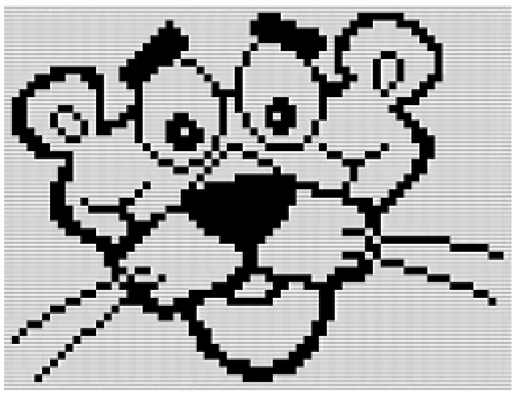


Pinkpanther or How does the music come out of the ZX81?

On our last ZX-Team micro fair you could hear it from all corners: the Pink Panther theme from the film. As far as I know Kelly Abrantes Murta (http://zx81.eu5.org/toddysofte.html) designed this program for ZX81 with ZON-X box attached, the ZX81 soundcard. He first used the PC program 'midi2ay.exe', to get a format suitable for AY-3-8912 or AY-3-8910. He passes the sound file to ZON-X using machine code. This is embedded into a BASIC-program and the ingenious idea is that the execution of the MC is coupled with the display-routine. This enables running of slow mode BASIC programs with playing sound in background.

Such a program I was looking after for a long time. On the one hand I wanted to convert midi files to ZON-X files to be played on my soundcard, on the other hand I wanted to run a BASIC-program.

Luckily the assembler sources were available, so I could type in the Program in ASDIS. Thats what it looks like:

```
4082     HALT           76        ;REM-Zeilenausgabe unterdrücken
4083     HALT           76        ;REM-Zeilenausgabe unterdrücken
4084     LD IX,SOUND    DD218940  ;Display-Routine verbiegen,
4088     RET            C9        ;sodass Tonausgabe angesprungen wird.
4089SOUNDLD A,R         ED5F
408B     LD BC,$1901    010119
408E     LD A,$F5       3EF5
4090     CALL $02B5     CDB502
4093     CALL $0292     CD9202
4096     CALL $0220     CD2002
4099     LD HL,(STACK)  2ACB40    ;HL := Adr. an der das Soundfile beginnt
409CLOOP LD A,(HL)      7E        ;Prüfe, ob
409D     OR A           B7        ;im File ein Trennzeichen (=00) steht.
409E     JR NZ,NXT1     2013      ;nein, dann ist es eine Tondauer
40A0     INC HL         23        ;ja, dann nächsten Wert prüfen,
40A1     LD A,(HL)      7E        ;ob Fileende erreicht
40A2     CP $FF         FEFF
40A4     JR Z,EXIT      2822      ;ja, zurück zu BASIC
40A6     CP $FE         FEFE      ;nein, prüfe, ob $FE vorliegt
40A8     JR Z,NXT2      2815      ;ja, Sprung zu NXT2
40AA     OUT REG,A      D337      ;nein, Sound ausgeben: Register ausgeben
40AC     INC HL         23        ;Nächste Adresse im Soundfile
40AD     LD A,(HL)      7E
40AE     OUT PRT,A      D317      ;Wert ausgeben
40B0     INC HL         23
40B1     JR LOOP        18E9      ;Weiter mit Tonausgabe bis $FF erreicht.
40B3NXT1 LD (STACK),HL  22CB40    ;Tondauer abarbeiten: Soundfileadresse
40B6     DEC A          3D        ;speichern und Wert für Tondauer - 1.
40B7     LD (HL),A      77        ;Neuen Wert an die gleiche Stelle
40B8     LD IX,SOUND    DD218940  ;speichern und zurück zu BASIC,
40BC     JP EXIT        C3C840    ;bis nächster Bildaufbau kommt.
40BFNXT2 INC HL         23        ;Aktuelle Soundfile-Adresse + 2
40C0     INC HL         23
```

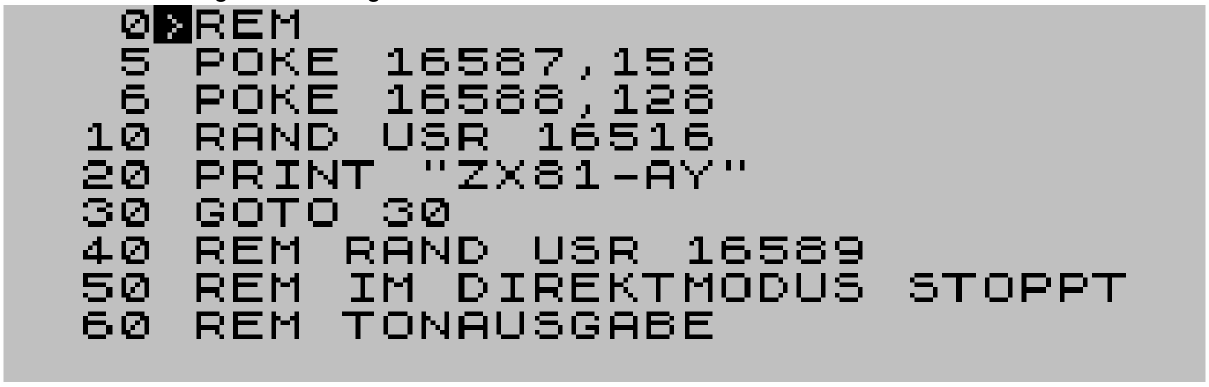
```
40C1      LD (STACK),HL  22CB40    ;Errechnete Adresse speichern und
40C4      LD IX,SOUND    DD218940  ;zurück zu BASIC
40C8EXIT JP $02A4       C3A402
40CBSTACK002=$02                  ;Speicheradresse für Soundfile-Adresse
40CD;-------------------------
40CDQUIETLD HL,CODES    21DD40    ;Soundchip auf 'Ruhe' stellen
40D0QLOOPLD A,(HL)      7E
40D1      CP $FF         FEFF
40D3      RET Z          C8
40D4      OUT REG,A      D337
40D6      INC HL         23
40D7      LD A,(HL)      7E
40D8      OUT PRT,A      D317
40DA      INC HL         23
40DB      JR QLOOP       18F3
40DD;;
40DDCODES0738080009000A00FF
```

'REG' is the address for register select of the sound chip
'PRT' is the port to put in the value for the register.

The addresses for the sound chips are:

My soundcard	: REG=$37	/	PRT = $17
original ZON-X	: REG=$CF	/	PRT = $1F
EightyOne and modified ZON-X	: REG = $DF	/	PRT = $0F

I used the following BASIC-Program to call the machine code:



```
   0 REM
   5 POKE 16587,158
   6 POKE 16588,128
  10 RAND USR 16516
  20 PRINT "ZX81-AY"
  30 GOTO 30
  40 REM RAND USR 16589
  50 REM IM DIREKTMODUS STOPPT
  60 REM TONAUSGABE
```

Lines 5 and 6 determine the start of the sound file. In this Case it is 32926. Line 10 calls the machine code. Line 20 writes to the screen - here you could print the name of the sound file for example, or you could run your basic program.
Line 40 won't have any affect in most cases as the sound module will receive new commands immediately. It only helps calling line 40 after FAST/SLOW. Doing FAST and SLOW stops the music from being played. As the register may still have values there may be a continuous tone. Line 40 sets the values to zero - 'silence'.

This program is just the beginning. What I don't like is that the sound file is being modified while playing. the length of each tone is used as a counter being set to zero when read. If you tried to restart the music all length of tones were zero and the music would be played within a second. So you have to reload the ZON-X file each time you want to play it.

How do we get the midi file into our ZX81?
First we need a midi file. You find them in the internet for free. Please take care about copyrights.
Then you need the program 'midi2ay.exe' and a PC. I will show the usage with an example.
'pinkpanther.mid' is our midi file to be converted and 'pink.bin' the sound file to be created.
In a DOS-box you enter
midi2ay -tap pinkpanther.mid pink.bin
This creates the sound file. We now have to check that the created sound file fits into our zeddys memory and save it to an SD-card.
From this SD-card you load pink.bin into the zeddy using LOAD"PINK.BIN;32768"

This loads pink.bin to adress $8000. Taking a look to the sound file explains why the sound file beginns at 32926. Again we look at PinkPanther:

The first 158 bytes of pink.bin contain a payer for the ZX Spectrum. That's why we load to adress 32768 and the soundfile starts at 32926.
then there are the register values. Its normally like this:
After two values (1 register, 1 content) we have $00 as delimiter. If there is no delimiter, then then the next value is the length of the sound, not followed by a delimiter.
Thats what the first 90 bytes of the sound file look like:

```
$00,$07,$38,$00,$08,$00,$00,$09,$00
$00,$10,$00,$48,$00,$0b,$00,$01,$01
$00,$08,$0c,$00,$02,$90,$00,$03,$01
$00,$09,$0c,$00,$04,$16,$00,$05,$02
$00,$0a,$0c,$09,$00,$fc,$00,$01,$00
$00,$02,$79,$00,$04,$f8,$00,$05,$01
$09,$08,$00,$00,$09,$00,$00,$0a,$00
$09,$00,$ee,$00,$08,$0c,$00,$02,$64
$00,$09,$0c,$00,$04,$db,$00,$0a,$0c
$09,$00,$e0,$00,$02,$50,$00,$04,$c1
```

For a better understanding I display the registers and their values:

Register	dezimal	128	64	32	16	8	4	2	1
	Bits	B7	B6	B5	B4	B3	B2	B1	B0
00	Tonhöhe Kanal A	8-Bit Feinabgleich							
01						4-Bit Grobabgleich			
02	Tonhöhe Kanal B	8-Bit Feinabgleich							
03						4-Bit Grobabgleich			
04	Tonhöhe Kanal C	8-Bit Feinabgleich							
05						4-Bit Grobabgleich			
06	Rauschfrequenz				5-Bit Abgleich				
07	Mixer I/O-Auswahl	/IN	OUT	/Rauschen			/Ton		
		IO-B	IO-A	C	B	A	C	B	A
08	Lautstärke A				M	4-Bit Abgleich			
09	Lautstärke B				M	4-Bit Abgleich			
0A	Lautstärke C				M	4-Bit Abgleich			
0B	Hüllfrequenz	8-Bit Feinabgleich							
0C		8-Bit Grobableich							
0D	Hüllkurvenform					Cont	Att	Alt	Hold
0E	I/O-Port A	8-Bit I/O Port A							
0F	I/O-Port B	8-Bit I/O Port B							

Now we look at the bytes and their meaning:

$00	delimiter or start
$07,$38	register 07 (mixer): A=input, B=input, no noise, sound output on A,B,C
$00	delimiter
$08,$00	register 08: volume A to 0
$00	delimiter
$09,$00	register 09: volume B to 0
$00	delimiter
$10,$00	register 0A (maybe here is an error in midi2ay?): volume C to 0
$48	length
$00,$0b	register 00: frequency channel A fine tuning

$00	delimiter
$01,$01	register 01: frequency channel A main tuning
$00	delimiter
$08,$0c	register 08: Volume Channel A $0c
$00	delimiter
$02,$90	Register 02: Frequency Channel B Fine tuning
$00	Delimiter
$03,$01	Register 03: Frequency Channel B Main tuning
$00	Delimiter
$09,$0c	Register 09: Volume Channel B: $0c
$00	Delimiter
$04,$16	Register 04: Frequency Channel C Fine tuning
$00	Delimiter
$05,$02	Register 05: Frequency Channel C Main tuning
$00	Delimiter
$0a,$0c	Register 0A: Volume Channel C: $0c
$09	Length $09
$00,$fc	Register 00: Frequency Channel A Fine tuning
$00	Delimiter
$01,$00	Register 01: Frequency Channel A Main tuning
$00	Delimiter
$02,$79	Register 02: Frequency Channel B Fine tuning
$00	Delimiter
$04,$f8	Register 04: Frequency Channel C Fine tuning
$00	Delimiter
$05,$01	Register 05: Frequency Channel C Main tuning
$09	Length $09 (Volume is not changed)
$08,$00	Register 08: Volume Channel A $00 (Sound off)
$00	Delimiter
$09,$00	Register 09: Volume Channel B $00 (Sound off)
$00	Delimiter
$0a,$00	Register 0a: Volume Channel B $00 (Sound off)
$09	Length $09
$00,$ee	Register 00: Frequency Channel A Fine tuning
$00	Delimiter
$08,$0c	Register 08: Volume Channel A $0c
$00	Delimiter
$02,$64	Register 02: Frequency Channel B Fine tuning
$00	Delimiter
$09,$0c	Register 09: Volume Channel B $0c
$00	Delimiter
$04,$db	Register 04: Frequency Channel C Fine tuning
$00	Delimiter
$0a,$0c	Register 0a: Volume Channel B $0c
$09	Length $09
$00,$e0	Register 00: Frequency Channel A Fine tuning
$00	Delimiter
$02,$50	Register 02: Frequency Channel B Fine tuning
$00	Delimiter
$04,$c1	Register 04: Frequency Channel C Fine tuning

and so on ...

Next I will change the program in a way that the sound file can be included in a giant REM-line, so complete music titles can be saved in a single program. We could also try to make the speed variable. There was a speed issue found by Siggi which is resolved already.

Have fun with the program whishes

Joachim