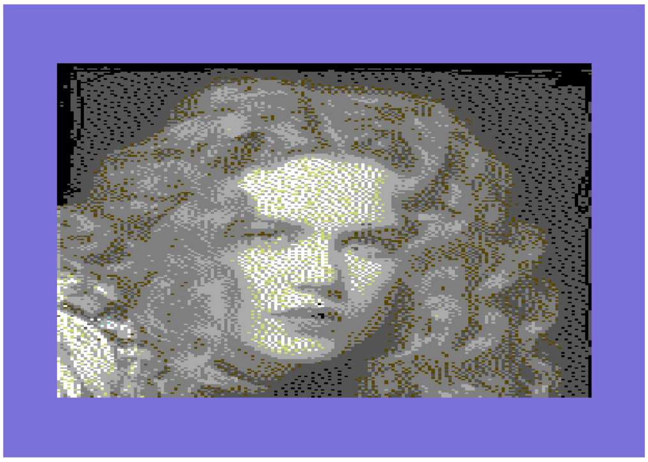
SIDin 3



“Nicole Kidman”

Vice snapshot with new palette

Made with the GIMP from a NK photo
and converted to C64 160x200
Multicolor Mode Bitmap
by Stefano Tognon
in 2002

“Sid Music forever”
...

Free Software Group

SIDin 3
version 1.00
7 February 2003

SIDin Contents

General Index

Editorials

Stefano Tognon <ice00@libero.it>

Hi, again.

In this issue of SIDin I present two soft articles even if this number is a bit in delay with the planned release date. Ok, there was the holiday, but the most reason is the too much activity I'm in in this period.

Don't worry, I'm working in a great reverse engineering work about Arkanoid tune, that can be probably released in the next issue.

Instead, if you were asking with witch music player a given Sid tune was made, now we are working in this direction; however your help will be very appreciated.

I let you know how it is possible to achieve this by pattern searching, and next time with a much great project!

The second article is a little guide of how to rip BASIC based music; I rip a easy kind of BASIC music, but there are more complex BASIC games that attends that one rip them.

Remember that the right of the presented code remain to Andrew Colin and Commodore.

Before give you an appointment for the next issue, remember that if you want to have your preferred music engine reverse engineered by me, please send me an email: there's a big probability that I do it (free time permitting).

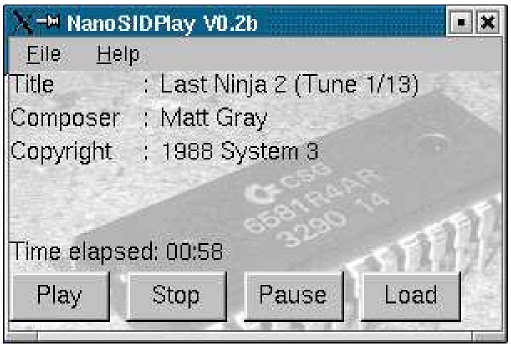
Bye

News

Some various news of players, libraries, programs, hardware and competitions:

- NanoSIDPlay v0.2b
- HVSC 5.2
- CGSC 1.10
- Libsidplay2 2.1.0
- Goattrk V1.4b Final
- ReSid 0.14
- BOV 2002
- Catweasel MK3 PCI/Flipper
- XSidplay 1.6.5.1a
- TSID 0.8
- Sid-Wine 2 compo
- Steppe and Yodelking' s Rip Compo

NanoSIDplay v0.2b



Title : Last Ninja 2 (Tune 1/13)
Composer : Matt Gray
Copyright : 1988 System 3
Time elapsed: 00:58

This windows player developed by Tom Roger Skauen is the first that use the NanoSID library by Laurent Ovaert.

This library use a custom sid file (ZSID) that contains SID data and information about the sound generation (like ADSR bug). So a very old computer can perform sid music with a quality near to RESID emulation.

The custom zsid files, the player and the library are available at: http://www.c64.no/nanosidplay and http://www.sid6581.org/NanoSID/

HVSC 5.2

Released on 21/12/2002 the upgrade 34 of HVSC is the most big update ever!

After this update, the collection should contain 20,325 SID files!

This update contains:

```
1009 new SIDs  (yes, over 1,000 SIDs!)
  64 fixed/better rips
 100 fixes of PlaySID/Sidplay1 specific SIDs (ie: RSID files)
   2 repeats/bad rips eliminated
 275 SID credit fixes
```

Download the collection/upgrade from: http://www.hvsc.c64.org but be sure to have the update tool at version 2.8.2 before apply the upgrade.

CGSC 1.10

Released on 01/01/2003 The new 1.10 version of The Compute' sGazette Sid Collection (http://www.c64music.co.uk/) with a little update file.

In the collection there are so:

5799 MUS, 1235 STR, 1527 WDS (including 73 with Extended Words)

Libsidplay2 2.1.0

Released on 27/12/2002 by Simon White the libsidplay2 version 2.1.0 and sidplay2 2.0.8.

Now all the stuff for compiling libsidplay2 is in one package and this made more simple to get the library compiled in Unix system.

Download all the staff from http://sidplay2.sourceforge.net

Goattrk V1.4b Final

The new final version of Cadaver' sPC Goattrk player and a stereo version are available at http://covertbitops.c64.org/tools/goattrk.zip and http://covertbitops.c64.org/tools/gstereo.zip and was release on 9 January 2003.

In this new version there are the implementation of a programmable (1 or 2 frame) hard restart and better steps programmable filter. Also available the tweak utility, and Goattrk songs to Ninjatrk format converter.

ReSid 0.14

The 0.14 version of Dag Lem' sReSid library was released on 14 January 2003 and is available here: ftp://ftp.funet.fi/pub/cbm/crossplatform/emulators/resid/resid-0.14.tar.gz

changes in it:

- The SID external audio input is now emulated: now we can have 8580 sample listen as with a external resistance.
- The filter settings are now updated immediately when the chip model is changed.
- Two bugs removed and improvement in exponential counter emulation.

BOV 2002

The Best Of Various for 2002 is being determine by the Sid fans (nomination ends 8 Feb.)
Check the state at The Shark site: http://www.dhp.com/~shark/BOV/

The Best Of Various is an attempt to determine the best tunes that are in the VARIOUS directory of the HVSC by an user voting system.

Catweasel MK3 PCI/Flipper

This new PCI based board have some nice features in it.

For a Sid fans, the support of the SID chip is the best of it: allowed 6581 or 8580; the board can programmable switch to PAL and NTSC frequency.

Check it at: http://www.jschoenfeld.de

XSidplay 1.6.5.1a

XSidplay continues to improve his integration with KDE3, QT3 and the new libsidplay2 with this release.

Download the Unix Sid player from:
http://www.geocities.com/SiliconValley/Lakes/5147/sidplay

TSID 0.8

Just some schedule minor updates are present in this version of the library that store your listening statistic about tunes in HVSC. I very hope that one day TSID2 will be as stable as TSID is being showing now.
Download from http://tsid.sourceforge.net

Sid-Wine 2 compo

A new Sid-Wine 2 C64 Online Music Competition will start from 21 February to 21 April 2003. In order to organize the event, a little survey is available: everyone can choose how the compo will be by filling this file: http://digilander.iol.it/ice00/tsid/sidwine2/survey.txt

Check the state of the event at: http://digilander.iol.it/ice00/tsid/sidwine2

Steppe and Yodelking's Rip Compo

A new compo is running until the 31 March 2003. It is for helping HVSC to grown by performing some ripping task:

- Rips some Sid Hunt request
- Makes a RSID rip version
- Fixes bugged rips
- Found image of file to rip

See the complete compo at: http://www.demodungeon.com/ripcompo/

Stephan Schmid Interview!

by Stefano Tognon

In this issue you can read this interview with Stephan that had occurs with some emails in Febrary on this year. Hopefully you know Stephan because he is now a HVSC crew member, or for his great work onto demos and music. But I, instead, know his fantastic “Tune 2” music that I consider one of the best Last Ninja 2 inspired composition!!

Hello, Stephan,
Before starting why not you say something about your real life: where do you live, what do you do...

I'm 28 years old and live in Germany in a rather small city called Erlangen.
Presently I'm still studying geology, but I hope to get my studies finished by June this year.
I'm the webmaster of the Demodungeon (www.demodungeon.com), a site meant as a tribute to the C64 demoscene. Since I had the idea to present all the SID tunes that played in the demos along with the screenshots I started ripping the ones that I couldn't find in HVSC. My ongoing contributions to HVSC finally ended up with the HVSC crew inviting me to join in October 2002.

You were active in C64 music production in 1991, then nothing else. Could you say something about this short period of activity?

Yep, my productivity period on the C64 was remarkably short, to be precise, it lasted only for about 3 weeks.
I got the Soundmonitor by Chris Hülsbeck that was back then published in the 64'er magazine as a type-in listing. I learned to use it really quickly, in fact I made my first tune (Erste Schritte) on the day I finished typing it in. I've been fascinated by those awesome SID tunes in the crack intros ever since, probably even more fascinated than by the game tunes. I tried to reproduce the sounds on the Soundmonitor, which was simply impossible. :-(So I quickly quit playing around with it, disappointed by the limited capabilities of the editor... I decided to look around for some better editors, but never really managed to get back to the keyboard - until this week when I finally started hammering it again, this time with the JCH editor. It's not quite easy to use, but I'm sure you'll hear more SID music from me soon. :-)

Now you are an HVSC crew member, so you probably can answer a common question: how much work this task require? (All the sid fans attends always a new HVSC update, but we probably don't know all the work behind this!).

This is really hard to answer, mostly because I don't want to admit to myself how much time I really waste (sorry: spend) with this project. ;-)
Most of the time I'd say I spend about one to two hours per day on HVSC stuff. This involves checking a few demos or disks for unripped tunes, ripping them when necessary, contacting composers for information and verification, writing mails to the quite busy HVSC mailing lists, maintaining the Ripped Tunes list and the RSID to do list. But on the other hand I don't really want to see it as "work", it's my favourite hobby and I really enjoy working with this awesome bunch of guys in the HVSC crew. The time put into HVSC usually increases one or two weeks before the update when it comes to thoroughly checking the upcoming beta-version of the update.

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

I admit that I use the emulator most of the time for productivity stuff, like ripping tunes and everything coding-related that is involved with it. And why not? Vice's emulation quality is slowly nearing perfection and doing that stuff in a multitasking environment along with other tools that you keep open in other windows and that you need in addition really makes it a pleasure to work with. Then theres the backside of the medal: I never use the emu for playing games (which I rarely do anyway) or to watch some demos alone or with friends. Emulators just can't capture the original feeling of a real C64. Same goes for composing, you just need a real SID chip, the background noise and the humming of the 1084. I have my C64 here on my desktop right beside my PC keyboard and a 1541-II on my PC case that is only used for transferring disks. That way I can quickly download and transfer everything over to the real machine if I want to.

6581 vs 8580 chip: any (musical) preference?

I like both chips, each one has its advantages.
Samples and filters on the 6581, the combined waveforms and the generally clearer sound on the 8580. To fit both demands I always have a breadbox and a C64-II handy to quickly change between them when I know that a demo or game really relies on a certain chip to make it sound as it originally was intended. One of my future projects (or let's call it a dream) is a C64 with two different SIDs that I can switch on demand, if possible even while the C64 is running. That would be really nice for demos that use tunes from different composers. For example the recently released Insomnia by 64ever has tunes from Jeff and Vip. Jeff's tunes heavily use the 6581 filters and really sound much better on the old SID, whereas Vip's tunes should really be listened to on a 8580. If anyone out there has ever built something like that and want to share some experience with me, please contact me at steppe@demodungeon.com.

What is the worst sid that you compose and the better one?

Haha, actually I think that all my own SIDs aren't too good. I'd give a big thumbs down to Erste Schritte, it's really nothing special, both musically and technically. I like my "Tune 2" with the unintentionally crept in Last Ninja II cover or the soft arpeggios in "Nicht Schlecht". Here I also made my first attempt on doing a pseudo 4 channel sound, by interweaving drums and bassline.

Who are your best sid authors?

That question is really hard to answer as I have a rather broad musical taste. From the classical game-composers I prefer Jeroen Tel, Galway, Follin, Matt Gray and Martin Walker. Ok, it would be unfair to leave Hubbard unmentioned, but I spend far less time listening to him than on other composers. Actually I like the "new" sound of the scene musicians far better anyway, so I spend surely 90% of my time listening through the /VARIOUS directory. To name a few outstanding ones: Mitch & Dane, Shogoon, Daf, Jeff, Vip, DJB, GRG, Goto80, Trident, Wacek and Wizard. I could go on with this list forever, but those are the ones I like most for one or the other reason.

What are the best sids ever in your opinion?

Aargh, that's again one of those questions! ;-) I have about 600 SIDs in my alltime favourite playlist and my preference really changes on a daily basis. But let me try to name a few outstanding ones from /VARIOUS, as I'm tired of repeating the same Hubbard/Galway/Follin playlists that everyone already knows anyway:

/VARIOUS/A-F/Bakewell_Dwayne/Old_Friends_PSID.sid
# if you manage to get past the rather long intro you'll get rewarded with a fantastic lead-melody! And the samples fit in really well, too.

/VARIOUS/A-F/Behdad_Arman/Sweet_Fantasies.sid
# from one of my favourite demos, Arcanum by Xenon.

/VARIOUS/A-F/Blues_Muz/Gallefoss_Glenn/Electronic_Transfer.sid
# Technically maybe one of the best SIDs ever done. I could name several others from his dir, be sure to check out the Soiled Legacy tunes at least!

/VARIOUS/G-L/Goto80/Datagoaskit_2002.sid
# So weird and yet so mature! I catch myself listening to that one for hours in the background. 8580!!!

/VARIOUS/G-L/Goto80/Slobband.sid
# For its minimalism and a perfect choice of instruments.

/VARIOUS/M-R/Mitch_and_Dane/Gloria.sid
# Still one of my alltime favourites by Mitch & Dane. Cheerful, makes me feel good.

/VARIOUS/S-Z/Shapie/Im_Back.sid
# SID popmusic at its best!

/VARIOUS/S-Z/Shogoon/Altered_States_Tune_1.sid
# I pulled my hairs out over which tune to pick from groovemaster Shogoon! He's musically lightyears ahead of the crowd, a truly outstanding musician which I admire both for his perfect sounds and compositions.

/VARIOUS/G-L/Jeff/Crest_Main_A.sid
# Just had to mention one of his numerous hits. The middle part of this tune starting at around 2:00 is mindblasting!

/VARIOUS/S-Z/Sonic/Man_With_No_Name_culture.sid
# Still blows me away, even after having heard it for so many times.

Finally, many thanks for the time you give for this interview, and now you can say any things you want that the people will read from you!

It was a pleasure answering those questions for your mag!
To all the active C64 people out there I want to say: Keep the 8 bit torch up, you did a fantastic job in the past and it amazes me that there's still some activity going on in the scene in the year 2003. Finally I want to draw your attention to the fact that Peter Sandén and me are doing a SID rip-compo at the moment. It's meant to fill out the last empty spots in HVSC with missing SID tunes that have been requested by many SID fans all over the world. Some of them are really hard to rip, but some are quite easy too. So if you have some ripping knowledge and want to help make HVSC a better and more complete collection, come to http://www.demodungeon.com/ripcompo/. There's even some nice prizes to win like original games, old C64 books, Instant Remedy's C64 remix CD and more.

Webography:

Demodungeon:
www.demodungeon.com

Music Engine pattern searching (part 1)

by Stefano Tognon <ice00@libero.it>

I don't know if you are asking like me with witch music player a given Sid tune is made by. It is just an information not needed by listing to the tune, but I think that it is more interesting to know with witch routine the sound was made.

Here I want to illustrate how simple can be determine the Music Engine used by one tune if we have a pattern of it and we use a language that allow pattern searching (like Perl)

Pattern searching

The concept is simple: scan a file to found if it contains some bytes in sequence that we know that only a file made by a particular music engine can have. If true than we can say that this Sid tune is made using that music engine.

Here the difficult in this is only one: found a representative pattern for a given music engine. I think that there are two kind of approach in this:

1. Found a pattern having a kind of source code of the player
2. Found a pattern not having a source code of the player

The first case is very simple: if the author of the player had released the source code, or there are some reverse engineering work about the player, we can found a pattern by looking for characteristic code used by the player.

For examples each music player can have their special instruments, patterns, notes representation, so looking for code that manage them are a good point for finding the player.
Else, looking for how some music effects like portamento, vibrato, arpeggio are made can be valid too.

Considering for example, the Martin Galway's engine source code that I will present in a next issue: you will see that the pattern can have vales greater than C0h, because those values are instructions for the music flow. So, this simple pattern found the Martin Galway engine (it is in Perl syntax, so see the last paragraph for further information):

```
# Martin Galway engine
# search for instruction in the read byte
'\xA0\x00'.             # LDY  #$00
'\xB1.'.                # LDA  ($xx),Y
'\xC9\xC0'.             # CMP  #$C0
'\x90',                 # BCC  $xxxx
```

So with 6 fixed bytes and a jolly byte in the fourth position, we can found the engine. As 7 bytes are very little, maybe some other music engines completely different can have the same code part with other meanings: if we will found that this is the case, the pattern should be changed.

Another example: look for the Driller source code presented in the last issue. For finding a Matt Gray engine, we can look for the code that check for portamento flag in the pattern of note. This is a good point as the code is very characteristic:

```
# Matt Gray engine
# search for the code that read the next pattern value and set the
# portamento flag
```

```
     '\xFE..'.          # INC  $xxxx,X
     '\xD0.'.           # BNE  $xxxx
     '\xC9\xFB'.        # CMP  #$FB
     '\x90\x20'.        # BCC  $xxxx
     '\xC9\xFB'.        # CMP  #$FB
     '\xD0\x18'.        # BNE  $xxxx
     '\xA9\x01'.        # LDA  #$01
     '\x9D..'.          # STA  $xxxx,X
     '\xC8'.            # INY
     '\xFE..'.          # INC  $xxxx,X
     '\xB1',            # LDA  ($xx),Y
```

As the cases that we have a source code of the player are very little, maybe we must use some disassembly files as shown in the next paragraphs.

Found the pattern

If you did not have the source code, then use a disassembly and obtains a spartan one. Remember to be sure to disassembly a tune you know that are made by that music player because the author say that or you have generate a test tune from the player editor just for this.

With this code you may choose a portion of it as a pattern and test if the choice were good. However I think that a better one is to choose a part that are not used by the other versions of the same music player. Here I show how we can found JCH Newplayer version 1, 2 and 3.

Below there are (part of) the disassembled code of 4 JCH tunes: Hoppin, Brown Ice, Cavern, 2Cvee.

All the code were generated by SIDedit disassembler and then copied into a spreadsheet (you may use and editor that allow a vertical selection for coping only the assembler instructions). Now simple insert some cells spacing in a manner to make the code to be the same in the same rows. As we are comparing engines to progressive versions, this is not so difficult to achieve, as the new versions probably have little more features before the previous (however this is not always true)

Hoppin V1	Brown Ice V1	Cavern V2	2Cvee V3
ASL	ASL	ASL	ASL
STA $166B,X	STA $16AA,X	STA $1727,X	STA $178D,X
INC $FB	INC $FB	INC $FB	INC $FB
BNE 10BB	BNE 10CC	BNE 10C8	BNE 112B
INC $FC	INC $FC	INC $FC	INC $FC
JMP $10BB	JMP $10CC	JMP $10C8	JMP $112B
CMP #$C0	CMP #$C0	CMP #$C0	CMP #$C0
BNE 1125	BNE 1140	BNE 113C	BNE 119F
PLA	PLA	PLA	PLA
ASL	ASL	ASL	ASL
ASL	ASL	ASL	ASL
ASL	ASL	ASL	ASL
ASL	**ASL**	**ASL**	**ASL**
	CLC	**CLC**	**CLC**
	ADC #$0B	**ADC #$0B**	**ADC #$0B**
	TAY	**TAY**	**TAY**
	LDA $175A,Y	**LDA $17E9,Y**	**LDA $1884,Y**

```
                EOR $1682,X     EOR $16FF,X     EOR $1765,X
                STA $175A,Y     STA $17E9,Y     STA $1884,Y
STA $165F,X
JMP $110D       JMP $111E       JMP $111A       JMP $117D
PLA             PLA             PLA             PLA
CMP #$FE        CMP #$FE        CMP #$FE        CMP #$FE
BNE 113C        BNE 1157        BNE 1153        BNE 11B6
DEC $1689,X     DEC $16C8,X     DEC $1745,X     DEC $17AB,X
BMI 110D        BMI 111E        BMI 111A        BMI 117D
LDA $1649,X     LDA $1679,X     LDA $16F6,X     LDA $175C,X
STA $FB         STA $FB         STA $FB         STA $FB
LDA $164C,X     LDA $167C,X     LDA $16F9,X     LDA $175F,X
STA $FC         STA $FC         STA $FC         STA $FC
JMP $10BB       JMP $10CC       JMP $10C8       JMP $112B
                                                CMP #$FD
                                                BNE 11C5
                                                LDA #$00
                                                STA $100C
                                                STA $D412
                                                JMP $1645
LDA $163D,X     LDA $166D,X     LDA $16EA,X     LDA $1019,X
STA $FB         STA $FB         STA $FB         STA $FB
LDA $1640,X     LDA $1670,X     LDA $16ED,X     LDA $101C,X
         ...             ...             ...             ...
LDY $1677,X     LDY $16B6,X     LDY $1733,X     LDY $1799,X
LDA $16E9,Y     LDA $175E,Y     LDA $17ED,Y     LDA $1888,Y
PHA             PHA             PHA             PHA
AND #$0F        AND #$0F        AND #$0F        AND #$0F
STA $1722,X     STA $16F2,X     STA $176F,X     STA $17D5,X
PLA             PLA             PLA             PLA
AND #$F0        AND #$F0        AND #$F0        AND #$F0
LSR             LSR             LSR             LSR
LSR             LSR             LSR
STA $1702,X     STA $16E9,X     STA $1766,X     STA $17CC,X
STA $102C,X     STA $16E3,X     STA $1760,X     STA $17C6,X
LDA #$00        LDA #$00        LDA #$00        LDA #$00
STA $16EF,X     STA $16EF,X     STA $176C,X     STA $17D2,X
SEC             SEC             SEC             SEC
SBC $102C,X     SBC $16E3,X     SBC $1760,X     SBC $17C6,X
STA $16F2,X     STA $16E6,X     STA $1763,X     STA $17C9,X
LDA $16EA,Y     LDA $175F,Y     LDA $17EE,Y     LDA $1889,Y
STA $1712,X     STA $16EC,X     STA $1769,X     STA $17CF,X
                                CPX #$00        CPX #$00
                                BNE 134F        BNE 13C1
LDA $16EB,Y     LDA $1760,Y     LDA $17F8,Y     LDA $1893,Y
PHA             PHA             PHA             PHA
AND #$F0        AND #$F0
```

```
STA $16AA,X           STA $16FB,X
PLA                   PLA
AND #$0F              AND #$0F               AND #$0F              AND #$0F
                                             BEQ 1349              BEQ 13BB
                                             ASL                   ASL
                                                                   ASL
                                             STA $1772             STA $17D8
                                             STA $1773             STA $17D9
                                             PLA                   PLA
                                             AND #$F0              AND #$F0
                                             CLC                   CLC
                                             ADC $1777             ADC $1009
                                             STA $D418             STA $D418
                                             LDA $17F7,Y           LDA $1892,Y
                                             STA $1775             STA $17DB
                                             LDA $17F6,Y           LDA $1891,Y
                                             PHA                   PHA
                                             AND #$F0              AND #$F0
                                             CLC                   CLC
                                             ADC #$01              ADC #$01
                                             STA $D417             STA $D417
                                             PLA                   PLA
                                             AND #$01              AND #$01
                                             STA $1774             STA $17DA
                                             LDA $17F5,Y           LDA $1890,Y
                                             STA $1776             STA $17DC
                                             STA $D416             STA $D416
                                             JMP $134F             JMP $13C1
                                             PLA                   PLA
                                             LDA #$00              LDA #$00
                                             STA $D417             STA $D417
                                             LDA $17EF,Y           LDA $188A,Y
                                             PHA                   PHA
                                             AND #$F0              AND #$F0
                                             STA $177E,X           STA $17E3,X
                                             PLA                   PLA
                                             AND #$0F              AND #$0F
STA $16A7,X           STA $16F8,X            STA $177B,X           STA $17E0,X
LDA $16EC,Y           LDA $1761,Y            LDA $17F0,Y           LDA $188B,Y
STA $16A4,X           STA $16F5,X            STA $1778,X           STA $17DD,X
LDA $16ED,Y           LDA $1762,Y            LDA $17F1,Y           LDA $188C,Y
PHA                   PHA                    PHA                   PHA
AND #$F0              AND #$F0               AND #$F0              AND #$F0
CMP #$10              CMP #$10               CMP #$10              CMP #$10
BNE 133E              BNE 1365               BNE 13A8              BNE 141A
PLA                   PLA                    PLA                   PLA
AND #$0F              AND #$0F               AND #$0F              AND #$0F
```

At this point we can see that the first two songs are made by two version 1 that are little different: in particular the second seems to be an improvement of the fist (look at the bold code to see the difference). For finding a pattern for the version 1 of the player, we must found a code that are common in the two first tunes, but that are not present in the other two. The solution is the blue code, while the red shows a pattern only use by version 2, and green by version 3 of the player.

What the selected code in the pattern means for the player we are not able to say: this require an accurate reverse engineering work, but we only need a pattern that find that player.
Now convert the pattern to the appropriate programming language (see next paragraph) and test the pattern to the tunes you know to be made by this engine. All the tunes may be founded, otherwise you have found a minor variant that need further investigation.

Write the pattern

Now we have found 3 peaces of code that represent the JCH NewPlayer version 1, 2 and 3. Enter again in SIDedit ant pick up the complete code (e.g byte codes + assembler instructions) that was found as a pattern; here the code for version 1:

```
;12BC    BC 77 16    LDY $1677,X
;12BF    B9 E9 16    LDA $16E9,Y
;12C2    48          PHA
;12C3    29 0F       AND #$0F
;12C5    9D 22 17    STA $1722,X
;12C8    68          PLA
;12C9    29 F0       AND #$F0
;12CB    4A          LSR
;12CC    4A          LSR
;12CD    9D 02 17    STA $1702,X
;12D0    9D 2C 10    STA $102C,X
;12D3    A9 00       LDA #$00
;12D5    9D EF 16    STA $16EF,X
;12D8    38          SEC
;12D9    FD 2C 10    SBC $102C,X
;12DC    9D F2 16    STA $16F2,X
;12DF    B9 EA 16    LDA $16EA,Y
;12E2    9D 12 17    STA $1712,X
;12E5    B9 EB 16    LDA $16EB,Y
;12E8    48          PHA
;12E9    29 F0       AND #$F0
;12EB    9D AA 16    STA $16AA,X
;12EE    68          PLA
;12EF    29 0F       AND #$0F
;12F1    9D A7 16    STA $16A7,X
;12F4    B9 EC 16    LDA $16EC,Y
;12F7    9D A4 16    STA $16A4,X
;12FA    B9 ED 16    LDA $16ED,Y
;12FD    48          PHA
;12FE    29 F0       AND #$F0
;1300    C9 10       CMP #$10
;1302    D0 3A       BNE 133E
```

Now the first thing to know if that absolute memory location are always relative to the point where the player was linking in memory, so they are not to be found. So, for example, of the first 6 bytes, we may have: BD ?? ?? B9 ?? ?? where the ?? means that this byte can have any value. The only case where absolute memory location are to be left are for those that are pointing to I/O space, like D4xx (the sid chip).

The same thing are to apply to zero-page locations, because even if it is most probably that the zero pages did not changes, maybe some player let the user to choose the zero-page locations that are to be used, so it is to prefer to let them unknown.

Even if relative location (like branch jumping) could not change, I prefer to not insert that branch value if it is outside the peace of code we are using as pattern.

At this point is simple: convert the remaining pattern values to the language you want to use: for example here the Perl code for the above pattern (a point **.** in a Perl string has the same meaning of ??, and \x is a prefix for a hexadecimal number) :

```
# JCH NewPlayer v1
# search for code common to all the v1 revisions. but not
# equal to >v1 version
'\xBC..'.             # LDY $xxxx,X
'\xB9..'.             # LDA $xxxx,Y
'\x48'.               # PHA
'\x29\x0F'.           # AND #$0F
'\x9D..'.             # STA $xxxx,X
'\x68'.               # PLA
'\x29\xF0'.           # AND #$F0
'\x4A'.               # LSR
'\x4A'.               # LSR
'\x9D..'.             # STA $xxxx,X
'\x9D..'.             # STA $xxxx,X
'\xA9\x00'.           # LDA #$00
'\x9D..'.             # STA $xxxx,X
'\x38'.               # SEC
'\xFD..'.             # SBC $xxxx,X
'\x9D..'.             # STA $xxxx,X
'\xB9..'.             # LDA $xxxx,Y
'\x9D..'.             # STA $xxxx,X
'\xB9..'.             # LDA $xxxx,Y
'\x48'.               # PHA
'\x29\xF0'.           # AND #$F0
'\x9D..'.             # STA $xxxx,X
'\x68'.               # PLA
'\x29\x0F'.           # AND #$0F
'\x9D..'.             # STA $xxxx,X
'\xB9..'.             # LDA $xxxx,Y
'\x9D..'.             # STA $xxxx,X
'\xB9..'.             # LDA $xxxx,Y
'\x48'.               # PHA
'\x29\xF0'.           # AND #$F0
'\xC9\x10'.           # CMP #$10
'\xD0'                # BNE xxxx
```

Now that we have the string pattern, for looking if a tune is of that engine, in Perl we need only one instruction:

```
if ($data =~ /$Search/s) {
  print “Found!!!\n”;
}
```

where $data will contain all the body of the tune to scan and $Search contains the string pattern to search for.

Conclusion

Can we do a better task?

Yes, in the next issue we will see the Lada ' RayLostak database of engines (try a look here: http://www.unreal64.net), that contains over 450 engines in it.

We will see how to create and entry and how people could collaborate to make the database grown: the main goal is to will be able to find the engines of more that 90% of HVSC tunes!

Ripping BASIC based tune

by Stefano Tognon <ice00@libero.it>

Although you know that using the BASIC language you cannot achieve good music result, there are many old games that where use the BASIC even for generating the music. As an old game you play is probably a good memory for you, maybe you would want to listen to that tune in a sid player. So the main point is: how we can rip an old BASIC game?

Here I illustrate the steps I used for ripping the Testcard demo presented by Andrew Colin in Introduction to Basic part 1, and The Arrival of The Queen of Sheba music program showed in Introduction to Basic part 2.

Note that the BASIC programs showed here are Copyrighted by Andrew Colin and Commodore: I have reproduced the code of Testcard as it is not showed in the books, while I' ve omitted the one for Sheba as it is present in the book.

The Arrival of the Queen of Sheba

This tune was presented in the Appendix A of the second part of the course: it purpose was to show what the machine code can do with all the 3 Sid voices. This let our work be easier: in fact the BASIC program is only used for setting up all the stuff needed by the custom machine code player to play the music.

Look at the code: the machine code routine is loaded into memory with the BASIC code from row 20 to row 280; then voice parameters used by the player are copied to the right place in memory and a table of frequency are generated too (rows 300-430). At this point Sid values are set for each voice (500 to 565) and finally each pattern of data for each voice are decoded and putted into memory. You know from the description in the book that for each voice there are the note value and his duration, coded in a particular manner. During this decoding process, the machine code routine is called so music can start while the BASIC program is still working.

All we need for making the rip in now simple: start the program with an emulator and in the monitor make a disassembly of the player code and look as it works. Then pick up all the memory blocks wrote by the BASIC program and store in a file. Now open an editor, insert the disassembled code of the player plus the saved data. Add the needed initialization part that the BASIC does for the SID chip in the initialization part of the PSID file, then setting the right IRQ timing as done by the player.

Now you can so see all the source of the rip:

```
;=========================================
; The arrival of the queen of Sheba
; by G. F. Handel
; Arranged by Andew Colin (1982/83)
;
; The originale song use a Basic/machine
; code for generating music
; This is a machine code port
; The code can generate a PRG or SID file.
;=========================================

.ifdef sid
 .byte "PSID"
 .word $0200          ; version 2
 .word $7C00          ; data offset
 .word $0000          ; load address in cbm format
 .byte >initMusic
 .byte <initMusic
 .byte >playMusic
 .byte <playMusic
 .word $0100         ; 1 song
 .word $0100         ; default song
 .word $FFFF
 .word $FFFF
 .byte "Arrival of the queen of Sheba",0,0,0
 .byte "Andrew Colin",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
 .byte "1982-83 Commodore Inc",0,0,0,0,0,0,0,0,0,0,0
 .word $0000
 .word $0000
 .word $0000

.endif

.org $0801

 ;=================================
 ; PRG start address
 ;=================================
 .byte $01,$08

 ;=================================
 ; BASIC startup program
 ;=================================
 .byte $0b,$08,$e8,$03,$9e,"2061",0,0,0

.org 2061
     jsr initMusic

     sei
     lda #<interr
     sta $0314
     lda #>interr
     sta $0315

     cli

aaa:
 jmp aaa

interr =*
     ;lda  $01
     ;and  #$fe
     ;sta  $01
     lda  contr+0
     bne  playMusic
kernal:
     ;lda  $01
     ;ora  #$01
     ;sta  $01
     jmp  $ea31

playMusic:
     ldx #$00
     lda instr+0,x                ; Voice 1: Control registers
     beq nend
     inc instr+5,x
     bne nend
     inc instr+6,x
     bne nend
     lda instr+1,x                ; Voice 1: notes address lo
     sta $fe
     lda instr+2,x                ; Voice 1: notes address hi
     sta $ff
     ldy #$00
     lda ($fe),y
     asl
     tay
     lda contr+2,y
     sta $d400,x                  ; Voice 1: Frequency control (lo byte)
     lda contr+3,y
     sta $d401,x                  ; Voice 1: Frequency control (hi byte)
     lda instr+3,x                ; Voice 1: notes duration address lo
     sta $fe
     lda instr+4,x                ; Voice 1: notes duration address hi
     sta $ff
     ldy #$00
     lda ($fe),y
     bne skip
     sta instr+0,x                ; Voice 1: Control registers
     jmp nend
skip:
     tay
loop:
     lda instr+5,x
     sec
     sbc contr+1
     sta instr+5,x
     lda instr+6,x
     sbc #$00
     sta instr+6,x
     dey
     bne loop
     inc instr+1,x                ; Voice 1: notes address lo
     bne skip1
     inc instr+2,x                ; Voice 1: notes address hi
skip1:
     inc instr+3,x                ; Voice 1: notes duration address lo
     bne skip2
     inc instr+4,x                ; Voice 1: notes duration address hi
skip2:
     lda #$00
     sta $d404,x                  ; Voice 1: Control registers
     lda instr+0,x                ; Voice 1: Control registers
     sta $d404,x                  ; Voice 1: Control registers
nend:
     txa
     clc
     adc #$07
     tax
     cmp #$15
     bne skip3
.ifdef sid
     rts
.endif
     jmp kernal
skip3:
     jmp playMusic+2
```

Here the init part as done by the BASIC

```
initMusic =*
                                  ; initialize control
registers
     lda #$00
     ;sta contr+0
     sta contr+2
     sta contr+3
     lda #$8
     sta contr+1
                                  ; erase all sid registers
     ldx  #24
     lda  #0
loop1:
     sta  $D400,x
     dex
     bpl  loop1
                                  ; initialze sid registers
     lda  #25
     sta  $D405                   ; Generator 1: Attack/Decay (voice 1)
     sta  $D40C                   ; Generator 1: Attack/Decay (voice 2)
     sta  $D413                   ; Generator 1: Attack/Decay (voice 2)
     lda  #79
     sta  $D418                   ; Select volume and filter mode
     lda  #200
     sta  $D402                   ; Voice 1: Wave form pulsation amplitude (lo byte)
     lda  #60
     sta  $D409                   ; Voice 2: Wave form pulsation amplitude (lo byte)
     sta  $D410                   ; Voice 3: Wave form pulsation amplitude (lo byte)
     lda  #50
     sta  $D416
     lda  #0
     sta  $D417

     rts

contr =*                          ; control register
  .byte 1;0
  .byte 8
  .byte 0
  .byte 0

freq =*
  .byte 76, 4
  .byte 141, 4
  .byte 210, 4
  .byte 28, 5
  .byte 105, 5
  .byte 188, 5
  .byte 19, 6
  .byte 112, 6
  .byte 210, 6
  .byte 57, 7
  .byte 167, 7
  .byte 28, 8
  .byte 152, 8
  .byte 26, 9
  .byte 165, 9
  .byte 56, 10
  .byte 211, 10
  .byte 120, 11
  .byte 39, 12
```

This is the frequency table as the BASIC had generated

```
  .byte 224, 12
  .byte 164, 13
  .byte 115, 14
  .byte 79, 15
  .byte 57, 16
  .byte 48, 17
  .byte 53, 18
  .byte 74, 19
  .byte 112, 20
  .byte 167, 21
  .byte 241, 22
  .byte 78, 24
  .byte 192, 25
  .byte 72, 27
  .byte 231, 28
  .byte 159, 30
  .byte 114, 32
  .byte 96, 34
  .byte 107, 36
  .byte 149, 38
  .byte 225, 40
  .byte 79, 43
  .byte 226, 45
  .byte 157, 48
  .byte 129, 51
  .byte 145, 54
  .byte 207, 57
  .byte 63, 61
  .byte 228, 64
  .byte 192, 68
  .byte 214, 72
  .byte 43, 77
  .byte 194, 81
  .byte 158, 86
  .byte 197, 91
  .byte 58, 97
  .byte 2, 103
  .byte 34, 109
  .byte 159, 115
  .byte 127, 122
  .byte 200, 129
  .byte 128, 137
  .byte 173, 145
  .byte 86, 154
  ;.byte 132, 163
  ;.byte 61, 173
  ;.byte 138, 183
  ;.byte 116, 194
  ;.byte 4, 206

instr=*
  .byte 65                        ; Voice 1: Control registers
  .byte <note1                    ; Voice 1: notes address lo
  .byte >note1                    ; Voice 1: notes address hi
  .byte <len1                     ; Voice 1: notes duration address lo
  .byte >len1                     ; Voice 1: notes duration address hi
  .byte 255
  .byte 255

  .byte 17                        ; Voice 2: Control registers
  .byte <note2                    ; Voice 2: notes address lo
  .byte >note2                    ; Voice 2: notes address hi
  .byte <len2                     ; Voice 2: notes duration address lo
  .byte >len2                     ; Voice 2: notes duration address hi
  .byte 255
  .byte 255

  .byte 17                        ; Voice 3: Control registers
  .byte <note3                    ; Voice 3: notes address lo
  .byte >note3                    ; Voice 3: notes address hi
  .byte <len3                     ; Voice 3: notes duration address lo
  .byte >len3                     ; Voice 3: notes duration address hi
  .byte 255
  .byte 255

note1 =*                          ; note value for voice 1
  .byte $2f, $2a, $27, $2a, $2f, $2a, $27, $2a
  .byte $2f, $2a, $27, $2a, $2f, $2a, $27, $2a
  .byte $25, $2f, $2e, $2c, $2a, $28, $27, $25
  .byte $27, $23, $25, $27, $28, $2a, $2c, $2e
  .byte $2f, $2a, $23, $2a, $2f, $2a, $27, $2a
  .byte $2f, $29, $25, $29, $2f, $2c, $25, $2c
  .byte $2e, $2c, $2a, $2c, $25, $29, $2a, $25
  .byte $27, $25, $27, $25, $27, $25, $22, $1e
  .byte $22, $25, $22, $1e, $22, $2a, $27, $23
  .byte $27, $2a, $27, $23, $27, $28, $25, $20
  .byte $25, $28, $25, $22, $25, $27, $1e, $20
  .byte $22, $23, $25, $27, $28, $2a, $27, $23
  .byte $27, $2a, $27, $23, $27, $2a, $25, $22
  .byte $25, $2a, $25, $22, $25, $27, $23, $20
```

```
.byte $23, $27, $23, $20, $23, $27, $22, $1e
.byte $22, $27, $22, $1e, $22, $2f, $2c, $28
.byte $2c, $2f, $2c, $28, $2c, $2f, $2a, $27
.byte $2a, $2f, $2a, $27, $2a, $28, $2a, $2c
.byte $2a, $28, $27, $25, $23, $22, $23, $25
.byte $23, $22, $20, $1e, $1c, $1b, $1e, $1c
.byte $20, $1e, $22, $20, $23, $22, $25, $23
.byte $27, $25, $28, $27, $2a, $28, $2a, $28
.byte $2a, $2c, $2a, $28, $27, $25, $27, $25
.byte $27, $28, $2a, $28, $2a, $2c, $2a, $2c
.byte $2a, $28, $27, $28, $27, $25, $27, $25
.byte $27, $28, $2a, $28, $2a, $2c, $2a, $2c
.byte $2a, $28, $27, $28, $27, $25, $27, $25
.byte $27, $28, $27, $28, $25, $27, $2a, $28
.byte $2c, $2a, $2e, $2c, $2f, $2e, $2c, $2a
.byte $2f, $28, $27, $25, $23, $1e, $22, $23
.byte $1e, $00, $23, $25, $27, $28, $27, $25
.byte $23, $2a, $28, $27, $28, $27, $28, $25
.byte $27, $28, $2a, $2c, $2a, $28, $27, $28
.byte $27, $28, $25, $27, $25, $23, $2f, $2a
.byte $27, $2a, $2f, $2a, $27, $2a, $2f, $2a
.byte $27, $2a, $2f, $2a, $27, $2a, $25, $2f
.byte $2e, $2c, $2a, $28, $27, $25, $27, $1e
.byte $17, $2a, $2c, $2a, $28, $27, $28, $2c
.byte $2c, $2a, $28, $27, $2a, $28, $27, $25
.byte $28, $27, $2a, $2a, $28, $27, $25, $28
.byte $27, $25, $23, $27, $25, $2a, $22, $23
.byte $25, $27, $25, $2a, $22, $23, $25, $27
.byte $25, $2a, $2a, $2a, $2a, $25, $22, $25
.byte $2a, $25, $22, $25, $2a, $25, $22, $25
.byte $2a, $25, $2a, $2e, $2c, $2a, $29, $27
.byte $25, $23, $22, $20, $22, $1e, $20, $22
.byte $23, $25, $27, $29, $2a, $2c, $2e, $2f
.byte $2e, $2c, $2a, $25, $23, $22, $23, $22
.byte $23, $20, $22, $23, $25, $27, $25, $29
.byte $2a, $2c, $2e, $2f, $2c, $2e, $2c, $2a
.byte $25, $22, $1e, $22, $25, $22, $1e, $22
.byte $25, $20, $1d, $20, $25, $20, $1d, $20
.byte $22, $1e, $1b, $1e, $22, $1e, $1b, $1e
.byte $22, $1d, $19, $1d, $22, $1d, $19, $1d
.byte $2a, $27, $23, $27, $2a, $27, $23, $27
.byte $2a, $25, $22, $25, $2a, $25, $22, $25
.byte $23, $25, $27, $25, $23, $22, $20, $1e
.byte $1d, $2c, $2e, $2c, $2c, $2a, $2f, $2e
.byte $2e, $2c, $2e, $2f, $2e, $2c, $2c, $2e
.byte $2c, $2a, $2c, $2e, $2c, $2a, $2a, $2c
.byte $2a, $29, $27, $29, $27, $25, $23, $25
.byte $23, $22, $20, $22, $20, $22, $23, $25
.byte $23, $25, $27, $25, $27, $25, $23, $22
.byte $23, $22, $20, $22, $20, $22, $23, $25
.byte $23, $25, $27, $25, $27, $25, $23, $22
.byte $23, $22, $20, $22, $20, $22, $23, $25
.byte $23, $25, $22, $2a, $20, $29, $2a, $25
.byte $1e, $27, $28, $2a, $2c, $2a, $28, $27
.byte $2a, $28, $27, $28, $27, $28, $25, $27
.byte $25, $23, $2f, $2a, $27, $2a, $2f, $2a
.byte $27, $2a, $2f, $2a, $27, $2a, $2f, $2a
.byte $27, $2a, $25, $2f, $2e, $2c, $2a, $28
.byte $27, $25, $27, $23, $25, $27, $23, $27
.byte $25, $25, $23, $28, $27, $27, $25, $27
.byte $28, $27, $25, $25, $27, $25, $23, $25
.byte $27, $25, $23, $23, $25, $23, $22, $23
.byte $25, $27, $28, $27, $25, $23, $2f, $2e
.byte $2c, $2e, $2c, $2e, $2b, $2c, $27, $23
.byte $27, $2c, $27, $23, $27, $2c, $27, $23
.byte $27, $2c, $27, $23, $27, $2c, $27, $2c
.byte $2f, $2e, $2c, $2b, $29, $27, $25, $23
.byte $22, $23, $20, $22, $23, $25, $27, $29
.byte $2b, $2c, $20, $28, $25, $27, $28, $2a
.byte $25, $27, $23, $25, $27, $28, $23, $25
.byte $22, $22, $23, $25, $27, $22, $23, $20
.byte $22, $23, $25, $22, $23, $25, $27, $23
.byte $25, $27, $28, $25, $24, $25, $2e, $2b
.byte $27, $2b, $2e, $2b, $27, $2b, $2c, $27
.byte $23, $27, $2c, $27, $23, $27, $2c, $2a
.byte $28, $27, $25, $23, $22, $20, $20, $2f
.byte $2c, $28, $2c, $2f, $2c, $28, $2c, $2f
.byte $2a, $27, $2a, $2f, $2a, $27, $2a, $2c
.byte $28, $25, $28, $2c, $28, $25, $28, $2c
.byte $27, $23, $27, $2c, $27, $23, $27, $28
.byte $25, $21, $25, $28, $25, $21, $25, $28
.byte $23, $20, $23, $28, $23, $20, $23, $2d
.byte $2c, $2a, $28, $27, $25, $23, $21, $20
.byte $1c, $20, $23, $20, $21, $23, $25, $23
.byte $21, $20, $2c, $2a, $28, $2a, $28, $2a
.byte $27, $28, $2f, $2d, $2c, $2d, $2c, $2d
.byte $2a, $2c, $2a, $28, $23, $20, $1c, $20
.byte $23, $20, $1c, $20, $25, $20, $1d, $20
.byte $25, $20, $1d, $20, $25, $22, $1e, $22
.byte $25, $22, $1e, $22, $27, $22, $1f, $22
```

```
  .byte $27, $22, $1f, $22, $27, $23, $20, $23
  .byte $27, $23, $20, $23, $28, $23, $20, $23
  .byte $28, $23, $20, $23, $28, $25, $21, $25
  .byte $28, $25, $21, $25, $2a, $25, $22, $25
  .byte $2a, $25, $22, $25, $27, $28, $2a, $2c
  .byte $2a, $28, $27, $2a, $28, $27, $28, $27
  .byte $28, $25, $27, $28, $27, $25, $27, $25
  .byte $23, $28, $27, $25, $27, $28, $2a, $2c
  .byte $2a, $2a, $22, $23, $25, $27, $28, $2a
  .byte $23, $22, $23, $25, $27, $28, $2a, $2c
  .byte $25, $24, $25, $27, $20, $25, $27, $28
  .byte $2a, $27, $28, $24, $25, $27, $28, $2a
  .byte $25, $27, $22, $23, $23, $21, $20, $21
  .byte $20, $1e, $20, $22, $23, $25, $23, $22
  .byte $23, $22, $20, $22, $1e, $22, $25, $22
  .byte $1e, $22, $2a, $27, $23, $27, $2a, $27
  .byte $23, $27, $2a, $25, $22, $25, $2a, $25
  .byte $22, $25, $27, $23, $20, $23, $27, $23
  .byte $20, $23, $27, $22, $1e, $22, $27, $22
  .byte $1e, $22, $2f, $2c, $28, $2c, $2f, $2c
  .byte $28, $2c, $2f, $2a, $27, $2a, $2f, $2a
  .byte $27, $2a, $28, $2a, $2c, $2a, $28, $27
  .byte $25, $23, $22, $23, $25, $23, $22, $20
  .byte $1e, $1c, $1b, $1e, $1c, $20, $1e, $22
  .byte $20, $23, $22, $25, $23, $27, $25, $28
  .byte $27, $2a, $28, $2a, $28, $2a, $2c, $2a
  .byte $28, $27, $25, $27, $25, $27, $28, $2a
  .byte $28, $2a, $2c, $2a, $2c, $2a, $28, $27
  .byte $28, $27, $25, $27, $25, $27, $28, $2a
  .byte $28, $2a, $2c, $2a, $2c, $2a, $28, $27
  .byte $28, $27, $25, $27, $25, $27, $28, $27
  .byte $28, $25, $27, $2a, $28, $2c, $2a, $2e
  .byte $2c, $2f, $2e, $2c, $2a, $2f, $28, $27
  .byte $25, $23, $1e, $22, $23, $00

len1 =*                            ; note duration for voice 1
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $02, $02, $02, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $02, $01, $01
  .byte $02, $02, $02, $01, $01, $02, $02, $02
  .byte $02, $04, $01, $01, $01, $01, $02, $02
  .byte $02, $04, $02, $02, $01, $01, $02, $02
  .byte $01, $01, $01, $01, $02, $02, $02, $01
  .byte $01, $02, $02, $02, $02, $04, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $02, $02
  .byte $04, $04, $01, $01, $01, $01, $02, $04
  .byte $02, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $02, $04, $02, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $02, $02, $01, $01
  .byte $01, $01, $02, $02, $01, $01, $01, $01
  .byte $02, $02, $02, $02, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $02, $02, $02, $04, $02, $02, $01, $01
  .byte $02, $02, $01, $01, $01, $01, $02, $02
  .byte $02, $01, $01, $02, $02, $02, $02, $04
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
```

```
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $06, $02, $02, $02, $02, $02, $02, $02
   .byte $02, $02, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $02, $02, $02, $02, $02, $02
   .byte $04, $01, $01, $01, $01, $02, $02, $02
   .byte $04, $02, $02, $01, $01, $02, $02, $02
   .byte $02, $04, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $04, $02
   .byte $02, $02, $02, $02, $02, $02, $02, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $02, $02, $02, $04, $02
   .byte $02, $01, $01, $02, $02, $02, $02, $02
   .byte $02, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $04, $04, $04, $06, $01, $01, $02
   .byte $02, $04, $06, $01, $01, $02, $02, $04
   .byte $04, $02, $01, $01, $02, $02, $02, $04
   .byte $01, $01, $02, $04, $01, $01, $02, $04
   .byte $01, $01, $02, $02, $02, $02, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $02, $01
   .byte $01, $02, $02, $04, $03, $01, $08, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $02
   .byte $02, $02, $02, $01, $01, $01, $01, $02
   .byte $02, $02, $04, $02, $02, $01, $01, $02
   .byte $02, $02, $04, $02, $02, $01, $01, $02
   .byte $02, $02, $02, $04, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $02, $02, $02, $04, $02, $02, $01, $01
   .byte $02, $02, $02, $02, $02, $02, $02, $04
   .byte $02, $02, $04, $02, $01, $01, $01, $01
   .byte $02, $02, $02, $04, $02, $01, $01, $04
   .byte $02, $01, $01, $04, $02, $01, $01, $04
   .byte $02, $01, $01, $04, $02, $02, $01, $01
   .byte $02, $02, $02, $02, $06, $01, $01, $02
   .byte $02, $02, $02, $04, $02, $01, $01, $02
   .byte $01, $01, $02, $01, $01, $06, $01, $01
   .byte $02, $01, $01, $02, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $02, $01, $01, $02, $02, $04
   .byte $02, $02, $04, $04, $08, $00

 note2 =*                           ; note value for voice 2
   .byte $17, $17, $16, $16, $14, $14, $12, $12
   .byte $10, $10, $12, $12, $17, $0b, $0f, $0f
   .byte $14, $14, $0d, $0d, $11, $11, $12, $1b
   .byte $1d, $0d, $12, $0a, $0d, $12, $12, $10
```

```
.byte $10, $0f, $0f, $14, $14, $0d, $0d, $12
.byte $12, $17, $0b, $1b, $1e, $1b, $1e, $19
.byte $1e, $19, $1e, $17, $1b, $17, $1b, $16
.byte $1b, $16, $1b, $14, $17, $14, $17, $12
.byte $17, $12, $17, $19, $19, $19, $10, $12
.byte $12, $12, $0a, $0b, $0d, $0f, $10, $12
.byte $14, $16, $17, $25, $27, $25, $27, $28
.byte $27, $25, $23, $22, $00, $22, $00, $25
.byte $00, $25, $00, $28, $2a, $28, $2a, $25
.byte $27, $25, $27, $22, $27, $22, $27, $25
.byte $2a, $25, $2a, $28, $2a, $28, $2a, $25
.byte $27, $25, $27, $22, $27, $22, $27, $25
.byte $27, $28, $25, $23, $22, $23, $25, $2c
.byte $2a, $20, $1e, $20, $23, $1e, $1c, $1b
.byte $1e, $00, $23, $25, $27, $28, $27, $25
.byte $23, $2a, $28, $27, $28, $27, $28, $25
.byte $27, $28, $2a, $2c, $2a, $28, $27, $28
.byte $27, $28, $25, $27, $25, $23, $2f, $2a
.byte $27, $2a, $2f, $2a, $27, $2a, $2f, $2a
.byte $27, $2a, $2f, $2a, $27, $2a, $25, $2f
.byte $2e, $2c, $2a, $28, $27, $25, $23, $1e
.byte $17, $2a, $2c, $2a, $28, $27, $28, $2c
.byte $2c, $2a, $28, $27, $2a, $28, $27, $25
.byte $28, $27, $2a, $2a, $28, $27, $25, $28
.byte $27, $25, $23, $27, $25, $2a, $22, $23
.byte $25, $27, $25, $2a, $22, $23, $25, $27
.byte $25, $2a, $2a, $2a, $22, $25, $22, $25
.byte $2a, $25, $22, $25, $2a, $25, $22, $25
.byte $2a, $25, $2a, $2e, $2c, $2a, $29, $27
.byte $25, $23, $22, $20, $22, $1e, $20, $22
.byte $23, $25, $27, $29, $22, $2c, $2e, $2f
.byte $25, $23, $22, $25, $23, $22, $23, $22
.byte $23, $20, $22, $23, $25, $27, $25, $29
.byte $2a, $2c, $2e, $2f, $2c, $2e, $2c, $2a
.byte $16, $19, $16, $19, $14, $19, $14, $19
.byte $12, $16, $12, $16, $11, $16, $11, $16
.byte $0f, $12, $0f, $12, $0d, $12, $0d, $12
.byte $14, $14, $14, $14, $0d, $2c, $2e, $2c
.byte $2c, $2a, $2f, $2e, $2e, $2c, $2e, $2f
.byte $2e, $2c, $2c, $2e, $2c, $2a, $2c, $2e
.byte $2c, $2a, $2a, $2c, $2a, $29, $27, $29
.byte $27, $25, $23, $25, $23, $22, $1d, $22
.byte $1d, $22, $20, $25, $20, $25, $23, $25
.byte $23, $25, $20, $22, $20, $22, $1d, $22
.byte $1d, $22, $20, $25, $20, $25, $23, $25
.byte $23, $25, $20, $22, $20, $22, $1d, $22
.byte $1d, $22, $20, $25, $20, $25, $22, $22
.byte $20, $20, $22, $25, $1e, $27, $28, $2a
.byte $2c, $2a, $28, $27, $2a, $28, $27, $28
.byte $27, $28, $25, $27, $25, $23, $2f, $2a
.byte $27, $2a, $2f, $2a, $27, $2a, $2f, $2a
.byte $27, $2a, $2f, $2a, $27, $2a, $25, $2f
.byte $2e, $2c, $2a, $28, $27, $25, $27, $23
.byte $25, $27, $23, $23, $22, $22, $20, $25
.byte $23, $23, $22, $27, $28, $27, $25, $25
.byte $27, $25, $23, $25, $27, $25, $23, $23
.byte $25, $23, $22, $23, $25, $27, $28, $27
.byte $25, $23, $2f, $2e, $2c, $2e, $2c, $2e
.byte $2b, $2c, $27, $23, $27, $14, $14, $12
.byte $12, $10, $10, $0f, $0f, $0d, $0d, $0f
.byte $0f, $08, $00, $14, $14, $0f, $0f, $14
.byte $10, $0d, $10, $14, $10, $0d, $1e, $1e
.byte $19, $19, $1e, $1b, $17, $1b, $1e, $1b
.byte $17, $10, $10, $0b, $0b, $10, $0d, $0a
.byte $0d, $10, $0d, $0a, $1b, $1b, $16, $16
.byte $1b, $17, $14, $17, $1b, $17, $14, $17
.byte $1f, $19, $16, $19, $1f, $19, $16, $19
.byte $20, $1b, $17, $1b, $14, $0f, $0b, $0f
.byte $14, $10, $0d, $10, $19, $14, $10, $14
.byte $13, $13, $19, $19, $20, $20, $20, $20
.byte $22, $20, $1f, $17, $01, $20, $23, $20
.byte $23, $1e, $23, $1e, $23, $1c, $20, $1c
.byte $20, $1b, $20, $1b, $20, $19, $1c, $19
.byte $1c, $17, $1c, $17, $1c, $19, $19, $1b
.byte $1b, $17, $20, $21, $23, $25, $23, $21
.byte $20, $2c, $2a, $28, $2a, $28, $2a, $27
.byte $28, $2f, $2d, $2c, $2d, $2c, $2d, $2a
.byte $2c, $2a, $28, $14, $17, $14, $17, $14
.byte $19, $14, $19, $16, $19, $16, $19, $16
.byte $1b, $16, $1b, $0b, $0f, $0b, $0f, $0b
.byte $10, $0b, $10, $0d, $10, $0d, $10, $0d
.byte $12, $0d, $12, $23, $28, $2a, $2c, $27
.byte $25, $23, $2a, $28, $27, $28, $27, $28
.byte $25, $27, $28, $27, $25, $27, $25, $23
.byte $28, $27, $25, $27, $28, $2a, $2c, $2a
.byte $2a, $22, $23, $25, $1e, $1e, $1b, $1b
.byte $19, $16, $19, $18, $16, $18, $14, $14
.byte $19, $19, $1b, $1b, $20, $1b, $1c, $20
.byte $18, $19, $1b, $19, $14, $12, $14, $19
.byte $14, $10, $19, $1e, $16, $14, $16, $17
```

```
  .byte $12, $10, $12, $0b, $0d, $23, $21, $20
  .byte $21, $20, $1e, $20, $22, $23, $25, $23
  .byte $22, $23, $22, $20, $22, $1e, $22, $25
  .byte $22, $1e, $22, $12, $17, $12, $17, $12
  .byte $16, $12, $16, $0f, $14, $0f, $14, $0f
  .byte $12, $0f, $12, $0b, $10, $0b, $10, $0b
  .byte $0f, $0b, $0f, $25, $27, $28, $27, $25
  .byte $27, $25, $23, $1e, $20, $22, $20, $1e
  .byte $20, $1e, $1c, $17, $00, $1e, $00, $25
  .byte $27, $25, $27, $28, $27, $25, $23, $22
  .byte $27, $22, $27, $25, $2a, $25, $2a, $28
  .byte $2a, $28, $2a, $25, $27, $25, $27, $22
  .byte $27, $22, $27, $25, $2a, $25, $2a, $28
  .byte $2a, $28, $2a, $25, $27, $25, $27, $22
  .byte $27, $22, $27, $25, $27, $28, $25, $1e
  .byte $22, $23, $27, $23, $20, $1e, $20, $1e
  .byte $1c, $17, $00

len2 =*                           ; note duration for voice 2
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $04, $04, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $04, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $04, $04, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $02, $02, $04, $02, $01
  .byte $03, $02, $02, $01, $01, $02, $02, $02
  .byte $02, $04, $01, $01, $01, $01, $02, $02
  .byte $02, $04, $02, $02, $01, $01, $02, $02
  .byte $01, $01, $01, $01, $02, $02, $02, $01
  .byte $01, $02, $02, $02, $02, $04, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $02, $02
  .byte $04, $04, $01, $01, $01, $01, $02, $04
  .byte $02, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $02, $04, $02, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $02, $02, $01, $01
  .byte $01, $01, $02, $02, $01, $01, $01, $01
  .byte $02, $02, $02, $02, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $02, $02, $02, $04, $02, $02, $01, $01
  .byte $02, $02, $01, $01, $01, $01, $02, $02
  .byte $02, $01, $01, $02, $02, $02, $02, $04
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $06, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $02, $02
  .byte $02, $02, $02, $02, $04, $01, $01, $01
  .byte $01, $02, $02, $02, $04, $02, $02, $01
  .byte $01, $02, $02, $02, $02, $04, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $04, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $02
  .byte $02, $02, $04, $02, $02, $01, $01, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $04, $04, $02, $02, $02, $02, $01
  .byte $01, $01, $01, $01, $01, $02, $02, $02
  .byte $02, $02, $01, $01, $01, $01, $01, $01
  .byte $02, $02, $02, $02, $02, $01, $01, $01
  .byte $01, $01, $01, $02, $02, $02, $02, $02
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
```

```
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $02, $02, $02, $02, $04, $04, $04, $02
   .byte $02, $04, $04, $04, $04, $02, $02, $02
   .byte $02, $02, $02, $02, $02, $02, $02, $02
   .byte $02, $02, $02, $02, $02, $02, $02, $02
   .byte $02, $02, $02, $02, $02, $02, $02, $02
   .byte $02, $08, $01, $01, $01, $01, $02, $02
   .byte $02, $04, $02, $02, $01, $01, $02, $02
   .byte $02, $04, $02, $02, $01, $01, $02, $02
   .byte $02, $02, $04, $02, $02, $02, $02, $02
   .byte $02, $02, $02, $02, $02, $02, $02, $02
   .byte $02, $02, $02, $02, $02, $02, $02, $02
   .byte $02, $02, $02, $02, $02, $02, $02, $02
   .byte $02, $02, $02, $01, $01, $01, $01, $02
   .byte $02, $02, $04, $02, $02, $01, $01, $02
   .byte $02, $02, $02, $02, $02, $02, $04, $02
   .byte $02, $04, $02, $01, $01, $01, $01, $02
   .byte $02, $02, $04, $02, $02, $02, $02, $02
   .byte $02, $02, $01, $01, $01, $01, $02, $02
   .byte $02, $02, $02, $02, $02, $02, $02, $02
   .byte $02, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $02, $02, $02, $02, $01, $01, $01
   .byte $01, $01, $01, $02, $02, $02, $01, $01
   .byte $02, $01, $01, $02, $01, $01, $06, $01
   .byte $01, $02, $01, $01, $02, $01, $01, $01
   .byte $01, $01, $01, $02, $02, $02, $02, $02
   .byte $02, $02, $02, $02, $02, $02, $02, $02
   .byte $02, $02, $02, $02, $02, $02, $02, $02
   .byte $02, $02, $02, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $04, $04, $04, $04, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $02
   .byte $02, $04, $04, $02, $02, $04, $04, $04
   .byte $04, $08, $00

 note3 =*                          ; note value for voice 3
   .byte $0b, $0b, $0a, $0a, $08, $08, $06, $06
   .byte $04, $04, $06, $06, $0b, $0b, $03, $03
   .byte $08, $08, $0d, $0d, $05, $05, $06, $17
   .byte $19, $0d, $06, $0a, $0d, $06, $06, $04
   .byte $04, $03, $03, $08, $08, $01, $01, $06
   .byte $06, $0b, $0b, $17, $1b, $17, $1b, $16
   .byte $19, $16, $19, $14, $17, $14, $17, $12
   .byte $16, $12, $16, $10, $14, $10, $14, $0f
   .byte $12, $0f, $12, $0d, $0d, $0d, $04, $06
   .byte $06, $06, $00, $00, $01, $03, $04, $06
   .byte $08, $0a, $0b, $10, $10, $10, $10, $12
   .byte $12, $12, $12, $12, $12, $12, $12, $1e
   .byte $1e, $1e, $1e, $1e, $1e, $1e, $1e, $12
   .byte $12, $12, $12, $0b, $0d, $0f, $10, $12
   .byte $10, $0f, $10, $12, $10, $12, $06, $0b
   .byte $17, $23, $25, $27, $28, $27, $25, $23
   .byte $25, $27, $28, $27, $25, $23, $25, $23
   .byte $25, $22, $23, $25, $27, $28, $27, $25
   .byte $23, $25, $23, $25, $22, $23, $1e, $1b
   .byte $17, $17, $16, $16, $14, $14, $12, $12
   .byte $10, $10, $12, $12, $0b, $17, $27, $28
   .byte $27, $25, $23, $25, $28, $28, $27, $25
   .byte $23, $25, $22, $23, $27, $27, $25, $23
   .byte $22, $23, $20, $22, $25, $1e, $20, $22
   .byte $23, $22, $25, $1e, $20, $22, $23, $22
   .byte $22, $22, $22, $12, $12, $11, $11, $0f
   .byte $0f, $0d, $0d, $0b, $0b, $0d, $0d, $12
   .byte $0b, $0a, $08, $06, $00, $22, $20, $1e
   .byte $20, $1e, $20, $1d, $1e, $20, $22, $23
   .byte $22, $20, $22, $23, $25, $27, $29, $2a
   .byte $25, $1e, $12, $16, $12, $16, $11, $14
   .byte $11, $14, $0f, $12, $0f, $12, $0d, $11
   .byte $0d, $11, $0b, $0f, $0b, $0f, $0a, $0d
   .byte $0a, $0d, $08, $08, $08, $08, $0d, $29
   .byte $2a, $29, $29, $27, $2c, $2a, $2a, $29
   .byte $2a, $2c, $2a, $29, $29, $2a, $29, $27
   .byte $29, $2a, $29, $27, $27, $29, $27, $25
   .byte $23, $25, $23, $22, $20, $22, $20, $1e
   .byte $0d, $0d, $0d, $0d, $19, $19, $19, $19
   .byte $0d, $0d, $0d, $0d, $0d, $0d, $0d, $0d
   .byte $19, $19, $19, $19, $12, $1b, $17, $19
   .byte $12, $23, $25, $27, $28, $27, $25, $23
   .byte $27, $25, $23, $25, $23, $25, $22, $23
   .byte $1e, $1b, $17, $17, $16, $16, $14, $14
   .byte $12, $12, $10, $10, $12, $12, $0b, $1b
   .byte $1c, $1e, $17, $00, $23, $25, $23, $22
   .byte $22, $23, $22, $20, $22, $23, $22, $20
   .byte $20, $22, $20, $1f, $20, $22, $23, $25
```

```
  .byte $23, $22, $20, $27, $25, $23, $25, $23
  .byte $25, $22, $23, $20, $1b, $17, $14, $14
  .byte $12, $12, $10, $10, $0f, $0f, $0d, $0d
  .byte $0f, $0f, $14, $00, $0c, $0c, $0c, $0c
  .byte $00, $16, $16, $16, $16, $00, $14, $14
  .byte $14, $14, $00, $13, $13, $13, $13, $00
  .byte $00, $00, $00, $0f, $0f, $0d, $0d, $0b
  .byte $14, $12, $12, $10, $10, $0b, $0d, $0f
  .byte $0d, $0f, $0f, $08, $00, $1c, $20, $1c
  .byte $20, $1b, $1e, $1b, $1e, $19, $1c, $19
  .byte $1c, $17, $1b, $17, $1b, $15, $19, $15
  .byte $19, $14, $17, $14, $17, $15, $15, $15
  .byte $15, $10, $1c, $1e, $20, $21, $20, $1e
  .byte $1c, $23, $21, $20, $21, $20, $21, $1e
  .byte $20, $2c, $2a, $28, $2a, $28, $2a, $27
  .byte $28, $23, $20, $17, $10, $14, $10, $14
  .byte $11, $14, $11, $14, $12, $16, $12, $16
  .byte $13, $16, $13, $16, $14, $0b, $14, $0b
  .byte $14, $0b, $14, $0b, $15, $0d, $15, $0d
  .byte $16, $0d, $16, $0d, $17, $00, $27, $25
  .byte $23, $25, $23, $25, $22, $23, $25, $23
  .byte $22, $23, $22, $20, $25, $23, $22, $23
  .byte $25, $27, $28, $27, $23, $1e, $20, $22
  .byte $17, $17, $17, $17, $12, $12, $12, $0d
  .byte $0d, $10, $10, $12, $12, $18, $18, $19
  .byte $20, $12, $00, $00, $16, $00, $12, $00
  .byte $00, $0f, $12, $17, $0f, $10, $14, $17
  .byte $10, $11, $19, $14, $11, $12, $0d, $0b
  .byte $06, $0b, $0f, $0b, $0f, $16, $0d, $16
  .byte $0d, $14, $0b, $14, $0b, $12, $0a, $12
  .byte $0a, $10, $14, $10, $14, $0f, $12, $0f
  .byte $12, $0d, $0d, $0d, $10, $12, $12, $12
  .byte $0a, $0b, $0d, $0f, $10, $12, $14, $16
  .byte $17, $10, $10, $10, $10, $12, $12, $12
  .byte $12, $12, $12, $12, $12, $12, $12, $12
  .byte $12, $12, $12, $12, $12, $12, $12, $12
  .byte $12, $0b, $0d, $0f, $10, $12, $10, $0f
  .byte $10, $12, $10, $12, $12, $0b, $00

len3 =*                           ; note duration for voice 3
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $04, $04, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $04, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $04, $04, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $04
  .byte $04, $01, $01, $01, $01, $02, $02, $01
  .byte $01, $01, $01, $02, $02, $02, $01, $01
  .byte $02, $02, $01, $01, $01, $01, $02, $02
  .byte $02, $01, $01, $02, $02, $02, $02, $04
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $04, $04, $04, $01
  .byte $01, $01, $01, $02, $04, $02, $01, $01
  .byte $02, $02, $02, $02, $04, $02, $01, $01
  .byte $02, $02, $02, $02, $02, $01, $01, $01
  .byte $01, $02, $02, $01, $01, $01, $01, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $04, $06, $04, $02, $02
  .byte $01, $01, $02, $02, $01, $01, $01, $01
  .byte $02, $02, $02, $01, $01, $02, $02, $02
  .byte $02, $04, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $06, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $08, $01, $01, $01, $01, $02, $02, $02
  .byte $04, $02, $02, $01, $01, $02, $02, $02
  .byte $02, $04, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $05, $01
  .byte $01, $01, $04, $0c, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $02, $02, $02, $04, $02, $02, $01, $01
  .byte $02, $02, $02, $02, $02, $02, $02, $02
```

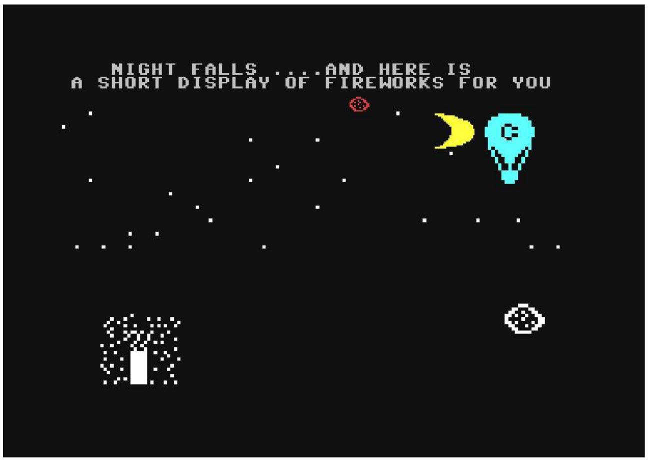
```
.byte $02, $02, $02, $02, $02, $02, $02, $02
.byte $02, $02, $04, $04, $02, $02, $02, $02
.byte $08, $02, $02, $02, $02, $08, $02, $02
.byte $02, $02, $08, $02, $02, $02, $02, $08
.byte $08, $08, $08, $02, $02, $02, $02, $02
.byte $02, $02, $02, $02, $02, $02, $02, $02
.byte $02, $02, $02, $04, $04, $02, $02, $02
.byte $02, $02, $02, $02, $02, $02, $02, $02
.byte $02, $02, $02, $02, $02, $02, $02, $02
.byte $02, $02, $02, $02, $02, $02, $02, $02
.byte $02, $08, $01, $01, $01, $01, $02, $02
.byte $02, $04, $02, $02, $01, $01, $02, $02
.byte $02, $04, $02, $02, $01, $01, $02, $02
.byte $02, $02, $02, $02, $02, $02, $02, $02
.byte $02, $02, $02, $02, $02, $02, $02, $02
.byte $02, $02, $02, $02, $02, $02, $02, $02
.byte $02, $02, $02, $02, $02, $02, $02, $02
.byte $02, $02, $02, $02, $04, $06, $04, $02
.byte $02, $01, $01, $02, $02, $02, $02, $02
.byte $02, $02, $04, $02, $02, $04, $02, $01
.byte $01, $01, $01, $02, $02, $02, $04, $02
.byte $02, $02, $02, $02, $02, $02, $04, $02
.byte $02, $02, $02, $02, $02, $02, $02, $02
.byte $02, $02, $02, $08, $02, $02, $02, $02
.byte $08, $02, $02, $02, $02, $02, $02, $02
.byte $02, $02, $02, $02, $02, $02, $02, $02
.byte $02, $02, $02, $02, $02, $02, $02, $02
.byte $02, $02, $02, $02, $02, $02, $02, $02
.byte $02, $02, $02, $02, $02, $02, $02, $02
.byte $02, $02, $02, $02, $02, $02, $02, $02
.byte $02, $02, $02, $02, $02, $02, $02, $02
.byte $02, $02, $02, $02, $02, $02, $02, $02
.byte $02, $02, $02, $02, $02, $02, $02, $02
.byte $02, $02, $02, $02, $02, $02, $02, $02
.byte $02, $02, $02, $02, $02, $02, $02, $02
.byte $02, $04, $04, $04, $04, $08, $00
```

At this point we can say that the rip could be done directly in the monitor, without generating a source of the rip. However, as the code is in low and the data is in high memory, we will obtain a big file (without a relocation of the player code).

As we present the code, just a few words about the player. As you see, each voice have his control register and the address of note pitch and duration stored in a local table. Two extra bytes are used for controlling the velocity of the reproduction that are set by the BASIC program during the initialization phase.

No special effect are used for each instrument: however the sound is pretty good.

Testcard


NIGHT FALLS ....AND HERE IS
A SHORT DISPLAY OF FIREWORKS FOR YOU

The Testcard program is probably one of the first demo ever done. It has sprites animation and music, all created by a BASIC program.

If you listen to the sound, it is quite simple: nothing comparable to Sheba.

This is due to the use of BASIC even for the sound generation of the two voice used together. However the code still use a portion of machine code routine, but while in Sheba the machine code manages both note pitch and duration, here the machine code manage only pitch generation, while duration is done using TI variable by the BASIC program.

Before goes ahead, it is the time to get a look at the BASIC source: the bold part it the one that involve the sound generation.

BASIC code

```
5 REM space for sprites                                                     x
10 REM space for sprites                                                     x
15 REM space for sprites                                                     x
20 REM space for sprites                                                     x
25 REM space for sprites                                                     x
30 REM space for sprites                                                     x
35 REM space for sprites                                                     x
40 REM space for sprites                                                     x
45 REM space for sprites                                                     x
50 REM space for sprites                                                     x
55 REM space for sprites                                                     x
60 REM space for sprites                                                     x
65 REM space for sprites                                                     x
70 REM space for sprites                                                     x
75 REM space for sprites                                                     x
90 DIM sp(10),n(60),v%(500,3),w%(500,3),c$(16)
95 GOTO 10210:skipround music player AND main loop which are time-critical
100 REM play a tune(nv voices,kk notes in part 1
105 FOR j=1 TO nv:p(j)=0:d(j)=0:t(j)=ti:NEXT j
110 FOR j=1 TO nv
115 IF t(j)>ti THEN 130
120 t(j)=t(j)+d(j):x=v%(p(j),j)
125 y=w%(p(j),j):p(j)=p(j)+1:d(j)=y*dv:m(j)=n(x)
130 NEXT j
140 f=0
145 FOR j=1 TO nv
150 IF t(j)<=ti THEN sp(j+7)=m(j):f=1
155 NEXT j
160 IF f=0 THEN 145
165 IF p(1)<=kk THEN 110
170 RETURN
500 GOTO 5000 :REM pointer to moved code
2000 REM golden rain(1)
2005 sp(7)=a+64080
2010 ON cw GOTO 2020,2040,2045,2050
2020 IF RND(0)>0.04 THEN 2100
2030 cw=2:cn=50:POKE 2040,34:POKE v+21,PEEK(v+21)+1:POKE v+39,1+7*RND(0)
2035 sp(10)=no:GOTO 2100
2040 cw=3:POKE 2040,35:GOTO 2100
2045 cw=4:POKE 2040,36:GOTO 2100
2050 cw=2:POKE 2040,34:cn=cn-1
2060 IF cn=0 THEN cw=1:POKE v+21,PEEK(v+21)-1
2100 REM ground explosions
2110 ON gr GOTO 2120,2140,2145,2150
2120 IF RND(0)>0.1 THEN 2540
2130 gr=2:POKE 2041,37:POKE v+21,PEEK(v+21)+2:POKE v+40,1+7*RND(0)
2135 sp(10)=no:GOTO 2540
2140 gr=3:POKE 2041,38:GOTO 2540
2145 gr=4:POKE 2041,39:GOTO 2540
2150 gr=1:POKE v+21,PEEK(v+21)-2
2540 REM rocket(2)
2610 ON qa GOTO 2620,2670,2690,2710,2730
2620 IF RND(0)>0.05 OR ba=4 THEN 2740
2630 qy=250:qx=INT(350*RND(0)):qv=-19
2640 qh=INT((350-qx-300*RND(0))/20):POKE 2045,33:POKE v+44,1+7*RND(0)
2650 sp(5)=a+256*qx+qy
2660 POKE v+21,PEEK(v+21)+32:qa=2:GOTO 2740
2670 qx=qx+qh:qy=qy+qv:qv=qv+1
2680 sp(5)=a+256*qx+qy:IF qv=3 THEN qa=3:sp(10)=no:qn=2:POKE 2045,37:GOTO 2740
2690 qn=qn-1:IF qn=0 THEN qn=3:qa=4:POKE 2045,38
2700 GOTO 2740
2710 qn=qn-1:IF qn<>0 THEN GOTO 2740
2715 qn=5:qa=5:POKE 2045,39:GOTO 2740
2720 sp(5)=sp(5)-11*257:GOTO 2740
2730 qn=qn-1:IF qn<>0 THEN 2740
2732 qa=1:POKE v+21,PEEK(v+21)-32
```

```
2735 POKE v+23,PEEK(v+23)AND 223:POKE v+29,PEEK(v+29)AND 223
2740 REM balloon
2810 ON ba GOTO 2820,2850,2870,2882
2820 IF RND(0)>0.02 THEN 3000
2830 sp(6)=a+350*256+80:bp=350 :ba=2:POKE 2046,42 :POKE v+23,PEEK(v+23)+64
2835 POKE v+29,PEEK(v+29)+64
2840 POKE v+45,1+7*RND(0):POKE v+21,PEEK(v+21)+64:GOTO 3000
2850 bp=bp-1:sp(6)=sp(6)-256
2860 IF bp=0 THEN POKE v+21,PEEK(v+21)-64:ba=1
2865 IF bp=0 THEN POKE v+23,PEEK(v+23)-64:POKE v+29,PEEK(v+29)-64
2867 GOTO 3000
2870 bn=bn-1:IF bn<>0 THEN 3000
2880 ba=4:POKE 2046,41:POKE v+45,1:bp=80:POKE v+21,PEEK(v+21)+64:GOTO 3000
2882 s1=0:s2=0:IF t1<tq THEN s1=n(v%(p1,1)):t1=t1+dv*w%(p1,1):p1=p1+1
2884 IF t2<tq THEN s2=n(v%(p2,2)):t2=t2+dv*w%(p2,2):p2=p2+1
2886 sp(8)=s1:sp(9)=s2
2890 bp=bp+1:IF bp=213 THEN POKE v+21,PEEK(v+21)-64:ba=1
2900 sp(6)=sp(6)+1
3000 p=PEEK(v+30)AND 96:IF(p<>96)OR(ba<>2)OR(qa<>2)THEN 3500
3010 REM rocket/balloon collision
3015 p1=0:p2=0:t1=tq+5:t2=t1
3020 qa=3:qn=2:POKE 2045,39:sp(10)=no:bn=8:ba=3:POKE v+21,PEEK(v+21)-64
3030 sp(6)=sp(6)+1028:POKE v+23,PEEK(v+23)-64:POKE v+29,PEEK(v+29)-64:GOTO 2000
3500 tq=tq+1:POKE v+21,PEEK(v+21)OR 128:IF tq<500 OR  ba<>1 THEN 2000
3505 FOR j=1 TO 1000:NEXT j
3510 FOR j=0 TO 10:sp(j)=0:POKE vv+24,0
3515 POKE v+21,0
3520 PRINT"<CLR>     finally, two pieces  "
3525 PRINT
3530 PRINT"     by  wolfgang amadeus"
3540 PRINT"           m o z a r t"
3550 PRINT:PRINT"  first,   andante"
3560 GOSUB 5040:GOSUB 100
3570 PRINT:PRINT"  next, an  allegro "
3580 GOSUB 5040:GOSUB 100
3590 RESTORE:
3600 PRINT:PRINT "the whole program will now be repeated"
3700 FOR j=832 TO 1000:READ a:NEXT j
3710 POKE vv+4,0:POKE vv+11,0:POKE vv+18,0:POKE vv+24,0
3720 GOTO 10290
4000 c$(1)="   <BLACK><RVS ON>               <RVS OFF><WHITE> black"
4010 c$(2)="   <WHITE><RVS ON>               <RVS OFF><WHITE> white"
4030 c$(3)="   <RED><RVS ON>               <RVS OFF><WHITE> red"
4040 c$(4)="   <CYAN><RVS ON>               <RVS OFF><WHITE> cyan"
4050 c$(5)="   <PURPLE><RVS ON>               <RVS OFF><WHITE> purple"
4060 c$(6)="   <GREEN><RVS ON>               <RVS OFF><WHITE> green"
4070 c$(7)="   <BLUE><RVS ON>               <RVS OFF><WHITE> blue"
4080 c$(8)="   <YELLOW><RVS ON>               <RVS OFF><WHITE> yellow"
4090 c$(9)="   <ORANGE><RVS ON>               <RVS OFF><WHITE> orange"
4100 c$(10)="   <BROWN><RVS ON>               <RVS OFF><WHITE> brown"
4110 c$(11)="   <LIGHT RED><RVS ON>               <RVS OFF><WHITE> light red"
4120 c$(12)="   <DARK GREY><RVS ON>               <RVS OFF><WHITE> dark gray"
4130 c$(13)="   <GREY><RVS ON>               <RVS OFF><WHITE> medium gray"
4140 c$(14)="   <LIGHT GREEN><RVS ON>               <RVS OFF><WHITE> light green"
4150 c$(15)="   <LIGHT BLUE><RVS ON>               <RVS OFF><WHITE> light blue"
4160 c$(16)="   <LIGHT GREY><RVS ON>               <RVS OFF><WHITE> light gray"
4170 POKE 53280,1:POKE 53281,6:POKE 208*256+21,0
4180 PRINT"<CLR> hello. welcome to the <RVS ON>cbm commodore 64<RVS OFF>"
4190 PRINT
4200 PRINT" here are some bars of colour to help"
4210 PRINT"        adjust your set"
4220 PRINT
4230 GOTO 10300
5000 REM set up notes in n
5010 a=2^31+33*2^16:q=720:p=2^(1/12)
5020 FOR jj=1 TO 60: n(jj)=a+q:q=q*p:NEXT jj
5030 REM enter here if n already set up
5040 vv=212*256:n(5)=a
5050 READ nv,ns,at,d,s,dv
5060 FOR jj=1 TO nv:pp(jj)=0:NEXT jj
```

```
5065 FOR kk=1 TO ns
5070 READ dn
5075 IF dn>1 THEN FOR jj=1 TO nv:pq(jj)=pp(jj):NEXT jj
5078 FOR jj=1 TO nv
5080 READ a$:READ b$
5090 FOR tt=1 TO LEN(a$)
5100 v%(pp(jj),jj)=ASC(MID$(a$,tt,1))-35
5110 w%(pp(jj),jj)=ASC(MID$(b$,tt,1))-48:pp(jj)=pp(jj)+1
5120 NEXT tt,jj
5121 IF dn=1 THEN 5129 :REM copy repeated bar
5122 FOR jj=1 TO nv
5123 pr=pp(jj)
5124 v%(pp(jj),jj)=v%(pq(jj),jj):w%(pp(jj),jj)=w%(pq(jj),jj)
5125 pq(jj)=pq(jj)+1:pp(jj)=pp(jj)+1
5126 IF pq(jj)<pr THEN 5124
5127 NEXT jj
5129 NEXT kk
5130 kk=pp(1)-1
5140 FOR jj=0 TO nv-1
5150 POKE vv+7*jj+5,16*at+d
5160 POKE vv+7*jj+6,16*s
5170 NEXT jj
5180 POKE vv+24,15
5190 RETURN
6100 REM machine code routine to service sprites  and sound generators
6110 REM it updates their position every interrupt
6120 DATA 165,47,133,251,165,48,133,252,160,0,177,251,201,83,208,8,200,177,251
6130 DATA 201,80,240,38,136,200,200,177,251,133,253,200,177,251,133,254,24,165,251
6140 DATA 101,253,133,251,165,252,101,254,133,252,197,50,208,212,165,251
6150 DATA 197,49,208,206,76,49,234
6160 DATA 152,24,105,6,168,162,0,200,200,24,177,251,41,1,240,1,56,102,255,200
6170 DATA 177,251,157,0,208,200,177,251,157,1,208,232,232,200,224,16,208,225
6180 DATA 165,255,141,16,208
6182 DATA 162,0,200,200,177,251,240,29,169,0,157,4,212,177,251,157,4,212,169,0
6184 DATA 145,251,200,177,251,157,1,212,200,177,251,157,0,212,76,208,3
6186 DATA 200,200,200,138,24,105,7,170,224,21,208,209,76,49,234
6190 REM insert wedge (code at 988)
6200 DATA 120,169,64,141,20,3,169,3,141,21,3,88,96
6205 REM sprite data- bright dot
6210 DATA 0,0,0,0,0,0,0,0,0
6215 DATA 0,0,0,0,0,0,0,0,0
6220 DATA 0,0,0,0,0,0,0,24,0
6225 DATA 0,60,0,0,60,0,0,24,0
6230 DATA 0,0,0,0,0,0,0,0,0
6235 DATA 0,0,0,0,0,0,0,0,0
6240 DATA 0,0,0,0,0,0,0,0,0
6300 REM golden rain(1)
6310 DATA 0,32,0,8,132,136,18,0,2
6315 DATA 32,162,164,9,16,146,48,105,40
6320 DATA 64,146,2,0,170,160,148,100,0
6325 DATA 0,90,2,72,36,144,0,124,8
6330 DATA 66,124,32,0,124,0,144,124,4
6335 DATA 66,124,136,0,124,0,32,124,36
6340 DATA 0,124,0,77,125,18,0,124,0
6350 REM golden rain(2)
6360 DATA 0,64,0,17,2,68,36,0,1
6365 DATA 65,33,82,16,0,0,33,105,8
6370 DATA 16,146,32,1,18,2,4,36,160
6375 DATA 144,72,0,0,36,2,72,124,144
6380 DATA 0,124,8,66,124,33,0,124,0
6385 DATA 144,124,4,66,124,136,32,124,0
6390 DATA 0,124,4,5,124,0,72,125,18
6400 REM golden rain(3)
6410 DATA 0,128,0,34,1,34,72,64,4
6415 DATA 130,16,161,16,128,4,2,36,128
6420 DATA 16,146,8,32,8,32,2,164,2
6425 DATA 4,72,160,144,36,0,0,124,2
6430 DATA 72,124,144,0,124,8,66,124,33
6440 DATA 0,124,0,144,124,4,66,124,136
```

```
6445 DATA 32,124,0,0,124,4,5,124,0
6450 REM explosion(1)
6455 DATA 0,0,0,0,0,0,0,0,0
6460 DATA 0,0,0,0,0,0,0,0,0
6465 DATA 0,0,0,0,56,0,0,198,0
6470 DATA 1,147,0,3,73,128,2,16,192
6475 DATA 3,66,192,1,137,128,0,195,0
6480 DATA 0,124,0,0,0,0,0,0,0
6485 DATA 0,0,0,0,0,0,0,0,0
6500 REM explosion(2)
6505 DATA 0,0,0,0,0,0,0,0,0
6510 DATA 0,0,0,0,16,0,0,16,128
6515 DATA 8,57,0,4,198,0,3,33,128
6520 DATA 6,0,192,12,72,224,12,4,120
6525 DATA 62,0,96,6,16,192,3,36,192
6530 DATA 5,129,160,8,254,16,0,48,0
6535 DATA 0,32,0,0,32,0,0,0,0
6550 REM explosion(3)
6555 DATA 0,16,0,0,16,0,0,16,16
6560 DATA 73,16,32,32,17,64,18,20,128
6565 DATA 8,17,64,68,2,16,2,84,0
6570 DATA 17,9,8,32,0,32,0,72,127
6575 DATA 254,0,0,0,2,4,18,32,72
6580 DATA 4,36,32,8,160,144,18,32,8
6585 DATA 32,33,4,64,32,2,0,32,0
6600 REM moon
6605 DATA 252,0,0,63,240,0,15,252,0
6610 DATA 7,255,128,3,255,240,1,255,248
6615 DATA 0,255,252,0,127,254,0,63,254
6620 DATA 0,63,255,0,63,255,0,63,255
6625 DATA 0,63,255,0,127,254,0,127,254
6630 DATA 0,255,252,1,255,248,3,255,224
6635 DATA 7,255,128,31,248,0,254,0,0
6650 REM parachute
6655 DATA 0,254,0,15,255,224,62,251,248
6660 DATA 126,250,252,254,26,254,254,216,126
6665 DATA 126,30,252,127,255,252,32,0,8
6670 DATA 16,0,16,8,0,32,4,0,64
6675 DATA 2,0,128,1,1,0,0,130,0
6680 DATA 0,84,0,0,124,0,0,56,0
6685 DATA 0,40,0,0,108,0,0,0,0
6700 REM commodore balloon
6705 DATA 0,127,0,1,255,192,3,255,224
6710 DATA 3,231,224,7,217,240,7,223,240
6715 DATA 7,217,240,3,231,224,3,255,224
6720 DATA 3,255,224,2,255,160,1,127,64
6725 DATA 1,62,64,0,156,128,0,156,128
6730 DATA 0,73,0,0,73,0,0,62,0
6735 DATA 0,62,0,0,62,0,0,28,0
7500 REM gavotte by g.f.handel
7502 DATA 3,2,0,10,0,15
7505 DATA 1
7510 DATA"(onoqojsqstsqovtsqoqonoqojsqstsqovtsqoq"
7520 DATA"811112211112222222241111221111222222224"
7530 DATA"((jjlolnolij(jjlolnolij"
7540 DATA"88442211222484422112224"
7550 DATA"(7;>cbcecb@gec<@><;97;>cbcecb@gec<@>"
7560 DATA"842211112222222211114221111222222224"
7565 DATA 2
7570 DATA"jhglhgejgechgecbcbcec>hghjgeghejcbcegec"
7580 DATA"211421142112222411112211111111221111224"
7590 DATA"cgece(ecbc(cc@>(((ccbcecb;"
7600 DATA"42112221122222288621111224"
7610 DATA";7496427;479;><;9;79bc7><;@<>27"
7620 DATA"4224211222222222222222211222224"
8000 REM marines march
8010 DATA 2,2,0,12,0,1
8015 DATA 1
8020 DATA"(aehhhhhmhefhhfcaaehhhhhmhefhhfcaml"
8030 DATA"81122223121122226112222312112222611"
```

```
8040 DATA"(5599(9(9(9(9999(44(4(4(459999(99(9(9(9(9999(44(4(4(459999("
8050 DATA"81111111111111122111111111111221111111111112211111111111122"
8055 DATA 1
8060 DATA"jfjfhjhmljfjmhaehhhhhmhefhhfca"
8070 DATA"222231211222261122223121122227"
8080 DATA"(:(5(:(5(9999((:(5(:(5(9999(99(9(9(9(9999(44(4(4(459999"
8090 DATA"1111111111112211111111111122111111111111221111111111113"
8500 REM andante
8510 DATA 2,2,3,11,0,17
8515 DATA 2
8520 DATA"(momoqmhhhoqoqrohhhtlmolmvsomltqvtsst("
8530 DATA"44111122224111122224111141111111122422"
8540 DATA"(eheaeheheahmlmolhlololhe(hjghmoooh("
8550 DATA"444111111114111111111111224442222422"
8595 DATA 2
8600 DATA"tqrtvroqrtrqoqmrqo(rqomltrqomjvromllm("
8610 DATA"41111411111111224224111141111111122422"
8620 DATA"(qnj(olhmljomaeah(olh(emj(gh((a("
8630 DATA"22222222222211112422222223122422"
8700 REM allegro
8710 DATA  2,2,1,9, 0,10
8715 DATA 2
8720 DATA"(qtrqom(mooooqm(mooooqmoqrtvxyvqst(tvxyvstvstvxytxvtvxyvstvst("
8730 DATA"82222242222211422222112111111122222422111111111111111111111142"
8740 DATA"(mqomhe(ehhhhe(ehhhhe(mlolomomololomomol("
8750 DATA"82222242222224222222448222222222222222242"
8795 DATA 2
8800 DATA"htqvtrotrqmrqomt(qtrqomoqrtvxyxmoqrtqomxy(t(r(o(m(hmmmhqqqmtoqm("
8810 DATA"2626262626444424222222111111118111111118222222224222222222243142"
8820 DATA"(amjomlhmlkjhfgh(mqomhe(qrotrq(qrotrq(q(o(h(e(<aaa<eeeah<a("
8830 DATA 426262626244444422222242222222422222222222224222222222224442""
10210 FOR j=832 TO 1000
10220 READ a:POKE j,a:NEXT j
10230 SYS(988)
10240 REM make space for sprites
10250 a=PEEK(43)+256*PEEK(44)
10260 IF a> 2048+14*64 THEN 10290
10270 POKE 43,PEEK(a):POKE 44,PEEK(a+1):GOTO 10250
10290 GOTO 4000
10300 FOR j=1 TO 10:PRINT c$(j):  q0=2048+64*j:GOSUB 11000:NEXT j
10305 FOR j=11 TO 16:PRINT c$(j):FOR k=1 TO 200:NEXT k,j
10310 v=208*256
10312 PRINT" now turn up the volume ... and wait"
10315 GOSUB 5000:GOSUB 100
10318 FOR k=1 TO 2000:NEXT k
10320 POKE 53280,0:POKE 53281,0:PRINT"<CLR>"
10330 POKE vv+24,0
10340 PRINT "<CLR>  <STOP><LIGHT GREY>  night falls ....and here is"
10350 PRINT" a short display of fireworks for you"
10360 FOR j=1 TO 30:POKE 1134+460*RND(0),46:FOR k=1 TO 100:NEXT k,j
10400 REM set up main loop
10405 sp(8)=2^31+2^16:sp(9)=2^31+2^16
10410 a=2^31:vv=212*256:POKE vv+19,12:POKE vv+20,0
10420 sp(0)=a+50*256+200:sp(1)=a+280*256+180
10430 sp(7)=a+250*256+80:POKE v+46,7:POKE 2047,40:POKE v+21,128
10440 no=2^31+129*2^16+5000:POKE vv+24,15
10450 cw=1:gr=1:qa=1:ba=1:GOSUB 5040:tq=0
10460 POKE 2042,33:POKE 2043,33:
10470 POKE v+23, 3:POKE v+29, 3
10480 GOTO 2000
11000 REM set up a sprioite in location q0
11010 FOR jj=0 TO 62
11020 READ kk
11030 POKE q0+jj,kk
11040 NEXT jj
11050 RETURN
```

Machine code rip

How we can produce a rip of this kind of BASIC program? I must admit that I did not have any idea of how to do it, as the main point needed for a rip is not present: the IRQ routine.

As you probably know, all the players that have not a full C64 environment emulation must use a IRQ calling for generating the sound. This is exactly what a standard PSID file use in the play section.

Without the hint of Warren P. I probably wasn't able to produce the rip. Even if I did not ask the permission to Warren to put this peace of email here, I think that this will can help everyone that would rip a BASIC program, so I write it anyhow:

> [...] in Testcard, the machine code only set up and play notes, while duration are controlled
> by the basic program (using TI variables). [....]

It would be difficult to do. However, as you've stated that the basic program uses TI variables, it could possibly be done. What you would have to do is convert those TI values to machine code data, and have the data read by a machine code routine so it knows to wait X number of cycles before progressing on to the next note. This is definitely not very easy.

Ok now it is more clear. What we have to do, it is to have an IRQ routine that “simulate” the TI timing before apply the needed note values. We else are very lucky, as the code in Sheba is exactly what we need: it manages the note as done in Testcard and else have a note duration controlled by IRQ: the minor change is to set the velocity parameter to be correctly set as a TI timing.

```
;=================================================
; Testcard demo for introduction to basic
; By Andew Colin (1982)
;
; The originale song use a Basic/machine
; code for generating music
; This is a machine code port
; The code can generate a PRG or SID file.
; Note: the original code don't use the IRQ
; routine. However, the engine used is a
; minor version of one used in  "Return of Sheba",
; so it is taken to be used here
;=================================================

.ifdef sid
 .byte "PSID"
 .word $0200         ; version 2
 .word $7C00         ; data offset
 .word $0000         ; load address in cbm format
 .byte >initMusic
 .byte <initMusic
 .byte >playMusic
 .byte <playMusic
 .word $0400         ; 4 songs
 .word $0100         ; default song
 .word $FFFF
 .word $FFFF
 .byte "Testcard",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
 .byte "Andrew Colin",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
 .byte "1982 Commodore Inc",0,0,0,0,0,0,0,0,0,0,0,0,0,0
 .word $0000
 .word $0000
 .word $0000
.endif

.org $0801

 ;=================================
 ; PRG start address
 ;=================================
 .byte $01,$08

 ;=================================
 ; BASIC startup program
 ;=================================
 .byte $0b,$08,$e8,$03,$9e,"2061",0,0,0

.org 2061
     jsr initMusic

     sei
     lda #<interr
     sta $0314
     lda #>interr
     sta $0315

     cli

aaa:
 jmp aaa

interr =*
     lda  contr+0
     bne  playMusic
kernal:
     jmp  $ea31

playMusic:
     ldx #$00
     lda instr+0,x                ; Voice 1: Control registers
     beq nend
     inc instr+5,x
     bne nend
     inc instr+6,x
     bne nend
     lda instr+1,x                ; Voice 1: notes address lo
     sta $fe
     lda instr+2,x                ; Voice 1: notes address hi
     sta $ff
     ldy #$00
     lda ($fe),y
     asl
     tay
     lda contr+2,y
     sta $d401,x                  ; Voice 1: Frequency control (hi byte)
     lda contr+3,y
     sta $d400,x                  ; Voice 1: Frequency control (lo byte)
     lda instr+3,x                ; Voice 1: notes duration address lo
     sta $fe
     lda instr+4,x                ; Voice 1: notes duration address hi
     sta $ff
     ldy #$00
     lda ($fe),y
     bne skip
     sta instr+0,x                ; Voice 1: Control registers
     jmp nend
skip:
     tay
loop:
     lda instr+5,x
     sec
     sbc contr+1
     sta instr+5,x
     lda instr+6,x
     sbc #$00
     sta instr+6,x
     dey
     bne loop
     inc instr+1,x                ; Voice 1: notes address lo
     bne skip1
     inc instr+2,x                ; Voice 1: notes address hi
skip1:
     inc instr+3,x                ; Voice 1: notes duration address lo
     bne skip2
     inc instr+4,x                ; Voice 1: notes duration address hi
skip2:
     lda #$00
     sta $d404,x                  ; Voice 1: Control registers
     lda instr+0,x                ; Voice 1: Control registers
     sta $d404,x                  ; Voice 1: Control registers
nend:
     txa
     clc
     adc #$07
     tax
     cmp #$15
     bne skip3
.ifdef sid
     rts
.endif
     jmp kernal
skip3:
     jmp playMusic+2
```

```
initMusic =*
                                  ; initialize control registers
     pha                          ; save tune number

                                  ; erase all sid registers
     ldx  #24
     lda  #0
loop1:
     sta  $D400,x
     dex
     bpl  loop1

     pla                          ; tune number
     asl a
     asl a
     asl a
     asl a
     tax

     lda songs,x
     sta contr+1                  ; velocity
     inx

     lda  songs,x
     sta  $D405                   ; Generator 1: Attack/Decay (voice 1)
     sta  $D40C                   ; Generator 1: Attack/Decay (voice 2)
     sta  $D413                   ; Generator 1: Attack/Decay (voice 3)
     inx
     lda  songs,x
     sta  $D406                   ; Generator 1: Sustain/Release (voice 1)
     sta  $D40D                   ; Generator 1: Sustain/Release (voice 2)
     sta  $D414                   ; Generator 1: Sustain/Release (voice 3)
     inx

     lda  #$0F
     sta  $D418                   ; volume

     lda  songs,x
     sta  instr+1
     inx
     lda  songs,x
     sta  instr+2
     inx

     lda  songs,x
     sta  instr+3
     inx
     lda  songs,x
     sta  instr+4
     inx

     lda  songs,x
     sta  instr+8
     inx
     lda  songs,x
     sta  instr+9
     inx

     lda  songs,x
     sta  instr+10
     inx
     lda  songs,x
     sta  instr+11
     inx

     lda  songs,x
     sta  instr+15
     inx
     lda  songs,x
     sta  instr+16
     inx

     lda  songs,x
     sta  instr+17
     inx
     lda  songs,x
     sta  instr+18
     rts


                                  ; initialze sid registers


songs =*
  .byte $0F                       ; velocity
  .byte $0A                       ; Attack/Decay
  .byte $00                       ; Sustain/Release
  .byte <nGavotte1                ; Voice 1: notes address lo
  .byte >nGavotte1                ; Voice 1: notes address hi
  .byte <lGavotte1                ; Voice 1: notes duration address lo
  .byte >lGavotte1                ; Voice 1: notes duration address hi
  .byte <nGavotte2                ; Voice 2: notes address lo
  .byte >nGavotte2                ; Voice 2: notes address hi
  .byte <lGavotte2                ; Voice 2: notes duration address lo
  .byte >lGavotte2                ; Voice 2: notes duration address hi
  .byte <nGavotte3                ; Voice 3: notes address lo
  .byte >nGavotte3                ; Voice 3: notes address hi
  .byte <lGavotte3                ; Voice 3: notes duration address lo
  .byte >lGavotte3                ; Voice 3: notes duration address hi
  .byte $00

  .byte $0E                       ; velocity
  .byte $0C                       ; Attack/Decay
  .byte $00                       ; Sustain/Release
  .byte <nMarines1                ; Voice 1: notes address lo
  .byte >nMarines1                ; Voice 1: notes address hi
  .byte <lMarines1                ; Voice 1: notes duration address lo
  .byte >lMarines1                ; Voice 1: notes duration address hi
  .byte <nMarines2                ; Voice 2: notes address lo
  .byte >nMarines2                ; Voice 2: notes address hi
  .byte <lMarines2                ; Voice 2: notes duration address lo
  .byte >lMarines2                ; Voice 2: notes duration address hi
  .byte <nNull                    ; Voice 3: notes address lo
  .byte >nNull                    ; Voice 3: notes address hi
  .byte <nNull                    ; Voice 3: notes duration address lo
  .byte >nNull                    ; Voice 3: notes duration address hi
  .byte $00

  .byte $11                       ; velocity
  .byte $3B                       ; Attack/Decay
  .byte $00                       ; Sustain/Release
  .byte <nAndante1                ; Voice 1: notes address lo
  .byte >nAndante1                ; Voice 1: notes address hi
  .byte <lAndante1                ; Voice 1: notes duration address lo
  .byte >lAndante1                ; Voice 1: notes duration address hi
  .byte <nAndante2                ; Voice 2: notes address lo
  .byte >nAndante2                ; Voice 2: notes address hi
  .byte <lAndante2                ; Voice 2: notes duration address lo
  .byte >lAndante2                ; Voice 2: notes duration address hi
  .byte <nNull                    ; Voice 3: notes address lo
  .byte >nNull                    ; Voice 3: notes address hi
  .byte <nNull                    ; Voice 3: notes duration address lo
  .byte >nNull                    ; Voice 3: notes duration address hi
  .byte $00

  .byte $0A                       ; velocity
  .byte $19                       ; Attack/Decay
  .byte $00                       ; Sustain/Release
  .byte <nAllegro1                ; Voice 1: notes address lo
  .byte >nAllegro1                ; Voice 1: notes address hi
  .byte <lAllegro1                ; Voice 1: notes duration address lo
  .byte >lAllegro1                ; Voice 1: notes duration address hi
  .byte <nAllegro2                ; Voice 2: notes address lo
  .byte >nAllegro2                ; Voice 2: notes address hi
  .byte <lAllegro2                ; Voice 2: notes duration address lo
  .byte >lAllegro2                ; Voice 2: notes duration address hi
  .byte <nNull                    ; Voice 3: notes address lo
  .byte >nNull                    ; Voice 3: notes address hi
  .byte <nNull                    ; Voice 3: notes duration address lo
  .byte >nNull                    ; Voice 3: notes duration address hi
  .byte $00


contr =*                          ; control register
  .byte 1;0
  .byte 8                         ; velocity
  .byte 0,0
freq =*
  .byte $02, $d0
  .byte $02, $fb
  .byte $03, $28
  .byte $03, $58
  .byte $00, $00
  .byte $03, $c1
  .byte $03, $fa
  .byte $04, $37
  .byte $04, $77
  .byte $04, $bb
  .byte $05, $03
  .byte $05, $4f
  .byte $05, $a0
  .byte $05, $f6
  .byte $06, $50
  .byte $06, $b0
  .byte $07, $16
  .byte $07, $82
  .byte $07, $f4
  .byte $08, $6e
  .byte $08, $ee
  .byte $09, $76
  .byte $0a, $06
  .byte $0a, $9e
  .byte $0b, $40
  .byte $0b, $eb
  .byte $0c, $a1
  .byte $0d, $61
  .byte $0e, $2d
  .byte $0f, $04
  .byte $0f, $e9
  .byte $10, $db
  .byte $11, $dc
  .byte $12, $ec
  .byte $14, $0c
  .byte $15, $3d
  .byte $16, $80
  .byte $17, $d7
  .byte $19, $41
  .byte $1a, $c2
  .byte $1c, $59
  .byte $1e, $09
  .byte $1f, $d2
  .byte $21, $b6
  .byte $23, $b7
  .byte $25, $d7
  .byte $28, $17
  .byte $2a, $79
  .byte $2d, $00
  .byte $2f, $ad
  .byte $32, $83
  .byte $35, $84
  .byte $38, $b2
  .byte $3c, $11
  .byte $3f, $a4
  .byte $43, $6c
  .byte $47, $6f
  .byte $4b, $ae
  .byte $50, $2e
  .byte $54, $f3

instr=*
  .byte 33                        ; Voice 1: Control registers
  .byte <nGavotte1                ; Voice 1: notes address lo
  .byte >nGavotte1                ; Voice 1: notes address hi
  .byte <lGavotte1                ; Voice 1: notes duration address lo
  .byte >lGavotte1                ; Voice 1: notes duration address hi
  .byte 255
  .byte 255

  .byte 33                        ; Voice 2: Control registers
  .byte <nGavotte2                ; Voice 2: notes address lo
  .byte >nGavotte2                ; Voice 2: notes address hi
  .byte <lGavotte2                ; Voice 2: notes duration address lo
  .byte >lGavotte2                ; Voice 2: notes duration address hi
  .byte 255
  .byte 255

  .byte 33                        ; Voice 3: Control registers
  .byte <nGavotte3                ; Voice 3: notes address lo
  .byte >nGavotte3                ; Voice 3: notes address hi
  .byte <lGavotte3                ; Voice 3: notes duration address lo
  .byte >lGavotte3                ; Voice 3: notes duration address hi
  .byte 255
  .byte 255

nGavotte1 =*                      ; note value for voice 1
  .byte $05, $2c, $2b, $2c, $2e, $2c, $27, $30
  .byte $2e, $30, $31, $30, $2e, $2c, $33, $31
  .byte $30, $2e, $2c, $2e, $2c, $2b, $2c, $2e
  .byte $2c, $27, $30, $2e, $30, $31, $30, $2e
  .byte $2c, $33, $31, $30, $2e, $2c, $2e

  .byte $27, $25, $24, $29, $25, $24, $22, $27
  .byte $24, $22, $20, $25, $24, $22, $20, $1f
  .byte $20, $1f, $20, $22, $20, $1b, $25, $24
  .byte $25, $27, $24, $22, $24, $25, $22, $27
  .byte $20, $1f, $20, $22, $24, $22, $20

  .byte $27, $25, $24, $29, $25, $24, $22, $27
  .byte $24, $22, $20, $25, $24, $22, $20, $1f
  .byte $20, $1f, $20, $22, $20, $1b, $25, $24
  .byte $25, $27, $24, $22, $24, $25, $22, $27
  .byte $20, $1f, $20, $22, $24, $22, $20, $00

lGavotte1 =*                      ; note duration for voice 1
  .byte $08, $01, $01, $01, $01, $02, $02, $01
  .byte $01, $01, $01, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $04, $01, $01, $01, $01
  .byte $02, $02, $01, $01, $01, $01, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $04
  .byte $02, $01, $01, $04, $02, $01, $01, $04
  .byte $02, $01, $01, $02, $02, $02, $02, $04
  .byte $01, $01, $01, $01, $02, $02, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $02, $02
  .byte $01, $01, $01, $01, $02, $02, $04

  .byte $02, $01, $01, $04, $02, $01, $01, $04
  .byte $02, $01, $01, $02, $02, $02, $02, $04
  .byte $01, $01, $01, $01, $02, $02, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $02, $02
  .byte $01, $01, $01, $01, $02, $02, $04, $00

nGavotte2 =*                      ; note value for voice 2
  .byte $05, $05, $27, $27, $29, $2c, $29, $2b
  .byte $2c, $29, $26, $27, $05, $27, $27, $29
  .byte $2c, $29, $2b, $2c, $29, $26, $27

  .byte $20, $24, $22, $20, $22, $05, $22, $20
  .byte $1f, $20, $05, $20, $20, $1d, $1b, $05
  .byte $05, $05, $20, $20, $1f, $20, $22, $20
  .byte $1f, $18

  .byte $20, $24, $22, $20, $22, $05, $22, $20
  .byte $1f, $20, $05, $20, $20, $1d, $1b, $05
  .byte $05, $05, $20, $20, $1f, $20, $22, $20
  .byte $1f, $18, $00

lGavotte2 =*                      ; note duration for voice 2
  .byte $08, $08, $04, $04, $02, $02, $01, $01
  .byte $02, $02, $02, $04, $08, $04, $04, $02
  .byte $02, $01, $01, $02, $02, $02, $04

  .byte $04, $02, $01, $01, $02, $02, $02, $01
  .byte $01, $02, $02, $02, $02, $02, $02, $08
  .byte $08, $06, $02, $01, $01, $01, $01, $02
  .byte $02, $04

  .byte $04, $02, $01, $01, $02, $02, $02, $01
  .byte $01, $02, $02, $02, $02, $02, $02, $08
  .byte $08, $06, $02, $01, $01, $01, $01, $02
  .byte $02, $04, $00

nGavotte3 =*                      ; note value for voice 3
  .byte $05, $14, $18, $1b, $20, $1f, $20, $22
  .byte $20, $1f, $1d, $24, $22, $20, $19, $1d
  .byte $1b, $19, $18, $16, $14, $18, $1b, $20
  .byte $1f, $20, $22, $20, $1f, $1d, $24, $22
  .byte $20, $19, $1d, $1b

  .byte $18, $14, $11, $16, $13, $11, $0f, $14
  .byte $18, $11, $14, $16, $18, $1b, $19, $18
  .byte $16, $18, $14, $16, $1f, $20, $14, $1b
  .byte $19, $18, $1d, $19, $1b, $0f, $14

  .byte $18, $14, $11, $16, $13, $11, $0f, $14
  .byte $18, $11, $14, $16, $18, $1b, $19, $18
  .byte $16, $18, $14, $16, $1f, $20, $14, $1b
  .byte $19, $18, $1d, $19, $1b, $0f, $14, $00

lGavotte3 =*                      ; note duration for voice 3
  .byte $08, $04, $02, $02, $01, $01, $01, $01
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $01, $01, $01, $01, $04, $02, $02, $01
  .byte $01, $01, $01, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $04

  .byte $04, $02, $02, $04, $02, $01, $01, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $01
  .byte $01, $02, $02, $02, $02, $02, $04

  .byte $04, $02, $02, $04, $02, $01, $01, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $01
  .byte $01, $02, $02, $02, $02, $02, $04, $00

nMarines1 =*
  .byte $05, $1e, $22, $25, $25, $25, $25, $25
  .byte $2a, $25, $22, $23, $25, $25, $23, $20
  .byte $1e, $1e, $22, $25, $25, $25, $25, $25
  .byte $2a, $25, $22, $23, $25, $25, $23, $20
  .byte $1e, $2a, $29, $27, $23, $27, $23, $25
  .byte $27, $25, $2a, $29, $27, $23, $27, $2a
  .byte $25, $1e, $22, $25, $25, $25, $25, $25
  .byte $2a, $25, $22, $23, $25, $25, $23, $20
  .byte $1e, $00

lMarines1 =*
  .byte $08, $01, $01, $02, $02, $02, $02, $03
  .byte $01, $02, $01, $01, $02, $02, $02, $02
  .byte $06, $01, $01, $02, $02, $02, $02, $03
  .byte $01, $02, $01, $01, $02, $02, $02, $02
  .byte $06, $01, $01, $02, $02, $02, $02, $03
  .byte $01, $02, $01, $01, $02, $02, $02, $02
  .byte $06, $01, $01, $02, $02, $02, $02, $03
  .byte $01, $02, $01, $01, $02, $02, $02, $02
  .byte $07, $00

nMarines2 =*
  .byte $05, $12, $12, $16, $16, $05, $16, $05
  .byte $16, $05, $16, $05, $16, $16, $16, $16
  .byte $05, $11, $11, $05, $11, $05, $11, $05
  .byte $11, $12, $16, $16, $16, $16, $05, $16
  .byte $16, $05, $16, $05, $16, $05, $16, $05
  .byte $16, $16, $16, $16, $05, $11, $11, $05
  .byte $11, $05, $11, $05, $11, $12, $16, $16
  .byte $16, $16, $05, $05, $17, $05, $12, $05
  .byte $17, $05, $12, $05, $16, $16, $16, $16
  .byte $05, $05, $17, $05, $12, $05, $17, $05
  .byte $12, $05, $16, $16, $16, $16, $05, $16
  .byte $16, $05, $16, $05, $16, $05, $16, $05
  .byte $16, $16, $16, $16, $05, $11, $11, $05
  .byte $11, $05, $11, $05, $11, $12, $16, $16
  .byte $16, $16, $00

lMarines2 =*
  .byte $08, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $02
  .byte $02, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $02, $02, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $02, $02, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $02, $02, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $02
  .byte $02, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $02, $02, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $02, $02, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $03, $00

nAndante1 =*
  .byte $05, $2a, $2c, $2a, $2c, $2e, $2a, $25
  .byte $25, $25, $2c, $2e, $2c, $2e, $2f, $2c
  .byte $25, $25, $25, $31, $29, $2a, $2c, $29
  .byte $2a, $33, $30, $2c, $2a, $29, $31, $2e
  .byte $33, $31, $30, $30, $31, $05

  .byte $05, $2a, $2c, $2a, $2c, $2e, $2a, $25
  .byte $25, $25, $2c, $2e, $2c, $2e, $2f, $2c
  .byte $25, $25, $25, $31, $29, $2a, $2c, $29
  .byte $2a, $33, $30, $2c, $2a, $29, $31, $2e
  .byte $33, $31, $30, $30, $31, $05

  .byte $31, $2e, $2f, $31, $33, $2f, $2c, $2e
  .byte $2f, $31, $2f, $2e, $2c, $2e, $2a, $2f
  .byte $2e, $2c, $05, $2f, $2e, $2c, $2a, $29
  .byte $31, $2f, $2e, $2c, $2a, $27, $33, $2f
  .byte $2c, $2a, $29, $29, $2a, $05

  .byte $31, $2e, $2f, $31, $33, $2f, $2c, $2e
  .byte $2f, $31, $2f, $2e, $2c, $2e, $2a, $2f
  .byte $2e, $2c, $05, $2f, $2e, $2c, $2a, $29
  .byte $31, $2f, $2e, $2c, $2a, $27, $33, $2f
  .byte $2c, $2a, $29, $29, $2a, $05, $00

lAndante1 =*
  .byte $04, $04, $01, $01, $01, $01, $02, $02
  .byte $02, $02, $04, $01, $01, $01, $01, $02
  .byte $02, $02, $02, $04, $01, $01, $01, $01
  .byte $04, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $02, $02, $04, $02, $02

  .byte $04, $04, $01, $01, $01, $01, $02, $02
  .byte $02, $02, $04, $01, $01, $01, $01, $02
  .byte $02, $02, $02, $04, $01, $01, $01, $01
  .byte $04, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $02, $02, $04, $02, $02

  .byte $04, $01, $01, $01, $01, $04, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $02, $02
  .byte $04, $02, $02, $04, $01, $01, $01, $01
  .byte $04, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $02, $02, $04, $02, $02

  .byte $04, $01, $01, $01, $01, $04, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $02, $02
  .byte $04, $02, $02, $04, $01, $01, $01, $01
```

```
   .byte $04, $01, $01, $01, $01, $01, $01, $01
   .byte $01, $02, $02, $04, $02, $02, $00

nAndante2 =*
   .byte $05, $22, $25, $22, $1e, $22, $25, $22
   .byte $25, $22, $1e, $25, $2a, $29, $2a, $2c
   .byte $29, $25, $29, $2c, $29, $2c, $29, $25
   .byte $22, $05, $25, $27, $24, $25, $2a, $2c
   .byte $2c, $2c, $25, $05

   .byte $05, $22, $25, $22, $1e, $22, $25, $22
   .byte $25, $22, $1e, $25, $2a, $29, $2a, $2c
   .byte $29, $25, $29, $2c, $29, $2c, $29, $25
   .byte $22, $05, $25, $27, $24, $25, $2a, $2c
   .byte $2c, $2c, $25, $05

   .byte $05, $2e, $2b, $27, $05, $2c, $29, $25
   .byte $2a, $29, $27, $2c, $2a, $1e, $22, $1e
   .byte $25, $05, $2c, $29, $25, $05, $22, $2a
   .byte $27, $05, $24, $25, $05, $05, $1e, $05

   .byte $05, $2e, $2b, $27, $05, $2c, $29, $25
   .byte $2a, $29, $27, $2c, $2a, $1e, $22, $1e
   .byte $25, $05, $2c, $29, $25, $05, $22, $2a
   .byte $27, $05, $24, $25, $05, $05, $1e, $05
   .byte $00

lAndante2 =*
   .byte $04, $04, $04, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $04, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $02, $02, $04, $04, $04, $02, $02, $02
   .byte $02, $04, $02, $02

   .byte $04, $04, $04, $01, $01, $01, $01, $01
   .byte $01, $01, $01, $04, $01, $01, $01, $01
   .byte $01, $01, $01, $01, $01, $01, $01, $01
   .byte $02, $02, $04, $04, $04, $02, $02, $02
   .byte $02, $04, $02, $02

   .byte $02, $02, $02, $02, $02, $02, $02, $02
   .byte $02, $02, $02, $02, $01, $01, $01, $01
   .byte $02, $04, $02, $02, $02, $02, $02, $02
   .byte $02, $03, $01, $02, $02, $04, $02, $02

   .byte $02, $02, $02, $02, $02, $02, $02, $02
   .byte $02, $02, $02, $02, $01, $01, $01, $01
   .byte $02, $04, $02, $02, $02, $02, $02, $02
   .byte $02, $03, $01, $02, $02, $04, $02, $02
   .byte $00

nAllegro1 =*
   .byte $05, $2e, $31, $2f, $2e, $2c, $2a, $05
   .byte $2a, $2c, $2c, $2c, $2c, $2e, $2a, $05
   .byte $2a, $2c, $2c, $2c, $2c, $2e, $2a, $2c
   .byte $2e, $2f, $31, $33, $35, $36, $33, $2e
   .byte $30, $31, $05, $31, $33, $35, $36, $33
   .byte $30, $31, $33, $30, $31, $33, $35, $36
   .byte $31, $35, $33, $31, $33, $35, $36, $33
   .byte $30, $31, $33, $30, $31, $05

   .byte $05, $2e, $31, $2f, $2e, $2c, $2a, $05
   .byte $2a, $2c, $2c, $2c, $2c, $2e, $2a, $05
   .byte $2a, $2c, $2c, $2c, $2c, $2e, $2a, $2c
   .byte $2e, $2f, $31, $33, $35, $36, $33, $2e
   .byte $30, $31, $05, $31, $33, $35, $36, $33
   .byte $30, $31, $33, $30, $31, $33, $35, $36
   .byte $31, $35, $33, $31, $33, $35, $36, $33
   .byte $30, $31, $33, $30, $31, $05

   .byte $25, $31, $2e, $33, $31, $2f, $2c, $31
   .byte $2f, $2e, $2a, $2f, $2e, $2c, $2a, $31
   .byte $05, $2e, $31, $2f, $2e, $2c, $2a, $2c
   .byte $2e, $2f, $31, $33, $35, $36, $35, $2a
   .byte $2c, $2e, $2f, $31, $2e, $2c, $2a, $35
   .byte $36, $05, $31, $05, $2f, $05, $2c, $05
   .byte $2a, $05, $25, $2a, $2a, $2a, $25, $2e
   .byte $2e, $2e, $2a, $31, $2c, $2e, $2a, $05

   .byte $25, $31, $2e, $33, $31, $2f, $2c, $31
   .byte $2f, $2e, $2a, $2f, $2e, $2c, $2a, $31
   .byte $05, $2e, $31, $2f, $2e, $2c, $2a, $2c
   .byte $2e, $2f, $31, $33, $35, $36, $35, $2a
   .byte $2c, $2e, $2f, $31, $2e, $2c, $2a, $35
   .byte $36, $05, $31, $05, $2f, $05, $2c, $05
   .byte $2a, $05, $25, $2a, $2a, $2a, $25, $2e
   .byte $2e, $2e, $2a, $31, $2c, $2e, $2a, $05
   .byte $00

lAllegro1 =*
```

```
  .byte $08, $02, $02, $02, $02, $02, $04, $02
  .byte $02, $02, $02, $02, $01, $01, $04, $02
  .byte $02, $02, $02, $02, $01, $01, $02, $01
  .byte $01, $01, $01, $01, $01, $02, $02, $02
  .byte $02, $04, $02, $02, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $04, $02

  .byte $08, $02, $02, $02, $02, $02, $04, $02
  .byte $02, $02, $02, $02, $01, $01, $04, $02
  .byte $02, $02, $02, $02, $01, $01, $02, $01
  .byte $01, $01, $01, $01, $01, $02, $02, $02
  .byte $02, $04, $02, $02, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $01
  .byte $01, $01, $01, $01, $04, $02

  .byte $02, $06, $02, $06, $02, $06, $02, $06
  .byte $02, $06, $04, $04, $04, $04, $02, $04
  .byte $02, $02, $02, $02, $02, $02, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $08, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $08
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $04, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $04, $03, $01, $04, $02

  .byte $02, $06, $02, $06, $02, $06, $02, $06
  .byte $02, $06, $04, $04, $04, $04, $02, $04
  .byte $02, $02, $02, $02, $02, $02, $01, $01
  .byte $01, $01, $01, $01, $01, $01, $08, $01
  .byte $01, $01, $01, $01, $01, $01, $01, $08
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $04, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $04, $03, $01, $04, $02
  .byte $00

nAllegro2 =*
  .byte $05, $2a, $2e, $2c, $2a, $25, $22, $05
  .byte $22, $25, $25, $25, $25, $22, $05, $22
  .byte $25, $25, $25, $25, $22, $05, $2a, $29
  .byte $2c, $29, $2c, $2a, $2c, $2a, $2c, $29
  .byte $2c, $29, $2c, $2a, $2c, $2a, $2c, $29
  .byte $05

  .byte $05, $2a, $2e, $2c, $2a, $25, $22, $05
  .byte $22, $25, $25, $25, $25, $22, $05, $22
  .byte $25, $25, $25, $25, $22, $05, $2a, $29
  .byte $2c, $29, $2c, $2a, $2c, $2a, $2c, $29
  .byte $2c, $29, $2c, $2a, $2c, $2a, $2c, $29
  .byte $05

  .byte $05, $1e, $2a, $27, $2c, $2a, $29, $25
  .byte $2a, $29, $28, $27, $25, $23, $24, $25
  .byte $05, $2a, $2e, $2c, $2a, $25, $22, $05
  .byte $2e, $2f, $2c, $31, $2f, $2e, $05, $2e
  .byte $2f, $2c, $31, $2f, $2e, $05, $2e, $05
  .byte $2c, $05, $25, $05, $22, $05, $19, $1e
  .byte $1e, $1e, $19, $22, $22, $22, $1e, $25
  .byte $19, $1e, $05

  .byte $05, $1e, $2a, $27, $2c, $2a, $29, $25
  .byte $2a, $29, $28, $27, $25, $23, $24, $25
  .byte $05, $2a, $2e, $2c, $2a, $25, $22, $05
  .byte $2e, $2f, $2c, $31, $2f, $2e, $05, $2e
  .byte $2f, $2c, $31, $2f, $2e, $05, $2e, $05
  .byte $2c, $05, $25, $05, $22, $05, $19, $1e
  .byte $1e, $1e, $19, $22, $22, $22, $1e, $25
  .byte $19, $1e, $05, $00

lAllegro2 =*
  .byte $08, $02, $02, $02, $02, $02, $04, $02
  .byte $02, $02, $02, $02, $02, $04, $02, $02
  .byte $02, $02, $02, $02, $04, $04, $08, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $04
  .byte $02

  .byte $08, $02, $02, $02, $02, $02, $04, $02
  .byte $02, $02, $02, $02, $02, $04, $02, $02
  .byte $02, $02, $02, $02, $04, $04, $08, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $04
  .byte $02

  .byte $04, $02, $06, $02, $06, $02, $06, $02
  .byte $06, $02, $04, $04, $04, $04, $04, $04
  .byte $02, $02, $02, $02, $02, $02, $04, $02
  .byte $02, $02, $02, $02, $02, $04, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $04, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $04, $04, $04

  .byte $04, $02, $06, $02, $06, $02, $06, $02
  .byte $06, $02, $04, $04, $04, $04, $04, $04
  .byte $02, $02, $02, $02, $02, $02, $04, $02
  .byte $02, $02, $02, $02, $02, $04, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $02, $02, $02, $02, $04, $02, $02, $02
  .byte $02, $02, $02, $02, $02, $02, $02, $02
  .byte $04, $04, $04, $00

nNull =*
  .byte $05, $00
```

Conclusion

If you are wondering why it could be a good point to rip a BASIC music, the answer is simple: there are many old BASIC games that are attending to be ripped: check the Sid Hunt at http://lala.c64.org/sid-hunt

At the end of this article I would like to thanks Andrew Colin for having created the Sheba music routine: I use it many years ago for experimenting Sid music (creating IMHO an horrible tune, but with good remembrance for me).

Probably the fact of not having the machine code spread out in the book and so having hidden the sound generation of the player had give me the impulse to study the computer science: a choice that is more that 20 years I' m follow!