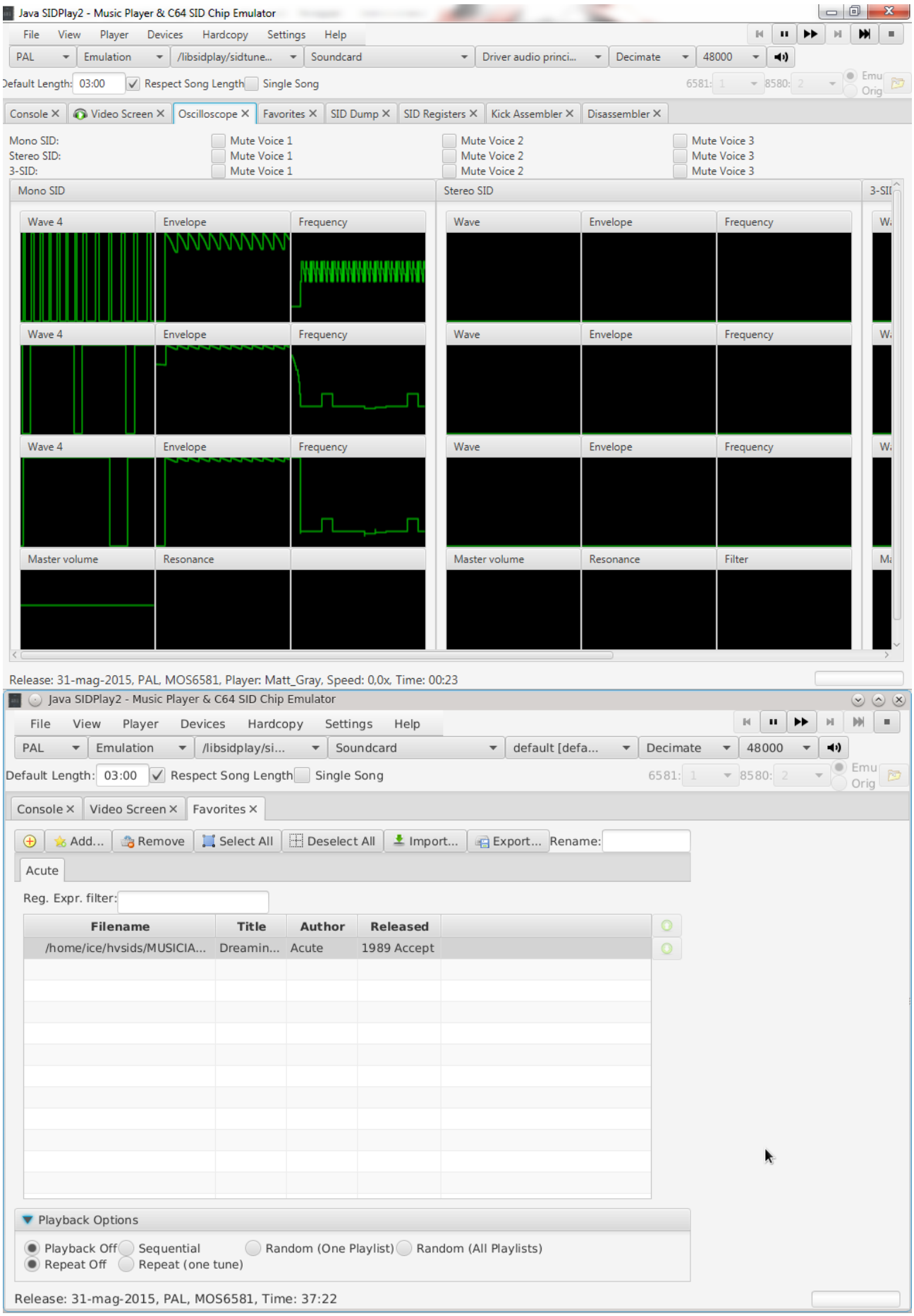
The **"Console"** tabs is just a big text box where you obtains texture messages from the program (divided for standard output and error output).

One of the advantage of using JSiplay2 is that you can use it without graphics thrown console invocation and even use his engine in other program.

For example I use it inside JITT64 tracker, by calling the appropriate class. In fact the JSidplay2 program is formed by more that 40 Java Jar packages (and some DLL for low lever functions for accessing input devices).

Going in the next tab, we are into "**Oscilloscope"**.

In this tab we can see in real time the *Wave*, *Envelope* and *Frequency* of each SID voices (there is one even for sample that has *Master Volume*, *Resonance* and *Filters*). This is available for one to three SID chip, depending from the music we are listening.

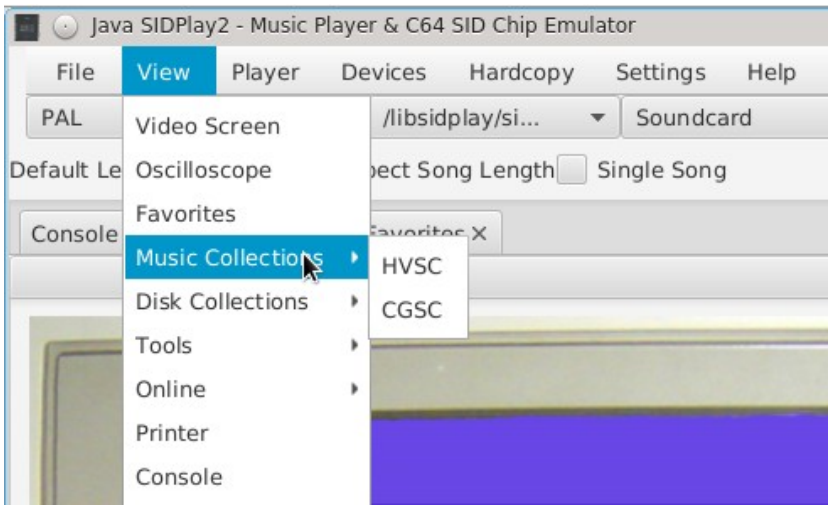

Java SIDPlay2 - Music Player & C64 SID Chip Emulator
File View Player Devices Hardcopy Settings Help
PAL Emulation /libsidplay/sidtune... Soundcard Driver audio princi... Decimate 48000
Default Length: 03:00 Respect Song Length Single Song
Console Video Screen Oscilloscope Favorites SID Dump SID Registers Kick Assembler Disassembler
Mono SID:
Stereo SID:
3-SID:
Mute Voice 1
Mute Voice 2
Mute Voice 3
Mono SID
Stereo SID
Wave 4
Envelope
Frequency
Wave
Master volume
Resonance
Filter
Release: 31-mag-2015, PAL, MOS6581, Player: Matt_Gray, Speed: 0,0x, Time: 00:23
Java SIDPlay2 - Music Player & C64 SID Chip Emulator
File View Player Devices Hardcopy Settings Help
PAL Emulation /libsidplay/si... Soundcard default [defa... Decimate 48000
Default Length: 03:00 Respect Song Length Single Song
Console Video Screen Favorites
Add... Remove Select All Deselect All Import... Export... Rename:
Acute
Reg. Expr. filter:
Filename Title Author Released
/home/ice/hvsids/MUSICIA... Dreamin... Acute 1989 Accept
Playback Options
Playback Off Sequential Random (One Playlist) Random (All Playlists)
Repeat Off Repeat (one tune)
Release: 31-mag-2015, PAL, MOS6581, Time: 37:22

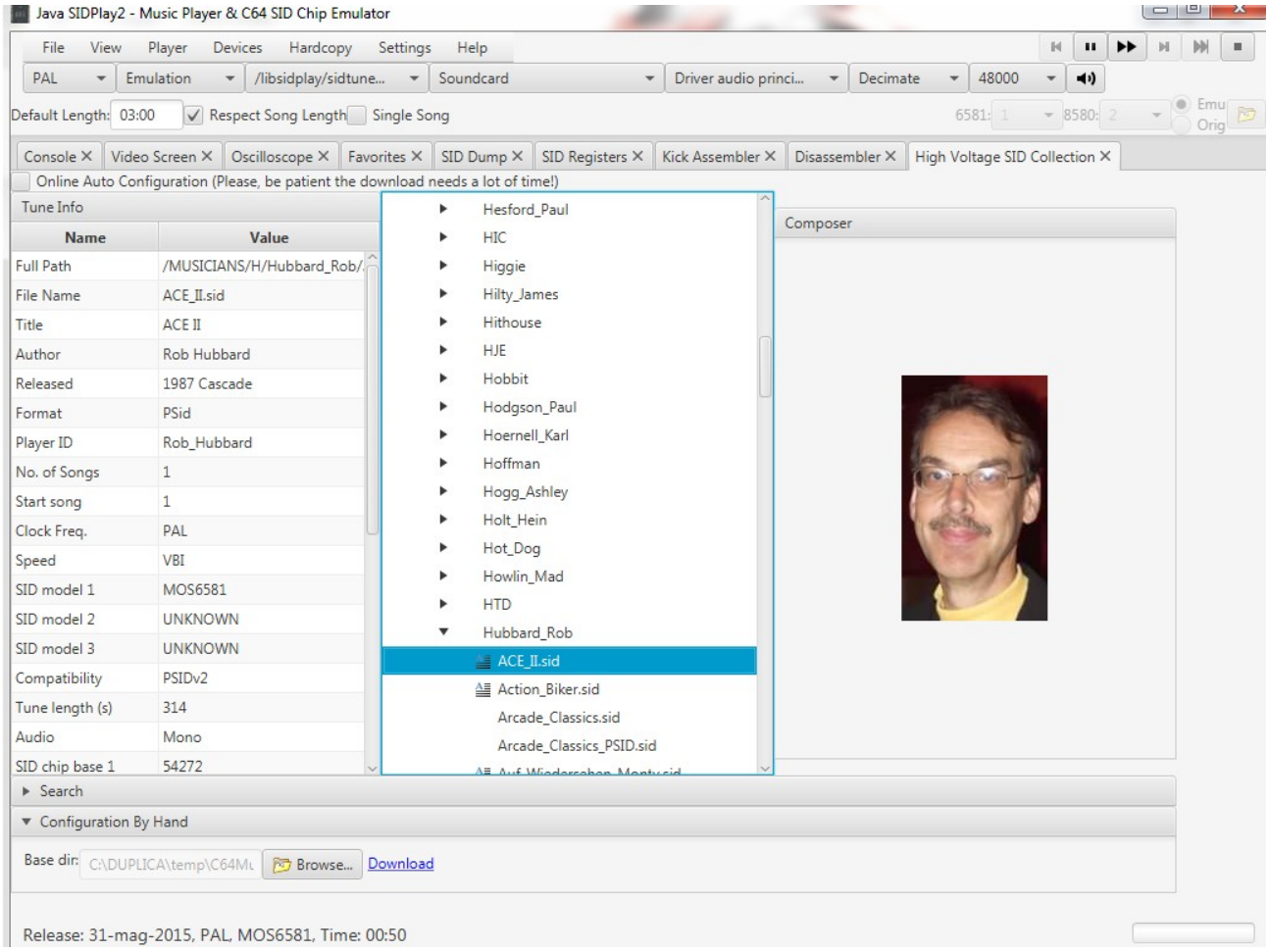
Another tab is **“Favorities”.**

This is the classical function of a Sid Player to add a playlist of tunes and let choose how to (random) play them.

But the best features is that you can add many playlists (using the + button, then a new tab appears) and then choose even to play random from all the playlists.

Looking at the menu, now two tabs can be opened: HVSC and CGSC music collections:





With the **“High Voltage Music Collection”** or **“Compute's Gazette Sid Collection”** tab, you can manage your copy of HVSC or CGSC locally or letting it download automatically from the web.

The tab has 3 blocks:

- The information about the selected tune inside HVSC/CGSC
- The directories/files structures of HVSC/CGSC
- The photo (if available) of the author of the Sid tune.

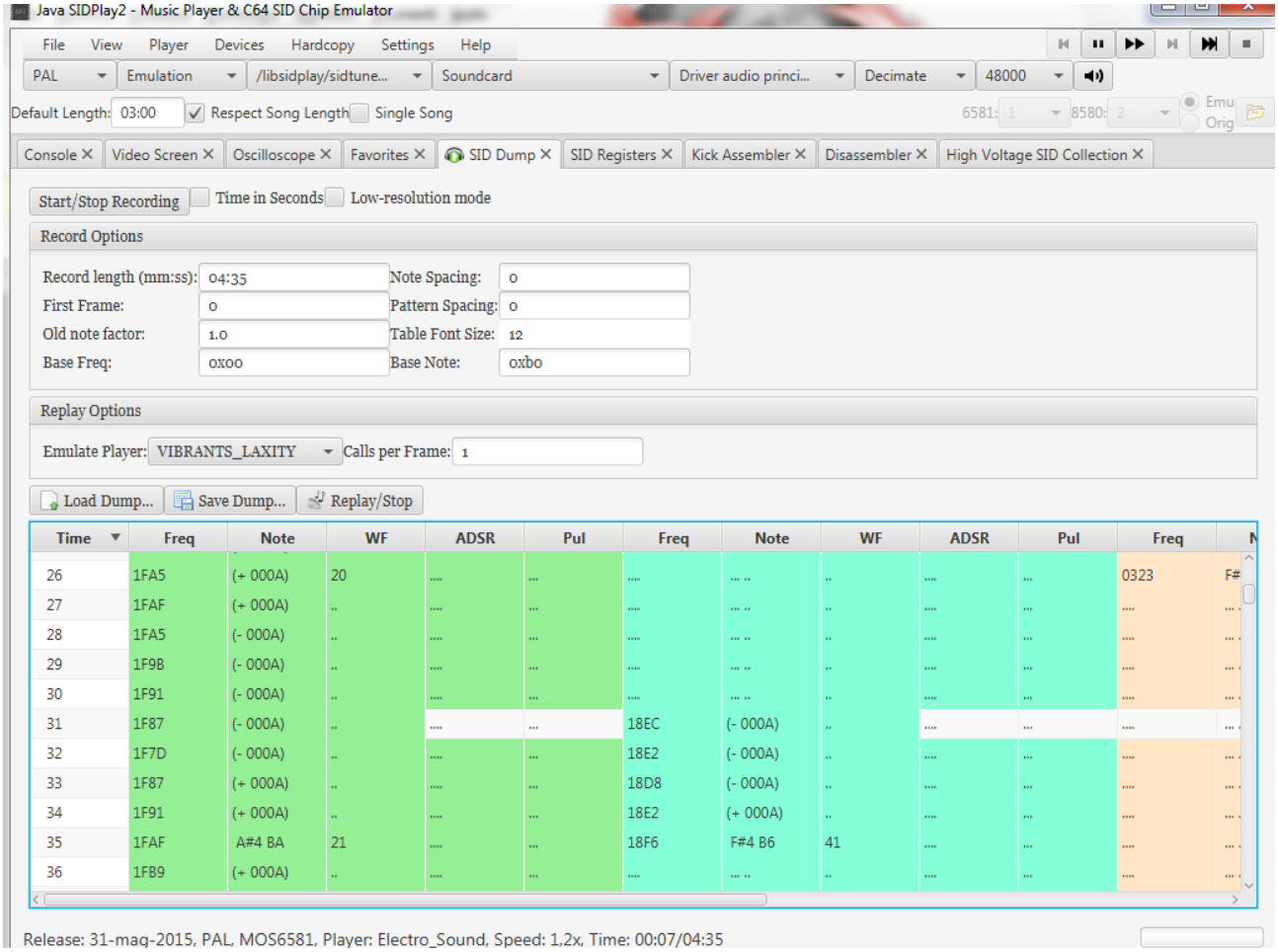
Another 3 tabs are possible to add and they are related to Disk Collections:

- High Voltage Music Engine Collections
- Demo
- Magazine

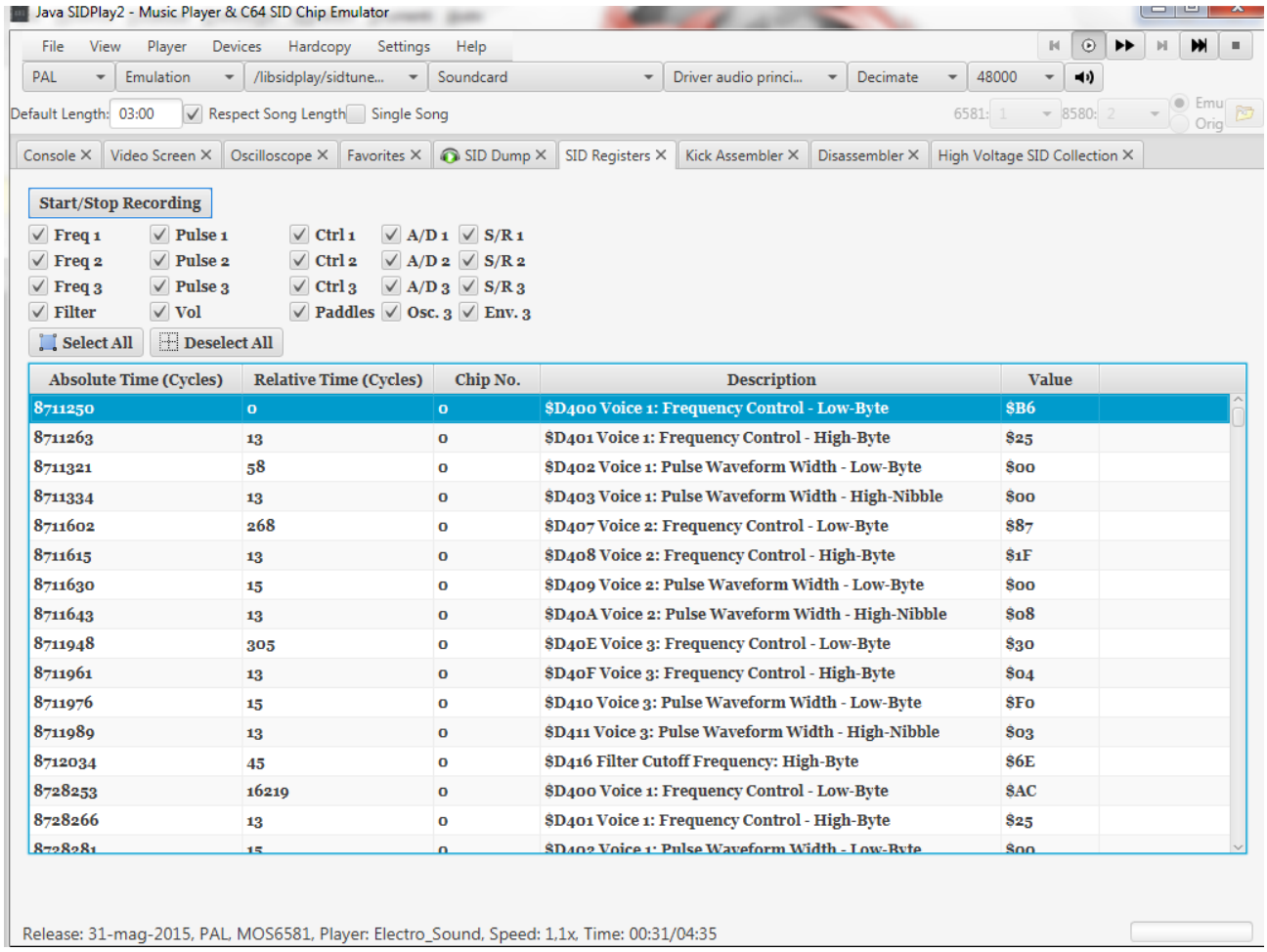
Each of this has the possibility to automatically download the stuff.

Other tabs that can be interesting for a programming view are related to the sid registers.

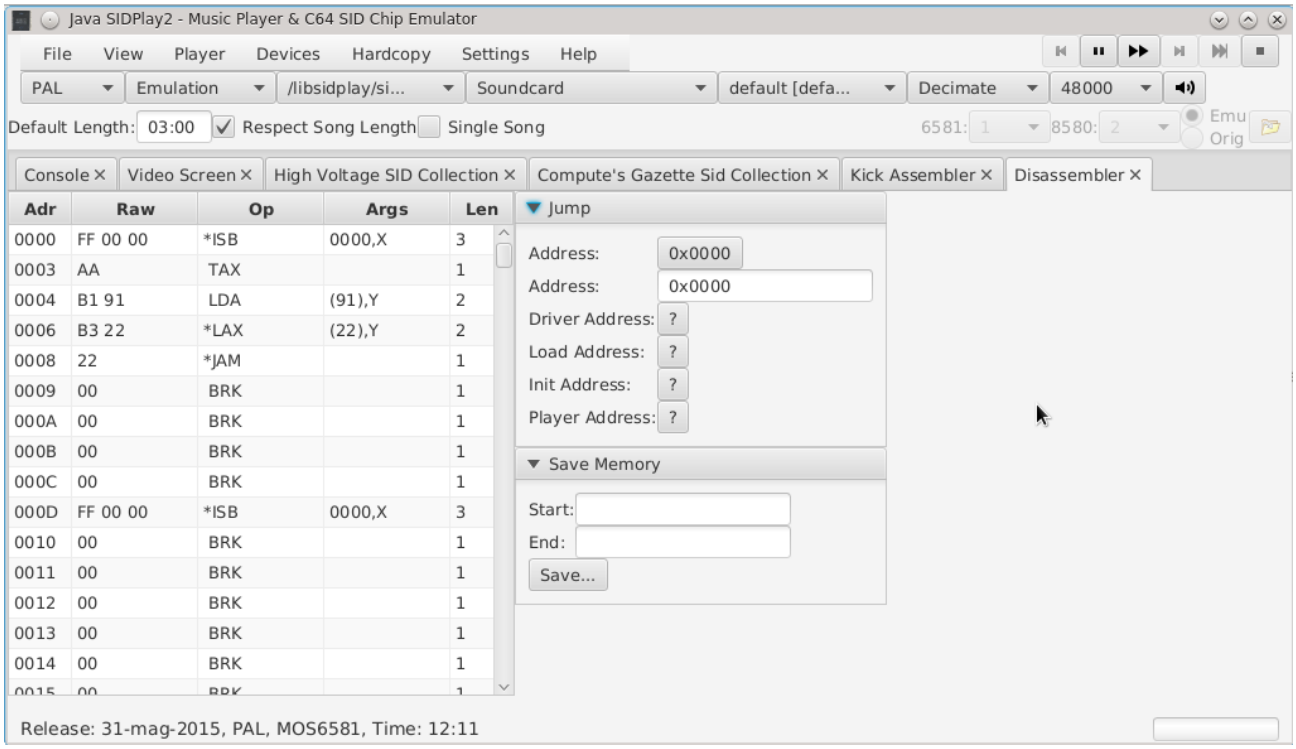
With **“SID Dump”** tab you have access to all sid register manipulated by the tune in real-time. That list can be exported for analyze it (I myself use it for testing JITT64 instruments and see that they play as expected).

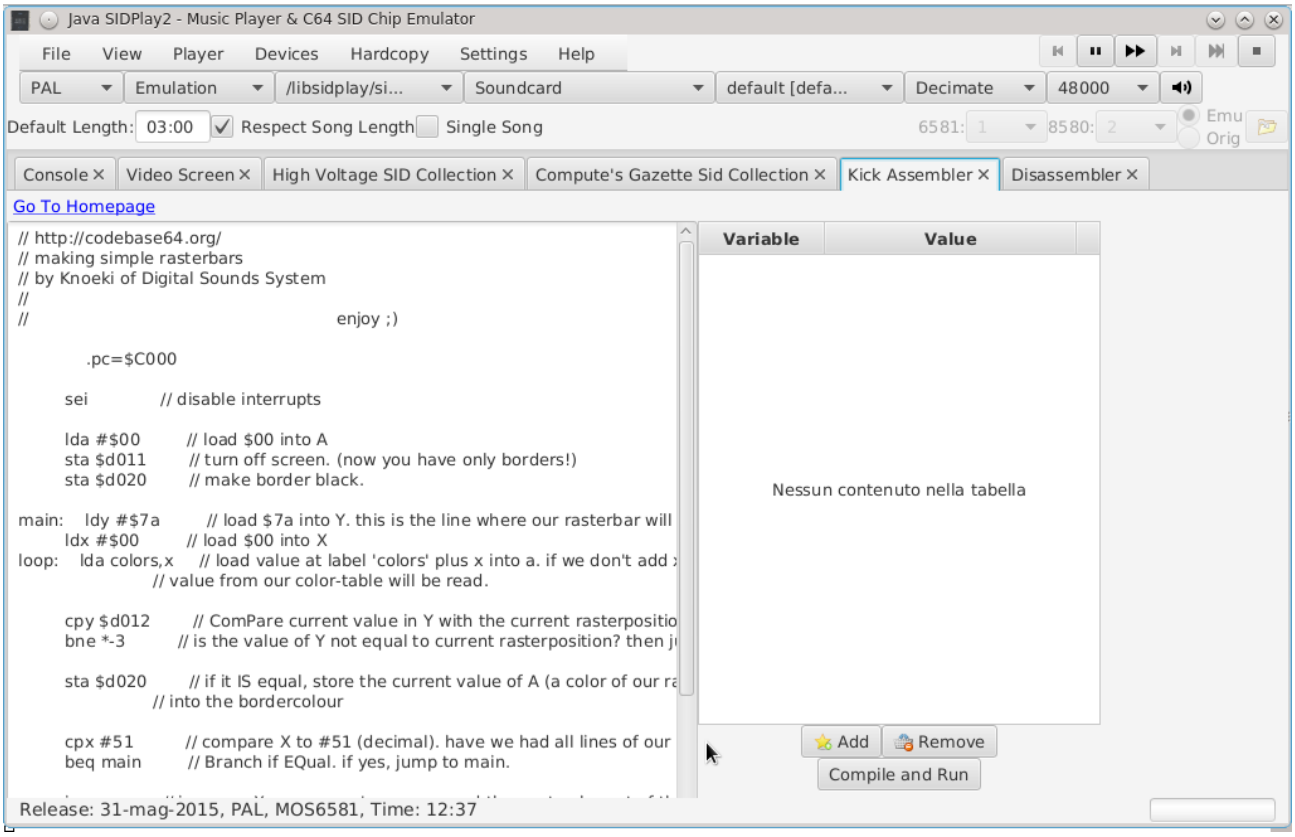


Instead the **“SID Registers”** tab is another way to have a dump of SID at event level with the possibility to filter what to see.

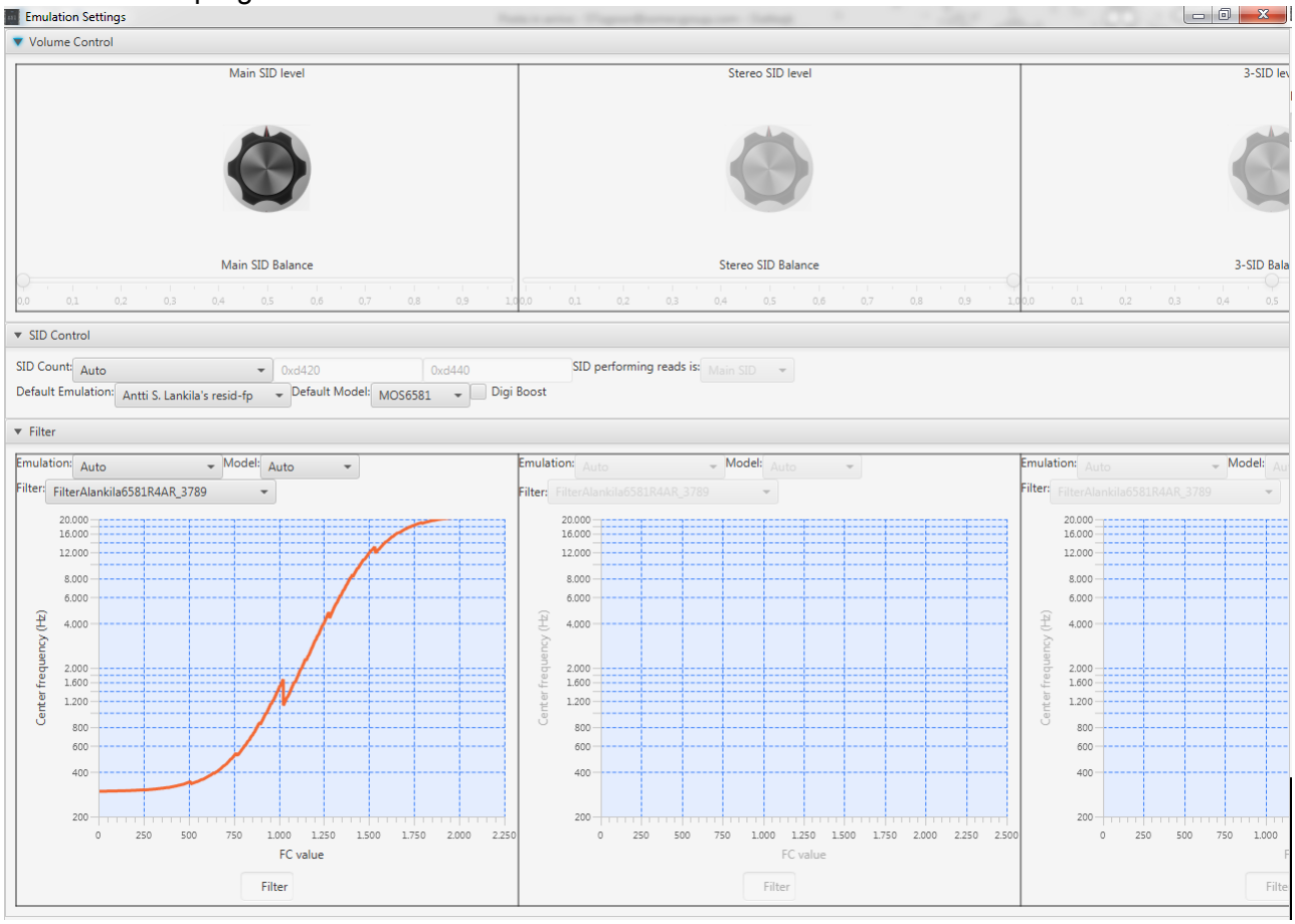


In the same Tools menu, you can now select **“Kick Assembler”** and **“Disassembler”** tab.





In one you have a disassembler that can be useful for look at the instructions of the program/music you are listening, while the other is a little assembler that you can use for build and run a little program.



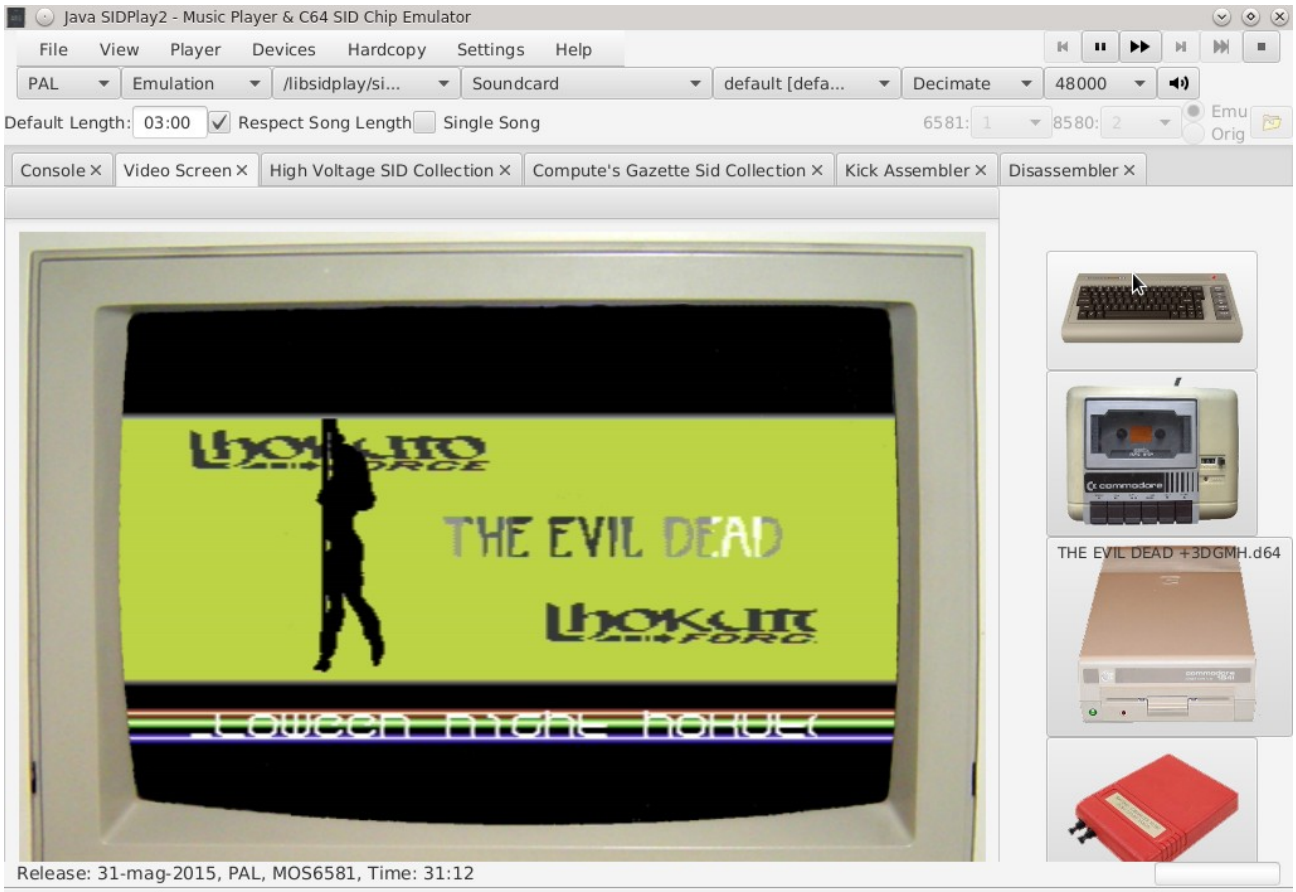
One interesting features is the possibility to change the emulation engines used by the player, as you can use resid 1.0 or resid-fp with lot of different filter response.

Other interesting features is the ability to play and generate a WAV or MP3 at the same time of what you are listening.

Conclusion

JSidPlay2 is a great software that start from a simple sid player and become an almost complete C64 emulator.

For example this is the “The Evil Dead” by Hokuto Force released just today that I'm listening inside JSidPlay2!!



However actually JSidPlay2 miss some emulator facilities like run it at the max velocity (there is only a fast forward) or to delay it like you can do in Vice, or speed up disk read, but all the program I test are running fine.

The is only a note for Linux users: you need to use official Oracle Java 1.8 to run it, as OpenJDK is not able to pick up all the stuff and execute it.

Download it from: https://sourceforge.net/projects/jsidplay2/

Good Halloween to all.

Din 15 end