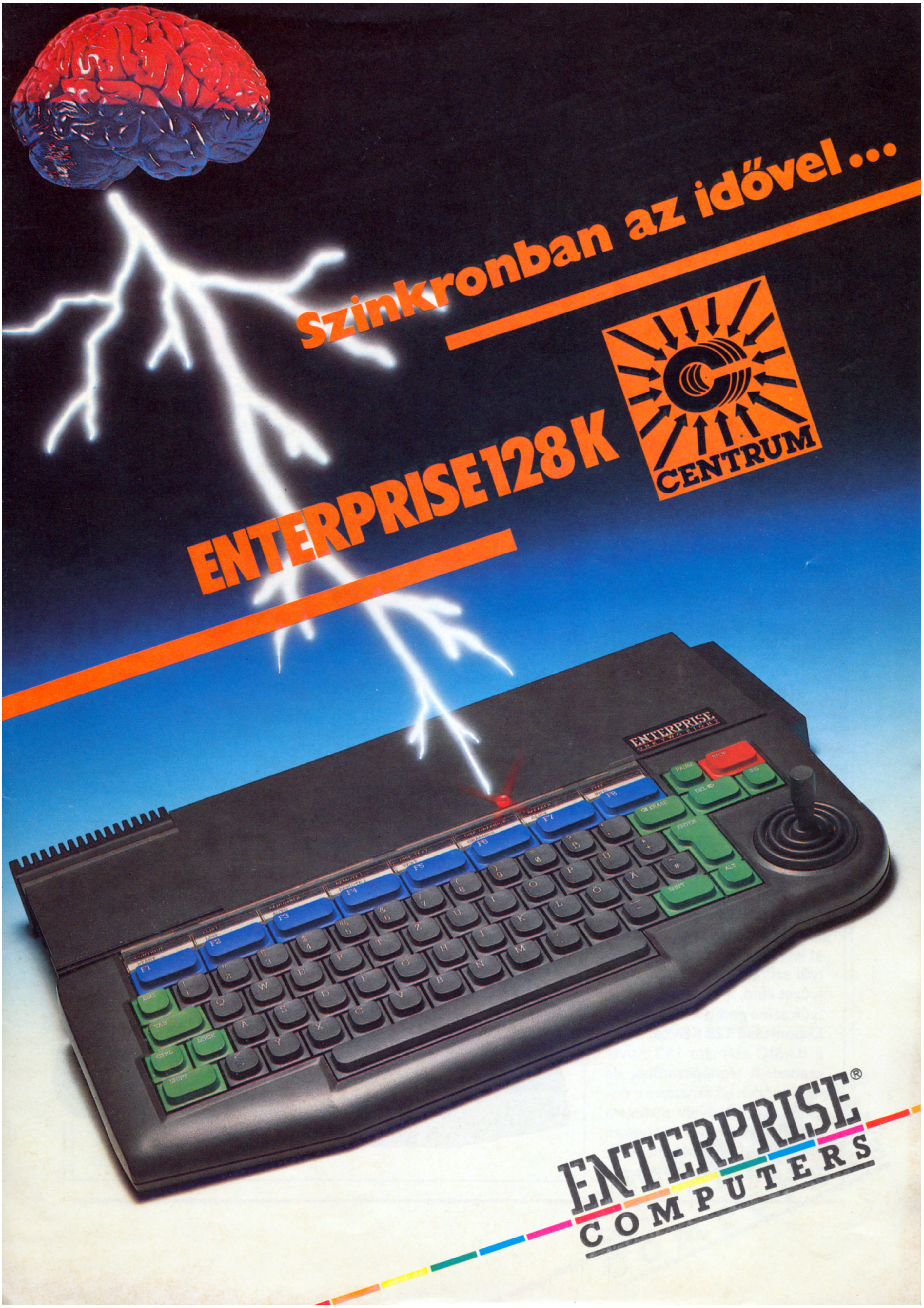
ENTERPRESS

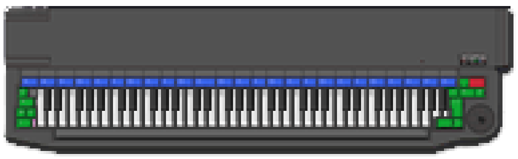
Magazine for the Enterprise users

2017/4, July–August

MIDI
C#2 D#2 F#2 G#2 A#2 C#3 D#3 F#3 G#3 A#3 C#4 D#4 F#4 G#4 A#4 C#5 D#5 F#5 G#5 A#5 C#6 D#6 F#6 G#6 A#6
C2 D2 E2 F2 G2 A2 B2 C3 D3 E3 F3 G3 A3 B3 C4 D4 E4 F4 G4 A4 B4 C5 D5 E5 F5 G5 A5 B5 C6 D6 E6 F6 G6 A6 B6 C7

Description of dave chip – Hisoft Pascal – Treasure Cave


Szinkronban az idővel...
ENTERPRISE 128 K
CENTRUM
ENTERPRISE® COMPUTERS



Enterprise MIDI by IstvanV

(for the time being in ep128emu)

Written by: Tamás Bodnár (szipucsu)

Introduction

A *.mid file doesn't contain any wav sample sound but only the „notes" themselves. The notes can be associated to musical instrument sounds. These instrument sounds are built into the operating system or synthesizer. Midi can sound even 16 layers (channels) at the same time and within one channel even more voices, chords. A special midi channel is the 10th where drums can sound. Our aim is to sound midi files through the EP Dave sound chip. As Dave can handle 3+1 channels altogether most midi files from the net need conversion to get suitable sounding.
Midi hardware also exists for the Enterprise but there are not many pieces. Now a PC midi player/editor application can be connected to the EP128Emu to get the EP sounding midi. To do this you need:
Installing a midi port e.g. loopMIDI. In Linux the Midi Through port can be used or editor programs (e.g. Qtractor) also can create a port that can be directly connected to the emulator. A hardware midi keyboard can also be used, in this case loopMIDI is not needed.
An EP128Emu version that supports midi input. (The latest emulator can be downloaded from sourceforge, this should be updated with this newer beta version. So it only needs unpacking and you should overwrite the existing files of the emulator.)

An EP application called midiplay

If using Windows is the case, you need to configure the midi input (loopMIDI port (I MMSystem)) in the Options -> Sound -> Configure menu item. After that, start a midi player, editor, or virtual keyboard on the PC where output the loopMIDI OUT is set. Start the emulator, load the application called Midiplay on it. After that, the midi player's will sound on the emulated EP, using DAVE's capabilities.
To minimize audio delay and timing errors, you should reduce the default latency setting (70 ms) and select an output device using a low audio API (ALSA or WASAPI), especially when using a MIDI keyboard. Also, it is advisable to avoid OpenGL video emulator modes requiring high CPUs or waiting (eg „resample to monitor refresh rate").
The EP application, Midiplay, includes several files, for example, envelope.txt, see below.

How to use Midiplay in the emulator?

With this application you can play the midi file with the emulated EP as the instrument. If you are using a midi editor (eg Cakewalk, Sonar, etc.), you can also hear the notes through the emulator. It is possible to convert an edited midi file into a loadable and playable EP file (midiconv, see below). Midiplay will automatically check if the default folder contains a converted file called mididata.bin (the maximum size is 1040 bytes of header + 8192 bytes of envelope + 31743 bytes of MIDI data) and, if it is found, the song will be played indefinitely in a loop. If there is no such file, the app will wait for notes received via loopMIDI. While the program is running, the screen will stay completely dark, but newer releases also include mididisp.com with support for simple visualization. It is desirable to run the app on a turbocharged (eg 10 MHz) configuration, but it also works at 4 MHz.

Once started the app, you can use these buttons:

F1: Restarts the program, rereads envelope.txt and (if available) mididata.bin .
F4: Switches off all sounds, useful for „stuck" sounds. The MIDI controller 123 can also be used for similar purposes.
F6: Full MIDI reset, all sounds, controllers, bend and instruments are reset, and the playback position is also returned to the start of the file. The controls and bending can also be reset with the 121 controller.
F8: Exit.

Which MIDI channel (1-16) correspond to each Dave channel? Midi channels are managed by the player as „logical" channels. So each of the sixteen stores their entire status separately:
- from the 128 keys, which one is pressed and, if so, the assigned DAVE channel.
- the program (instrument).
- 4 controllers (7, 10, 76, 77)
- bend (only the top 8 bits are taken into account, the range is fixed at +/- 2 semitones)
This information is independent of the DAVE channels in which the MIDI sounds are played. This is determined by the following algorithm:
- in case of an incoming note on the percussion Midi channel (10), it will be played unconditionally on the DAVE channel 3, immediately interrupting the previous sound.
- otherwise, from the 0-2 Dave channels, one is better choice than the other two when they are through the envelope process (off> release> attack / decay> sustain), then
- if you have not been able to make a decision and you need to interrupt the active sound because all 3 music channels are busy, the older (or previously released) sound will be replaced
- if they are equal then a fixed order will be followed:

MIDI channel 1: 0 > 2 > 1
MIDI channel 2: 2 > 0 > 1
MIDI channel 3: 0 > 2 > 1
MIDI channel 4: 2 > 0 > 1
...etc.

Therefore, Dave channel 1 will be in use only when 3 music sounds are active at the same time. Exactly the other two are certainly 0 and 2. Any of the logical MIDI channels can have any number of voices, but always have a maximum of 3 + 1 percussion.

The volume of the left and right channels can be approximated in the following way:

Left = ENV_L * ((VELOCITY + AFTERTOUCH + 1) / 128) * ((VOLUME + 1) / 128) * cos(PAN * (PI / 2) / 127.5) * sqrt(2)

Right = ENV_R * ((VELOCITY + AFTERTOUCH + 1) / 128) * ((VOLUME + 1) / 128) * sin(PAN * (PI / 2) / 127.5) * sqrt(2)

Where ENV_L = left envelope, ENV_R = right envelope, VELOCITY = strength of the key hit, AFTERTOUCH = extra pressure strength, VOLUME = volume (control 7), PAN = stereo position (control 10). If ENV_L or ENV_R would be multiplied by greater than 1, the multiplier is limited to 1. So, adjusting the stereo position only works correctly if VELOCITY and VOLUME together result in a minimum 3 dB reduction in the volume. The volume multiplication is actually only 1/128 resolution. The AFTERTOUCH parameter is set by the Axh and Dxh MIDI events, the former separately for each key, while the latter for all active notes on the same channel. For a new sound (event 9xh), the initial aftertouch value is always zero.

Special settings for the MIDI files

The Midiplay player therefore pairs the individual midi channels with a certain logic to the Dave channels. However, we can also specify that a given MIDI channel must match a given Dave channel. In addition to the dynamic reservation, you can also set a fixed channel reservation. In fact, in most cases, it is necessary to use Dave's capabilities and convert the midi file downloaded from the net to a meaningful sound on EP.

For this, in the midi file, different controls can be used for different Dave specific settings. (Controllers are generally available in midi editing programs in the Controllers menu.)

Using MIDI Controllers

71 (Harmonic Content) and 76 Controllers: STYLE Parameter / 4

The Style parameter is the same as the one after the Basic command SOUND STYLE, only divided by 4. So, for example, if 16 (low distortion) is desired, then 16/4 = 4 must be entered. If 144 (ring modulation + low distortion)must be applied, then 144/4 = 36 should be specified.

When using 4 or 5 bit distortion, the frequency is automatically corrected to avoid „bad" pitch values where there is no sound or the pitch would be much higher than expected. These occur because of how polynomial counter distortion is implemented on DAVE channels 0 to 2: the counters run constantly at a clock frequency of 250 kHz, and are looping a pseudo-random pattern of 2^N-1 (15, 31 and 127 for 4, 5 and 7 bits, respectively) samples. These counters are then sampled at a frequency of 250000 / (F + 1) Hz, where F is the pitch value written into DAVE registers A0h to A5h. Therefore, if F+1 and the length of the polynomial counter pattern can be divided evenly by the same integer number, then a shorter pattern will be output, possibly no sound if the length is reduced to 1.

In the case of 4-bit distortion, if bit 1 of the controller is set (2 is added to the value), then the frequency is corrected so that the pattern length will be 5 instead of the maximum (15).

The correction works by adding an offset to the DAVE pitch, depending on the remainder of dividing the original value by 15 (4-bit) or 31 (5-bit). For 5-bit distortion, 1 is added (slightly lower frequency) if the remainder is 30. 4-bit distortion uses the following tables:

Pattern length = 15: 0, 0, -1, 0, -1, 1, 0, 0, -1, 1, 0, 1, 0, 0, 1

Pattern length = 5: 2, 1, 0, -1, 1, 0, -1, 1, 0, -1, 1, 0, -1, -2, 3

The latter can overflow if the pitch is very low.

Sound Variation (70) and Control 77 (Channel Control) only affect the bits 2 to 4:

- 0 = dynamic reservation
- 4 = fixed channel 0 (one note only)
- 8 = fixed channel 1
- 12 = fixed channel 2
- 16, 24 = channel 0, up to 3 simultaneous voices system (the notes are alternated on the same channel at a frequency of 50 Hz)
- 20, 28 = Channel 1, up to 3 voices may alternate in the same way

Fixed assignment of DAVE channels is not exclusive, the dynamic MIDI channels can continue in use.

How to use ENVELOPE.TXT

Midi files play the voices of various instruments. Envelope.txt will associate the sound (envelope curve) with these instruments, using this to use Dave's capabilities on EP. It is only necessary to use it for playback on a real-time input (MIDI port) and header-free mididata.bin, newer-format converted files also include the instruments.

The definition of an „instrument" looks like this:

```
INSTRUMENT [D] [P] [: + CHANNEL, + PITCH] [, INSTRUMENT2 ...] {
  [L | R | S] DURATION, LEFT, RIGHT, BEND, DISTORTION;
  ...
}
```

Multiple instruments can use the same envelope, multiple numbers separated by commas. The number of instruments is between 0 and 127 for normal channels, or between -1 and -127 for drums. Optionally, the number can be followed by the ‚D' or ‚P' characters (see below), or select two additional parameters separated by a colon. When using the latter, each note on the instrument will be duplicated with the specified offsets added to the MIDI channel and pitch (the values wrap around if they overflow). For example, with 1:2,12, Instrument 1 also plays a second note an octave higher on the MIDI channel 2 higher (or 14 lower if it would overflow) than the current one.

The ‚D' character after the instrument number forces the full envelope curvature to be released even when the key is released early, for drums and ‚R' envelope curves, this is the default, so there is no use for it, and combining ‚D' with ‚L' or ‚S' results in an error. ‚P' changes the stereo position of the sounds depending on the pitch, no effect on the drum. By default, the ranges between 48 to 79 are set to 1..125, this can be changed with controller 10. With the 9 instrument (Glockenspiel), the effect is enabled by default and is turned off by ‚P'.

The duration (which can be set in video interrupt units, 0 changes the parameters immediately), the character ‚S' means that the beginning of the segment is a „sustain" position from which the player will only continue when the key is released. It does not make any sense for the drums. A section of the envelope can be repeated using the ‚L' character, which will play in a loop infinite the rest of the envelope without adding a release phase. The effect of ‚R' is similar to ‚S', but in contrast, the envelope always runs to the end with ‚R', but with ‚S' it immediately stops if the key is released before reaching the specified point. When using ‚L' and ‚R' together, the interval between them is repeated (from the beginning of the L segment to the end of the segment before R), and at the time of release, the

repeat disappears and the envelope runs to the end.
Unlike EXOS, the volume is an absolute between 0 and 63, it is given the new volume at the beginning of the next segment that is reached by linear interpolation. However, it is also possible to define exponential envelope curves if the volume value is ‚*' before the volume value, then the number / 128 is multiplied by volume at each interruption. For example * 96, 0.75 is the multiplier, * 128 does not change the volume, etc. Bending is between -2048 and +2047, 64 corresponding to a musical semitone. Linear interpolation is also done here and the specified value reaches the beginning of the next segment. Drums can not be bent.
Distortion (STYLE) can not be interpolated in contrast with the other parameters, but its value is immediately changed, and it needs to be watched when it changes in sound.
Spaces, Tab characters and new rows can be used arbitrarily, the program ignores them. In the case of the ‚#' character, the rest of the line is considered a comment and ignored. The player program automatically stores the compiled (binary) envelope data into a file called envelope.bin. The format is:
256 bytes define the duplication of notes for each instrument, the first byte is +CHANNEL and the second is +PITCH, or both are FFh if the feature is not used by the instrument
the following 256 bytes are the same for the drums, in the same format
256 bytes envelope start position for normal instruments: bottom, top byte alternately, or 8080h if the instrument is not defined. The start position is relative to the beginning of the envelope data and is divided by 2, and the top 4 bits have a special function, so the maximum size of the addressable envelope data is 8192 bytes. The flag bits default to 2000h, which is not changed by using ‚S'.
bit 15: disables the instrument
bit 14: pitch-dependent stereo position (P)
bit 13: indicates the absence of a repeating stage (L)
bit 12: releases the full envelope curve (D or R)
256 bytes envelope position for drums
up to 8192 bytes envelope data, every 50 Hz interrupt 4 bytes:
left volume between 0 and 63. Setting bit 6 is the beginning of a repeating phase (L), and 8 is the end that no longer repeats. The two bits are set together to hold the current sample and then indicate a release phase (only R or S). It is a special case when the left volume has bit 7 set and the right volume is FFh, this indicates the end of the envelope and turning off the sound.
- right volume between 0 and 63, or 255 at the end of the envelope
- bend lower 8 bits, its resolution is half a - 1/64 part. Not used in drums
- distortion and (for normal instruments only) the upper 4 bits of the bend are stored. In the case of a percussion instrument, it defines the full byte distortion.

Converting MIDI files to midiplay format

Once we have finished editing a MIDI file, and it sounds correctly with DAVE instruments, it can be converted so it can be listened on the EP standalone. midiconv is a command line application. On Windows, the command line interface (cmd.exe) can be accessed via the Start Menu. After entering the directory where midiconv.exe is with the CD command, it can be used as follows:

```
midiconv INFILE.MID OUTFILE.BIN ENVELOPE.BIN|ENVELOPE.TXT|-raw [IRQFREQ] [-optsort] [-renumber] [-quantN] [-biasN] [-render] [-1..-9]
midiconv ENVELOPE.TXT ENVELOPE.BIN -env [-1..-9]
```

The second form of the command converts text format instrument definitions to binary format. When converting MIDI a file, the following options can be used:
IRQFREQ (defaults to 50.0363 Hz): the assumed interrupt frequency of the playback, currently this can only be the 50 Hz video interrupt generated by the NICK chip, but other values may be useful to speed up or slow down the music. If the original tempo is 125 or 150 BPM, or other value where the duration of a quarter note is an exact integer multiple of 20 ms, it is recommended to set IRQFREQ to 50
-optsort: sorts note events that occur at the same time in a way that attempts to minimize the size of the output MIDI data, otherwise the sorting is done by channel number
-renumber: renumbers instruments starting from 0, but does not change the numbering of percussive instruments
-quantN: if N is not 0 (it can be an integer number in the range 1 to 9), then the tempo is adjusted automatically so that the duration of a quarter note becomes an exact integer multiple of N interrupts
-biasN: adjusts the rounding of MIDI event times when converted to Enterprise interrupts, N is an integer from 0 to 99 (the default is 25), 0 rounds down, 99 rounds up, and 50 rounds to the nearest integer
-N: enables compression if N is not 0, the allowed range is 0 to 9. Greater values may reduce the size of the output file, but this is not guaranteed, N above 6 will sometimes increase the file size by a small amount
-render: generates raw DAVE register data (16 bytes per frame) by emulating midiplay
After running the command, the .bin file immediately appears in the same folder, and it can be played with the midiplay.com or midi_asm.com application on the Enterprise.

The format of the converted files:

16 bytes EXOS header:
00h, 6Dh
2 bytes: file size without the header (LSB first), in the case of future compressed formats, this would be the total size of the compressed data
2 bytes: uncompressed envelope data size
2 bytes: uncompressed MIDI data size
1 byte: reserved for future versions, currently always 00h
1 byte: compression type, 0 means no compression, 2 is epcompress -m2, others are currently not supported by the player. Compressed files can only be played with midi_asm.com or mididisp.com, the C version of the player (midiplay.com) does not implement this feature yet. To unpack a compressed MIDI file, use midipack, which is a modified version of epcompress
2 bytes: compressed envelope data size, or 0 if the file is uncompressed
4 bytes: reserved for future versions, currently always 00h
envelope data (envelope.bin) in the above described format, midiconv automatically removes any unused envelopes
MIDI data, the format is similar to standard MIDI files (SMF), but it contains only channel messages on a single track without header, and the delta time between events has a fixed resolution of 1/IRQFREQ seconds. For example, the following sequence of bytes plays the 440 Hz „A" note on the first MIDI channel for one second, using instrument 11:

```
00 C0 0B 00 90 45 7F 32 45 00
```

When using the -raw parameter, the program converts to a simple headerless format that includes only the raw MIDI data. In this case, it is not necessary to specify a path to an envelope file on the command line.

INTERVIEW INTERVIEW INTERVIEW INTERVIEW INTERVIEW INTERVIEW

I am very fond of the old days of Enterprise

Perhaps we don't have to introduce István Matusa (Tutus) to anyone. He used to be the chief editor of Enterpress, the only Hungarian magazine that dealt with topics revolving around Enterprise regularly. It had been a bimonthly magazine between 1990 and 1995, but was revived in 2015 to celebrate the computer's 30th birthday. But how did it all begin back in the day? What motivated Tutus in editing a magazine? What contact did he even have with our favorite computer? Let's ask him!

Tamás Bodnár (Szipucsu): How did you first „meet" Enterprise? Was it love at first sight?

István Matusa (Tutus): Our story started in the autumn of 1987. My father worked at Kontakta Factory (factory of electric machinery parts in Hungary from 1963 to 1988) and was persuaded by one of his colleagues to buy an Enterprise. We soon bought the computer at Centrum Super-market in Csepel. Later we bought an EX-DOS interface and a 5,25-inch Videoton floppy disk drive for it. We mostly played games on the machine, but I sometimes tried to write small programs in BASIC. I am very fond of these days and my early memories of Enterprise. My only regret is that we didn't take photos of these good old days.

T.B.: What do you do for a living now? Besides your Enterprise hob-by, do you also work in informatics?

I.M.: I am working at a creative agency as system administrator, webmaster and publication designer. I am a graduate pressman and I worked as a typesetter for years. After the change of political sys-tem in 1989 though, when many printing companies were shutting down, an old colleague of mine offered me the oppor-tunity to start to work as a publication de-signer at my current workplace. Dealing with informatics has been continuous for me since then and now, as system admin, it's half of my job.

T.B.: What is your favorite topic when it comes to Enterprise? You mentioned that you worked with BASIC. What kind of programs did you try to write? Did you manage to finish any? What else did you use Enterprise for?

I.M.: What interests me most is how to handle a database. I remember spending a lot of time with dBase II running on IS-DOS (As webmaster I can make use of some of this experience even now in MyS-QL) I am also interested in editing music. I really hope we can make Zozo's MIDI card and all the software relying on it work. As for writing my own programs, I did finish one in the past. It was a photo catalogue that helped in searching for a particular negative among all the others. I wrote this in BASIC, and then Zoltán Németh (Zozo) helped me to write the program in machine code too. I was supposed to learn machine coding in the process, but I didn't manage to do so completely. Since then, I have forgotten everything I learned back then. I am planning on relearning though once I have some free time.

T.B.: You mentioned that publication de-signing is something you do on a profes-sional level now. But where did you get idea to edit an Enterprise themed magazine?

I.M.: When I saw that Enterpress – then edited by László Újlaki and Csaba Hajnal – was going to cease to exist I knew I had to do something. I was work-ing at Ameko Kft (publisher/printing com-pany) at the time. I talked to my superior (Gábor Kovács) there and he agreed to continue to publish the magazine under one condition: if there would be enough subscribers. It was a thriller: we managed to get the satisfactory number in the last minute. I had learned about magazine editing in trade school, but of course I had to settle into it. I got a lot of help from the previous editors.

T.B.: What do you know about László Újlaki and Csaba Hajnal? Who are they? How did they start magazine editing and why did they stop? Where are they now?

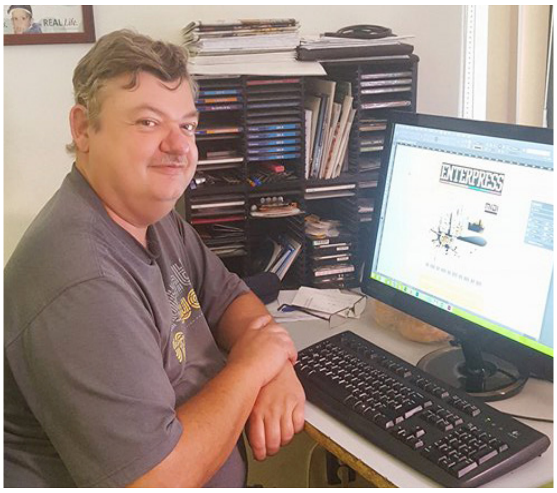
I.M.: They were simple Enterprise users too in the beginning, but later they got in touch with the Székesfehérvár-based Mátrix Kft (future publisher of Enterpress). **Csaba Hajnal** was a professional in ma-chine coding. He wrote all the pieces of the Assembly series in the magazine. They stopped because of the waning public interest. I think the bankruptcy of the English manufacturer of Enterprise was a contributing factor to this. There were no new software or hardware in de-velopment. As for the present, I am trying to reach past members of the Enterprise community all the time, but I have had no luck with the two of them so far.

T.B.: How did you find the users with the magazine?

I.M.: This was the most difficult part as there was no internet back then. We had to send actual letters to the street addresses of used-to-be subscribers. It wasn't simple.

T.B.: Where did you find editing person-nel?

I.M.: Most of them came from the



Újlaki-Hajnal team of Enterprise, but of course I needed new faces too. It was obvious to give the assistant editor-in-chief position to **Zoltán Németh (Zozo).** I had met him and his father, **József Németh (aka Apuci)** in 1991. We had lots of meetings later when we were working on Enterpress. They helped me a lot back then and Zozo does so even today. (His father passed away since then. He is greatly missed.)

T.B.: What would you say was your greatest success with the magazine and what posed the most difficulties?

I.M.: The fact that the magazine didn't disappear in 1993 was a gigantic success in itself. However, the difficulties that were present then did live on too and caused the eventual downfall of the magazine two years later. In the end we were losing public interest again. I didn't get enough material from the writers to publish and we were losing subscribers too. The last two issues I had to publish myself, because Ameko Kft didn't want to do it anymore. In the summer of 1995 I released a double issue, but after that I gave up too. There was no point in going on. It was the saddest moment in my career with Enterpress.

T.B.: But you came back with a nice surprise for the 30th anniversary of Enterprise when everyone at the birthday get-together got a new issue of Enterpress. Where did you get the idea to publish an anniversary issue in 2015?

I.M.: This is a story for the ages! Zozo was the one organizing the 30th anniversary Enterprise event. He asked me two weeks in advance if I wanted to come. He told me Bruce Tanner from England - who developed IS-BASIC (among other stuff) for Enterprise - would be there. I was really excited and it was obvious in that instant that I have to put a magazine together for this occasion! I managed to finish a 12-page issue in time. It had its shortcomings, but it was done. Me and Zozo wanted it to be a surprise and it turned out great. It was truly touching to see club members with the anniversary Enterprise in their hands. You could tell they were thinking *"I can't believe this is happening"*. This gave plenty of motivation to go on with the project and it's something that drives us ever forward.

T.B.: You said you are really interested in editing music. Have you used Enterprise for editing before? What plans do you have with the MIDI sound card?

I.M.: I haven't used Enterprise for editing music yet, but once the MIDI card and the associated software are all working fine I will surely buy a MIDI keyboard and get down to making music.

T.B.: Am I right to assume that it's much easier to edit and distribute a magazine nowadays than it was in the past?

I.M.: Yes. Editing is much easier. InDesign is a professional software. Back in the day we didn't even have Windows. I remember using the Ventura publication editor software on DOS. Distribution is also easier, because we have an online format now, but we didn't give up on the print magazine either, because there is still a need for it.

T.B.: What kind of programs or hardware does today's Enterprise still need in your opinion?

I.M.: An old dream of mine is to have a dual-pane file manager on Enterprise, like there is Total Commander on PC. I know it's not easy to create such a program, but I also know that there is more than one genius Enterprise programmer out there who could make it work. I hope one day my dream will come true. As for hardware, I really can't wait to have the EP-NET card, but we know that Bruce Tanner will soon finish the final version, so it's only a matter of time now. Going back to file managers, a built-in FTP client would also be possible once the EP-NET card is released. Also an "all-in-one" card that has a time and date chip, RAM and FLASH ROM on it would not hurt either. Of course my biggest dream is to have a completely redesigned Enterprise that would keep the exterior, but on the inside - with today's technology - everything nice and new would be on the motherboard! Of course this would require more professionals with free time and a lot of money, but I think if Spectrum Next was such a success, EP Next could be too.

T.B.: After the 2015 anniversary meeting you didn't only revive the magazine, but restarted the Enterprise Club too. How hard is it to organize a club meeting? Who is helping you and how? Who brings the computers? How many members does the club have and how many of them show up actually at a meeting?

I.M.: I thought this was important too. It's always good to see our old EP friends, but there are also a lot ideas being born at a club meeting. Fortunately we managed to find a really good and cheap venue in Budapest at Nyugati square. At the first meeting in November 2016 we had so many people there that the landlady was in quite a shock too. I always get help in organizing the events and we split the task of bringing computers and monitors. The club already has two easy-to-carry, slim CRT monitors. We have 43 registered members, which is a little low, but I hope in 2018 this will change into the positive direction. Many of these 43 people are from outside of Budapest, but an average number of 20-25 are usually present at our meetings.

T.B.: What do you think of today's Enterprise community?

I.M.: They are fantastic! I often check the online forum, but the buzz is so intense that I can barely follow the news. People find solutions to each other's problems in the blink of an eye. They develop programs and games together and they talk it out on the forum. Too bad they don't have more time for our beloved computer. Of course we are talking about a generation that is 40-50 years old now so it's hardly anyone's fault that they don't always have time for this. It's just a hobby for everyone.

T.B.: Thank you for your work with Enterpress and the EP Club in the name of all Enterprise fanatics! We wish you all the best for the future!
I.M.: Thank you very much!

EXOS Compatible Memory Management - part III.

Written by: Zoltán Németh (Zozosoft)

What we need is to get rid of fixed segments so we should be able to write programs fully adaptable to different configurations. Let's see what to do if we need memory.

To start, there is the earlier mentioned page-0. There, EXOS uses the 0030-005BH area, and the header program 5 is loaded at 0100H. The rest of memory can be used freely for the currently active user program.

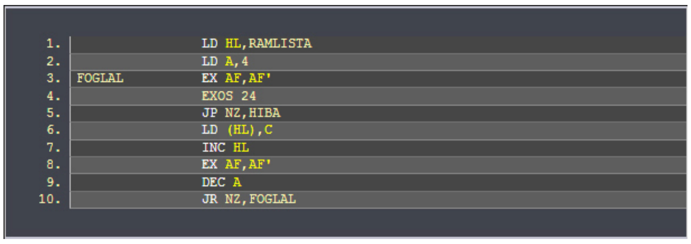
As I mentioned earlier, the segment number of the page-0 may be different in the different configurations, the current number can be read from the B0H port, the EXOS system variable* at BFFCH. This will be important later when we start to get a bit tricky with page-0, but basically you better do not want to poke it

* The system variable points to the valid address only if the FFH system segment is accessed on page-2.

Let's look further, what if page-0 is not enough, or doesn't meet our needs?

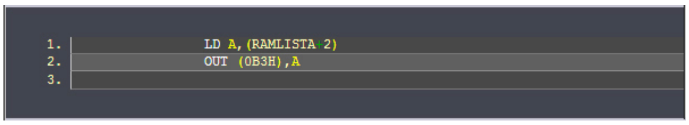
It is very simple, we request additional segments from EXOS. Since we are no longer stuck to fixed segment numbers, things become a lot more simpler, everything we get is good for us . It is convenient to store the number of received segments, it doesn't harms us to know which segments are involved on the paging operations , and also, we must release them at exit.

One possible method, for example, is to employ 4 segments:



```
                LD HL,RAMLISTA
                LD A,4
FOGLAL          EX AF,AF'
                EXOS 24
                JP NZ,HIBA
                LD (HL),C
                INC HL
                EX AF,AF'
                DEC A
                JR NZ,FOGLAL
```

Later on, for example, segment 3 can be referenced like this:

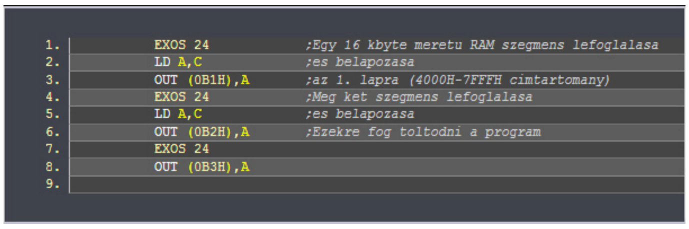


```
                LD A,(RAMLISTA+2)
                OUT (0B3H),A
```

Here I must also mention that, at the beginning of the Spectrum World series of articles, when discussing the Spectrum cassette scan program, it is too a matter of correct memory management:

„EXOS 24 - Segment selection. It is advisable to request memory from EXOS, as this certainly will prevent other programs to start to work in the memory we want to use. The C register will store the number of allocated segments.

Our program occupies 3 segments, which is 48k so that every SPECTRUM program can fit into that size".



```
        EXOS 24         ;Egy 16 kbyte meretu RAM szegmens lefoglalasa
        LD A,C          ;es belapozasa
        OUT (0B1H),A    ;az 1. lapra (4000H-7FFFH cimtartomany)
        EXOS 24         ;Meg ket szegmens lefoglalasa
        LD A,C          ;es belapozasa
        OUT (0B2H),A    ;Ezekre fog toltodni a program
        EXOS 24
        OUT (0B3H),A
```

But there is no error handling, so in case of insufficient memory, no matter if EXOS alerts us that there is a shortage, the program will not notice it and will malfunction.

This smooth segment-on-demand method can serve you perfectly when you don't need screen management or if your program works through the EXOS Video Driver. If we want to create the graphics by ourselves, we will need its own video segment(s) with its own LPT table.

And here begin the nuisances...

What is the complication with video segments? This also answers one of the questions Povi asked: we can not directly request a video segment, so we are forced to invoke the already familiar cycling segment-claim method.

„Until finally we found e.g. our wished FC segment"

This is again an incorrect attitude. We can look for it, but we will not get it from EXOS on an EP64 as that segment is already assigned to page-0.

So the correct attitude is that FC or greater is acceptable. Well, let's take this number with caution, where's the problem here? This will add us some extra work here!

If a general purpose segment is needed, it doesn't matter which one we receive. In the case of a video segment this is no longer true!

To know why, first let's look at Nick's chip memory, which has not been discussed so far:

We already know that the 64K on motherboard is also video memory. Nick is able to read it directly without Z80 intervention, and the addresses it uses are called video addresses. Of course, they may also be in the range 0000-FFFFH.

But these addresses are completely independent of the Z80 addresses!

The 0000-3FFFH range represents the FCH segment, 4000-7FFFH is FDH, 8000-BFFFH is FEH, C000-FFFFH is FFH. And these addresses are independent of where the segments in question are accessed by the Z80 or whether they are accessed at all.

So when we're working on video memory, it may be necessary to get the address of the area that the Z80 is accessing and convert it to a video address for Nick.

And there is another computation, needed when we give the address of the LPT table to the Nick chip, because here we only pass it 12 bits.

The easiest way this can be achieved is placing the LPT table at the 0000H video address, i.e. at the beginning of the FC segment, as the SpV Spectrum loader does.

In the LPT table, you will also need to enter the video address of the screen data. For the sake of simplicity, that loader uses the FD segment for this purpose, so the screen will begin with a 4000H video address, which is the starting point used by the LPT generating routine.

But our goal is to get rid of fixed segments!

To produce the Spectrum screen, you will need two video segments, one for the LPT table, another for the Spectrum screen, which will also be the first 16K of Spectrum RAM(?) memory.

The LPT table does not fill a full segment (the standard Spectrum LPT table is 3200 bytes long), so it can be placed in the EXOS shared segment. Therefore, the preferred method is to request segments by the known method until a video one is obtained, we add this for the Spectrum screen and the next to LPT. If we do the reverse, we might get a full segment for only the LPT table, and the ZX video memory, which requires a whole segment, may receive only a smaller zone in the shared segment, causing an error...

On the 128 machine, a reverse configuration is created as in the SpV loader: the FCH becomes the video segment, while the FDH is the LPT segment. With this, the original Spectrum LPT builder routine, that has become inoperable, needs to be modified.

How to calculate video addresses:

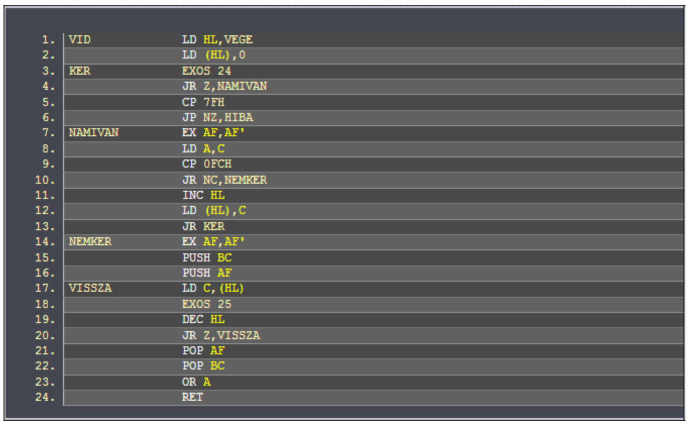
I have the Z80 address, along with its segment number.

From the Z80 address, the top two bits are „removed" and replace the bottom two bits of the segment number.

When we write the LPT address to Nick, then the bottom 4 bits should be „thrown away".

Let's continue our article with more practical parts.

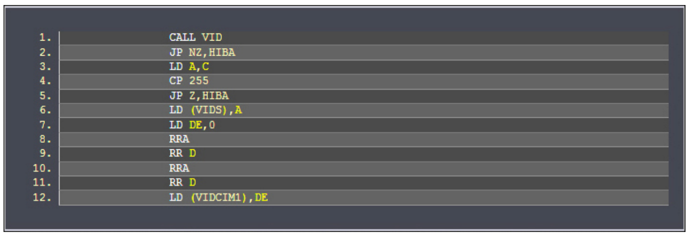
This is the routine I use on the Spectrum conversion loaders to request video segments:



```
VID             LD HL,VEGE
                LD (HL),0
KER             EXOS 24
                JR Z,NAMIVAN
                CP 7FH
                JP NZ,HIBA
NAMIVAN         EX AF,AF'
                LD A,C
                CP 0FCH
                JR NC,NEMKER
                INC HL
                LD (HL),C
                JR KER
NEMKER          EX AF,AF'
                PUSH BC
                PUSH AF
VISSZA          LD C,(HL)
                EXOS 25
                DEC HL
                JR Z,VISSZA
                POP AF
                POP BC
                OR A
                RET
```

The routine stores the number of the unnecessary segments after the VEGE address (logically after the program code). Once there is one segment which can be shared, the excess of memory is returned to EXOS.

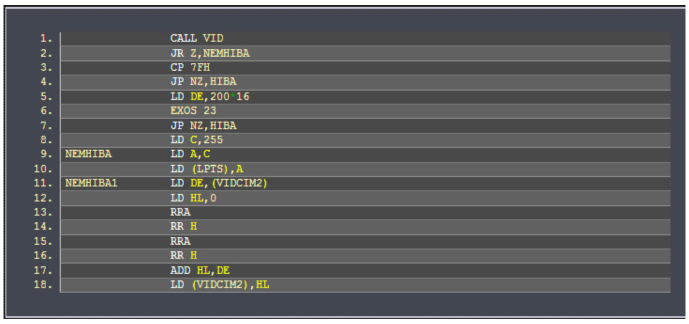
First, we need a full segment for the screen:



```
                CALL VID
                JP NZ,HIBA
                LD A,C
                CP 255
                JP Z,HIBA
                LD (VIDS),A
                LD DE,0
                RRA
                RR D
                RRA
                RR D
                LD (VIDCIM1),DE
```

From the segment received, we calculate the start address of the area, and this will be needed later when the LPT is generated.

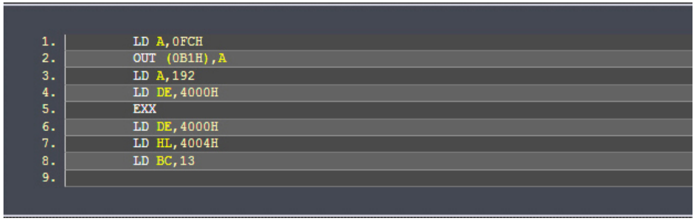
Later we request an LPT segment, which can be shared even if the LPT table is fit (ie the EXOS border can be set to the appropriate location).



```
                CALL VID
                JR Z,NEMHIBA
                CP 7FH
                JP NZ,HIBA
                LD DE,200*16
                EXOS 23
                JP NZ,HIBA
                LD C,255
NEMHIBA         LD A,C
                LD (LPTS),A
NEMHIBA1        LD DE,(VIDCIM2)
                LD HL,0
                RRA
                RR H
                RRA
                RR H
                ADD HL,DE
                LD (VIDCIM2),HL
```

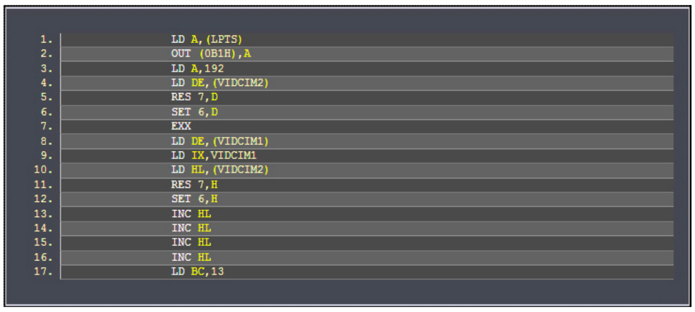
Here, too, we calculate the video address, this time the beginning of the LPT table. An attentive observer can notice that the value on VIDCIM2 is still in play here. This is basically zero, but on a 64K machine the memory will be rearranged so that the LPT table is not at the beginning of the segment, so the required offset rate has already become too high for VIDCIM2.

Next, let's look at how the LPT generating routine published in SpV also changes, the start of the original routine:



```
LD A,0FCH
OUT (0B1H),A
LD A,192
LD DE,4000H
EXX
LD DE,4000H
LD HL,4004H
LD BC,13
```

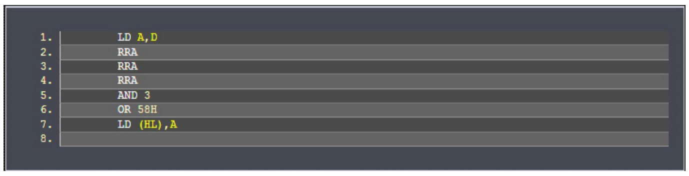
And this would happen:



```
LD A,(LPTS)
OUT (0B1H),A
LD A,192
LD DE,(VIDCIM2)
RES 7,D
SET 6,D
EXX
LD DE,(VIDCIM1)
LD IX,VIDCIM1
LD HL,(VIDCIM2)
RES 7,H
SET 6,H
INC HL
INC HL
INC HL
INC HL
LD BC,13
```

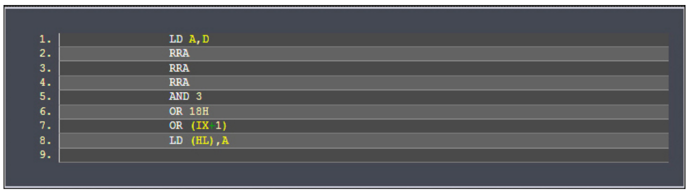
Instead of the fixed segment number, the stored one is used. DE will point at the beginning of the LPT on Tab 1, but due to the possible offset mentioned above, this will be counted from the LPT video title to the 1st address. Then, after EXX, DE will show the beginning of our video memory, which was in the original routine fixed to 4000H, as the FDH was fixed for this purpose. HL will also display the LPT with a 4 byte shift, and it is used for addressing LPT rows.

There is no mention to IX, it points to the variable that stores the address of our video memory. Because of this, we need to do more changes to the original routine, even when calculating the attribute addresses:



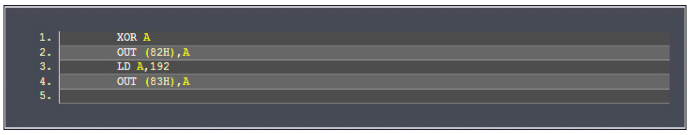
```
LD A,D
RRA
RRA
RRA
AND 3
OR 58H
LD (HL),A
```

Modified:



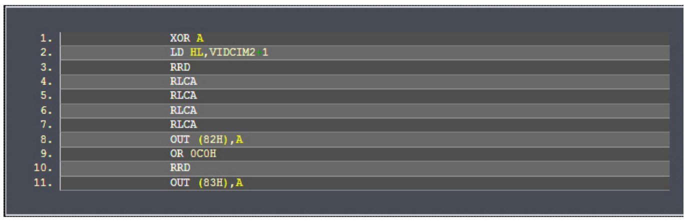
```
LD A,D
RRA
RRA
RRA
AND 3
OR 18H
OR (IX+1)
LD (HL),A
```

Then only the LPT activating OUT instructions have to be modified, the original is very simple as it uses the fixed 0000H video address:



```
XOR A
OUT (82H),A
LD A,192
OUT (83H),A
```

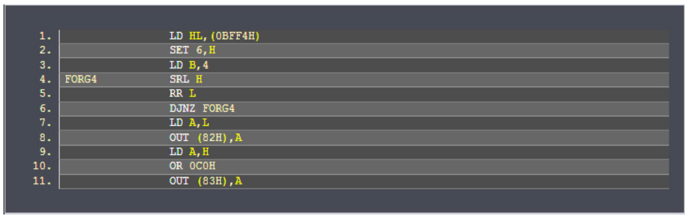
And modified to use our calculated LPT address:



```
XOR A
LD HL,VIDCIM2+1
RRD
RLCA
RLCA
RLCA
RLCA
OUT (82H),A
OR 0C0H
RRD
OUT (83H),A
```

Note, this method spoils the VIDCIM2 variable, but it is no longer needed.

But here is another method used here to restore the original EXOS LPT:



```
        LD HL,(0BFF4H)
        SET 6,H
        LD B,4
FORG4   SRL H
        RR L
        DJNZ FORG4
        LD A,L
        OUT (82H),A
        LD A,H
        OR 0C0H
        OUT (83H),A
```

The address should be read from EXOS's LP_POINTER variable (so, the system segment is positioned on page 2). The next SET 6 is there to convert the EXOS variable to video address, but if we use our own calculated video address, then this is not necessary.

If you are interested, I can continue with how we can have 3 free-to-work segments for a 48K Spectrum simulation on a 64K machine when only two are free, and even the LPT board should have one.

To be continued!

HiSoft Pascal

Part 3

**Written by:
Zoltán Povázsay
(Povi)**

As I promised in the previous chapter, this section is about graphics. There are two possibilities to exploit the graphical capabilities of the machine: one when creating a video page for ourselves, at a fixed memory address and completely dominating the machine.

In this case, we have to re-write all the already written graphic routines (drawing pixels, lines, circles, etc.) that EXOS provide to us, but in return they will be much faster. István's GRAPH16.HPU library, which also supports mouse management, is a very good example of this(hopefully Lacika would write an article or at least one description).

The other, a simpler method, is when we rely on EXOS: in this case, EXOS function calls are used to open and close the video channel, to display the video page, and to use the escape sequences interpreted by the video channel for drawing. The „black book" (aka EXOS Technical Description) and the methods described in the previous chapter (calling the EXOS functions from Pascal, accessing the arguments of Pascal procedures and functions), will help us, almost as a child's play, to perform the calls to the graphic subroutines supplied by EXOS from inside a Pascal program. A similar article already appeared in the ENTERPRESS columns (January-February 1995, Zoltán Horváth), unfortunately, with some cracking solutions.

In this chapter, I will use the latter method, which will not be complete but, based on the examples, anyone can complete, supplement, modify, and enhance the described procedures.

In the first round, let's see how to display a four colour high-definition graphics page in the same way as the IS-BASIC GRAPHICS command! First of all, by writing the EXOS variables, we will set the parameters of the new video channel before opening it. Fortunately, with HiSoft Pascal 1.2, we will be able to use the SetVar method:

```
SetVar (22, 1)  { high resolution graphics};
SetVar (23, 1)  { four-colour mode};
SetVar (24, 40) { 40 characters wide};
SetVar (25, 20) { 20 characters high};
```

The next step is to open the video channel. This can be done with the EXOS 1 function call, which requires two parameters: the channel number in register A and the address with the „VIDEO:" string in the DE register. Fortunately, this string can be found at address 0x148a, which can be used without a doubt.

It would be good to see if opening the page was successful. Any EXOS call (including the channel opening function) clears the zero flag bit and sends an error code in the A register if it failed. We can solve this problem by ourselves, but in this case, we can also use the routine at 0x022c to print the error message and finish the program execution.

Let's see the implementation!

```
procedure OpenVideoChannel(ch: integer);
  inline(#dd,#7e,#02) { ld   a, (ix+2)  };
  inline(#11,#8a,#14) { ld   de, 148ah) };
  inline(#f7,#01)       { exos 1 };
  inline(#c2,#2c,#02) { jp   nz, 022ch  };
end;
```

Of course, we also must ensure that the channel is closed, because if not, the program will stop with the „Channel exists" error message at the next execution time:

```
procedure CloseChannel(ch: integer);
begin
  inline(#dd,#7e,#02) { ld   a, (ix+2) };
  inline(#f7,#03)     { exos 3 };
end;
```

There is nothing more to do than displaying the video page. Therefore, we make the call to the BASIC DISPLAY command.

```
procedure Display(ch, dAt, dFrom, dTo:
integer);
begin
  inline(#dd,#7e,#08) { ld   a, (ix+8) };
  inline(#06,#01)     { ld   b, 1  };
  inline(#dd,#4e,#04) { ld   c, (ix+4) };
  inline(#dd,#56,#02) { ld   d, (ix+2) };
  inline(#dd,#5e,#06) { ld   e, (ix+6) };
  inline(#f7,#0b)     { exos 11 };
  inline(#c2,#2c,#02) { jp   nz, 022ch };
end;
```

Iteration with the video page is done by writing escape sequences (see the black book). Line drawing requires at least the following three procedures: „positioning beam" (escA <xx> <yy>), „beam on" (escS) and „beam off" (escs).

The latter two are the simplest because they do not expect a parameter (outside the channel number), only two bytes are written on the video page:

```
procedure SetBeamOn(ch : integer);
begin
  inline(#dd,#6e,#02) { ld   l, (ix+2) };
  inline(#7d)           { ld   a, l  };
  inline(#06,#1b)       { ld   b, 27 };
  inline(#f7,#07)       { exos 7 };
  inline(#7d)           { ld   a, l };
  inline(#06,#53)       { ld   b, 'S' };
  inline(#f7,#07)       { exos 7 };
end;
```

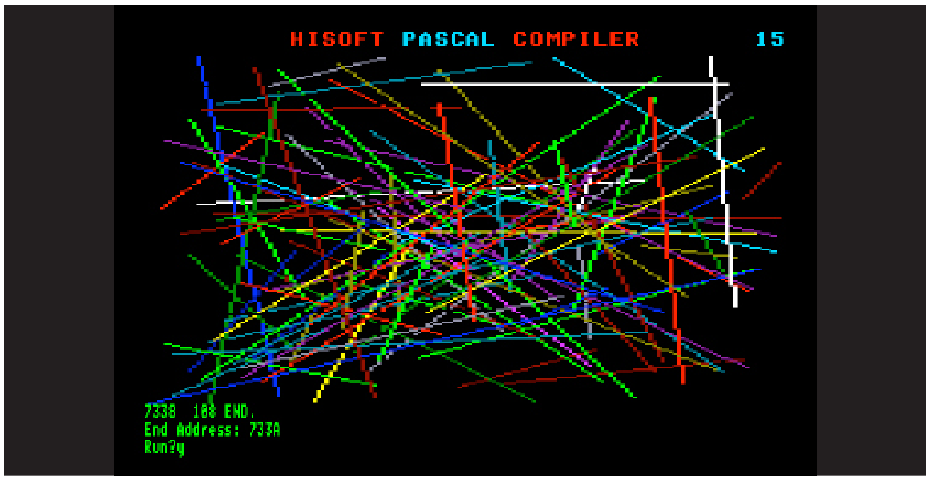
The SetBeamOff procedure only differs from this on the penultimate line, where inline(#06,#53) is changed to inline(#06,#73).

Let's look at the beam positioning!

```
procedure Plot(ch, x, y : integer);
begin
  inline(#dd,#6e,#06) { ld   l, (ix+6) };
  inline(#7d)         { ld   a, l };
  inline(#06,#1b)     { ld   b, 27 };
  inline(#f7,#07)     { exos 7 };
  inline(#7d)         { ld   a, l };
  inline(#06,#41)     { ld   b, ‚A' };
  inline(#f7,#07)     { exos 7 };
  inline(#7d)         { ld   a, l };
  inline(#dd,#46,#04) { ld   b, (ix+4) };
  inline(#f7,#07)     { exos 7 };
  inline(#7d)         { ld   a, l };
  inline(#dd,#46,#05) { ld   b, (ix+5) };
  inline(#f7,#07)     { exos 7 };
  inline(#7d)         { ld   a, l };
  inline(#dd,#46,#02) { ld   b, (ix+2) };
  inline(#f7,#07)     { exos 7 };
  inline(#7d)         { ld   a, l };
  inline(#dd,#46,#03) { ld   b, (ix+3) };
  inline(#f7,#07)     { exos 7 };
end;
```

It is a more elegant solution if we do not write the escape sequences in a row as bytes, but we send it all to an exos 8 call, so instead of the six exos function calls, we would only need one. This will also bring us some acceleration, as every exos call has a pretty big overhead. To do this, you first need to write the bytes in a buffer. For this purpose, the 127 bytes buffer at Pascal 0x15ae address is perfectly suited for this purpose:

```
procedure Plot(ch, x, y : integer);
begin
  inline(#21,#b3,#15) { ld   hl, 15aeh + 5};
  inline(#dd,#7e,#03) { ld   a, (ix+3) };
  inline(#77)         { ld   (hl), a };
  inline(#2d)         { dec  l };
  inline(#dd,#7e,#02) { ld   a, (ix+2) };
  inline(#77)         { ld   (hl), a };
  inline(#2d)         { dec  l };
  inline(#dd,#7e,#05) { ld   a, (ix+5) };
  inline(#77)         { ld   (hl), a };
  inline(#2d)         { dec  l };
  inline(#dd,#7e,#04) { ld   a, (ix+4) };
  inline(#77)         { ld   (hl), a };
  inline(#2d)         { dec  l };
  inline(#36,#41)     { ld   (hl), ‚A' };
  inline(#2d)         { dec  l };
  inline(#36,#1b)     { ld   (hl), 27 };
  inline(#eb)         { ex   de, hl };
  inline(#dd,#7e,#06) { ld   a, (ix+6) };
  inline(#01,#06,#00) { ld   bc, 6 };
  inline(#f7,#08)     { exos 8 };
end;
```



```
HISOFT PASCAL COMPILER          15
7338  108 END.
End Address: 733A
Run?y
```

By using the above three procedures, you can make a line drawing routine on your own:

```
procedure Line(ch, x1, y1, x2, y2 : integer);
begin
  SetBeamOff(ch);
  Plot(ch, x1, y1);
  SetBeamOn(ch);
  Plot(ch, x2, y2);
end;
```

It would be good if we could write text on the graphic page, but without having to make a separate process, only using the existing Write and WriteLn procedures! You should know that these procedures are written for EDITOR channel 121. Fortunately, this default channel number can be overwritten at any time in our code at 0x0328. A beneficial side effect of this „hack" is that the scope of the GotoXY and ClrScr procedures will apply to the given channel! What you need to do is to reset the default channel to 121 at the latest command before exiting the program!

Let's look at how to exploit these writing escape sequences:

```
procedure Circle(ch, x, y : integer);
begin
  poke(#0328, chr(ch));
    write(chr(27), ‚E', lo(x), hi(x), lo(y),
hi(y));
  poke(#0328, ‚y');
end;

procedure SetInk(ch, color : integer);
begin
  poke(#0328, chr(ch));
  write(chr(27), ‚I', chr(color));
  poke(#0328, ‚y');
end;
```

Is more beautiful and easier to read this code than the Plot procedure outlined above, but do not forget that it will result in a much slower code, so it is not worth using it! On the contrary, I prefer the insertion of the two lines on the above two procedures based on the example of Plot

Finally, a very simple demo program (without procedures, they must of course be written to the beginning of the code between „program" and „begin" lines):

```
program grdemo;
begin
 SetVar(22, 1)       { graphics hi-res };
 SetVar(23, 2)      { 16 colours mode };
 SetVar(24, 40)     { 40 characters wide };
 SetVar(25, 20)      { 20 characters high };
OpenVideoChannel(1);  { open the video
                              page };
  Display(1, 1, 1, 20); { display video page };
  repeat
    SetInk(1, trunc(random * 15) + 1);
   Line(1, trunc(random * 1280), trunc(ran-
dom * 720),
      trunc(random * 1280), trunc(random
* 720));
  until inch <> chr(0);
  CloseChannel(1);
  User(#01ec);
end;
```

In the last line, the User procedure reviews the 120-text video page. The routine at this address is the same as Display (120, 1, 1, 24).

This is necessary because without this, the top 20 lines of the text video sheet will no longer be displayed after the graphics video is closed.

The next part of this series will be file management.

Some amendment to DAVE's documentation

A0h-A5h ports

The operation of the sound generator: a 12-bit counter counts backwards from the specified frequency at 250 kHz (this concerns the 4 MHz machine if the first bit of the BFh port is not set). When overflow happens (e.g. -1 would come after 0), then 12-bit value will come into the counter instead of -1 again in the registers and one of the following occurs depending on the polynomial counter distortion:

- if there is no distortion, the (flip-flop) output of the square-wave generation is simply switched (átbillen?),

- if there is some distortion, the current output of the selected polynomial counter (pseudo-random number generator) will come to the output; the polynomial counters are continuously running at 250 kHz clock speed, which means, for example, that if the set frequency is the integer multiple of the polynomial counter length (e.g., for a 4-bit counter it is 15-1, 30-1, 45-1, etc.) , there will be no sound, because the same value will always remain when sampling.

The high pass filter means that ..... -XXX- The overflow sensor means that the flip-flop output is set to 0 for all the output(?) bits of the output channel (including any distortion, pitch, and ring modulator). Therefore, if the frequency of the clock signal channel is greater than the sampling channel, the frequency doubles.

The ring modulation occurs after the floating-point sounding (if enabled) and XNOR performed to the output of the other output channel (possibly already distorted, filtered or ring-modulated).

Interestingly, sound generators do not work with the highest (0) frequency, so it is not possible to produce a 125 kHz quadrature. The counter may be running at this time (e.g. for interruptions, but such a quick interruption does not make much sense) and it may be that a polynomial counter is selected, but this should be verified on a real machine.

A6h port

The polynomial counter used for the noise channel is upgraded to the falling edge of the clock signal channel output(possibly distorted, filtered, ring-modulated) and not at a fixed 250 kHz speed as in voice channels. The 31.25 kHz mode corresponds to the use of a 4-1 frequency clock signal.

The 4th bit (7 and 17 bit counter swapping) means that an optional 9/11/15/17 bit counter can be used for voice channels instead of 7 bits (and this will also be a 250 kHz clock speed) , and the polynomial counter of the noise channel will be of 7 bits (250 kHz instead of the optional clock speed).

At the noise channel, the order of the effects will be (if enabled):

- low pass filter: only the output of the polynomial counter (but otherwise running at the set frequency) of the channel 2 (distorted, filtered, etc.) is redirected to the output, instead of the falling edges of the clock.

- high pass filter: similar to the audio channels: on the downward edges of the channel 0, the output of the storage is set to 0 until the next polynomial counter is sampled.

- ring modulation: As with audio channels, XNOR is performed on channel 1.

Polynomial counters

The DAVE chip contains 4 polynomial counters:

- a 4 bit one that always runs at 250 kHz clock speed and can only be used by voice channels,

- a 5 bit one that always runs at 250 kHz clock speed and can only be used by voice channels,

- a 7 bit one which operates at 250 kHz clock speed and can be used by the audio channels (A6h port bit 4 = 0), or the gate channel input (A6h port bit 4 = 1)

- an optional length of 9/11/15/17 bit which either operates at the clock speed of port A6h (A6h port bit 4 = 0) or clock speeds of 250 kHz and can be used by voice channels (A6h port bit 4 = 1).

The counters work when their output is upgraded:

The value of the counter is shifted to the left by a bit by the result of the XOR operation between the two bit values of the new bit 0 output(selected for the given counter) output will be the new bit 0. The bits between which the XOR operation occurs, i.e. the binary „polynomial „

For 4 bit counters, bit 3 and bit 2

For 5 bit counters, bit 4 and bit 2

For 7 bit counters, bit 6 and 5

For 9 bit counters, bit 8 and 4

For 11-bit counter, bit 10 and bit 8

For 15 bit counters, bit 14 and 13

For 17 bit counters, bits 16 and 13

The pseudonym number sequence is 2 ^ N-1 (ie, for a 5 bit counter, eg 31), because the counter can not be 0 because in the endless cycle only 0 can be the output (0 XOR 0 = 0). At startup, maybe every bit = 1, but this can not be known, and there is not much significance.

A7h port

b0-b2: Activating the „synchronization" here means that:

- the audio channel counter is not running

- keeps the counter continuously at the frequency set in registers A0h-A5h

- the flip-flop output of the square generator is set to 0 b3-b4: when the D / A mode is switched on, the output on the A8h or ACh port is set to four times (ie its effect is like the output of all four channels is continuously 1 and the their volume would be the same

with channel 0)

5-b6: here interruption is not sound generators distorted, cracked, etc. output, but the counter of the given sound generator directly, after which every 0 has passed, the bit 0 of the B4h port is passed and, if enabled, the bit 1 is set. The 1 kHz interruption corresponds to 250-1 sound generator frequencies and 50 Hz to 5000-1, that is, not the video interrupt speed, but exactly 80000 Z80 cycles (or 120000 if bit BFh port 1 is set).

B4h port:

bit 0-1: 50 Hz / 1 kHz / sound generator interruption

bit 2-3: 1 Hz interruption

bit 4-5: video interrupt

bit 6-7: (not used?)

Bits 0, 2, 4, and 6 are always readable, even if that interruption is disabled; In these bits, 1’can be enabled, write 0 ,, interrupt can be disabled.

Bits 1, 3, 5, and 7 only work when a proper interrupt is enabled, otherwise it is always 0; if interruption is enabled then bit 1 and 3 are set to each (up and down) edge of bit 0 and bit 2 and bit 5 and 7 are only for the downward edges of bit 4 and 6.

For Z80, the interrupt request is the result of the OR operation between the 1st, 3rd, 5th and 7th bits; these bits can either be erased by 1 ,bit or disable the given interrupt.

Otherwise, until all 4 bits are erased, the Z80 generates a continuous interruption. Bit 4 comes from NICK and is a VINT bit copy of the current LPB in that line; I think NICK rewrites LPB at the beginning of every line (but maybe it would be better to check on the real machine).

This also shows that the video interruption occurs only at the beginning of the first, first VINT bit-free line after the interrupt request LPB. The same reason is that there can be no video interruptions in two consecutive queues.

B5h port:

DAVE 3 provides address decoding for external I / O devices, combined with writing / reading signals, simplifying the structure of the external device.

The 3 I / O ports are B5h, B6h, B7h, each with a separate control signal for writing and reading, these are the outputs of the DAVE WR0 / RD0, WR1 / RD1, WR2 / RD2, connected to the various outputs and inputs , tape recorder, printer, serial port, joystick). RD2 is not used! (This could have made it easy to build an 8-bit input into the machine, such as connecting an A / D converter chip to the cassette deck and reading it by RD2, the digitizer might have been ready ...).

When writing B5h (WR0), 8 bits are stored in the U25 signal in the 74LS273 IC storage. This 8 bit is used by several devices:

bit 0-3: keypad matrix selection (only line 0 to 9 is a key)

This allows you to select a row of keyboard matrices. This is done by the U26 signal 74LS145 IC, which is a BCD decoder with 10 enable outputs. If this 4-bit value is between 0h-9h, then the correct output is set to 0. None of the outputs between Ah-Fh is active. Output 10 is the KB0-KB9 signals, these are for the 10-pin connector of the foil, and are also output to the CONTROL 1/2 connectors, sharing KB0-KB4, CONTROL 2 with the CONTROL 1 connector KB5-KB9.

Here, however, there was a possibility for development if the later bits of Enterprise would have been completely decoded by 4 bits, with a more complicated, multi-button keyboard having up to 16x8 or 128 buttons.

The remaining 4 bits pass through a single inverter, meaning that every single value of 1 activates the function:

bit 4: STROBE output to printer port

bit 5: Turning off the tape (if bit 1, no sound)

bit 6: Activate REM1 (if bit 1, REM1 is on)

bit 7: Activate REM2 (if bit 1, REM2 is on)

Reading:

The U27 sign reads the selected line of the keyboard matrix via IC 74LS373:

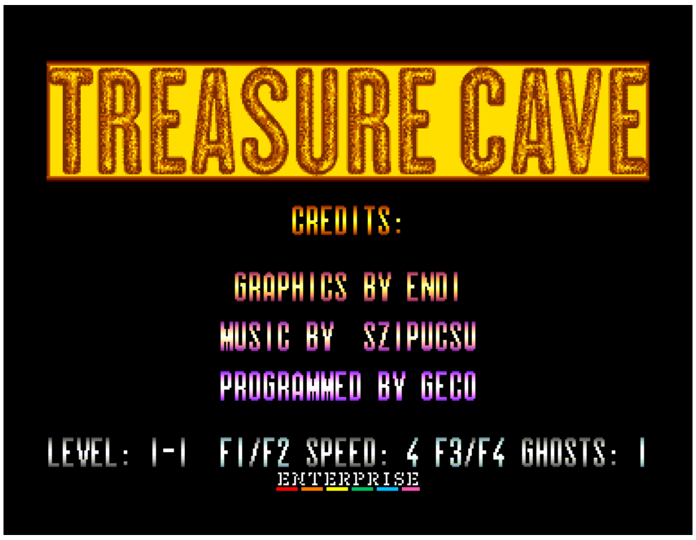
bit 0-7: the current status of the selected row of the keyboard matrix; if a key is pressed, the corresponding bit is 0 ,, otherwise, 1’.

To be continued!

IstvanV

Sor	**Bit**							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	N	\	B	C	V	X	Z	SHF_L
1	H	LOCK	G	D	F	S	A	CTRL
2	U	Q	Y	R	T	E	W	TAB
3	7	1	6	4	5	3	2	ESC
4	F4	F8	F3	F6	F5	F7	F2	F1
5	8		9	-	0	^	ERASE	
6	J		K	;	L	:	]	
7	STOP	le	jobb	fel	PAUSE	bal	ENTER	ALT
8	M	DEL	,	/	.	SHF_R	SPACE	INS
9	I		0	@	P	[		

Treasure Cave

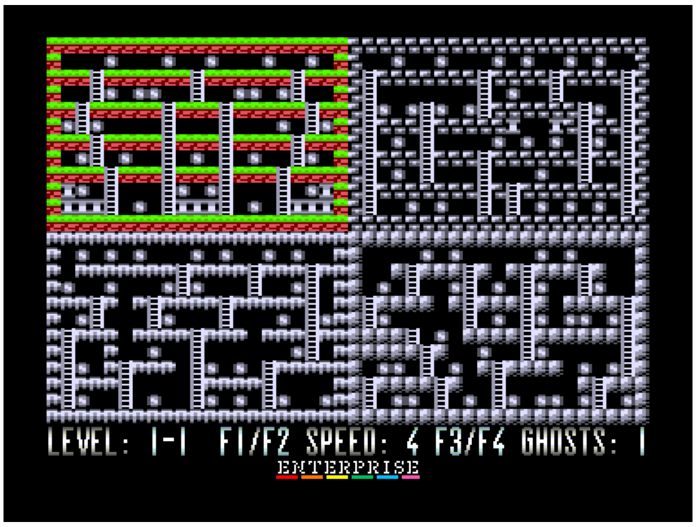


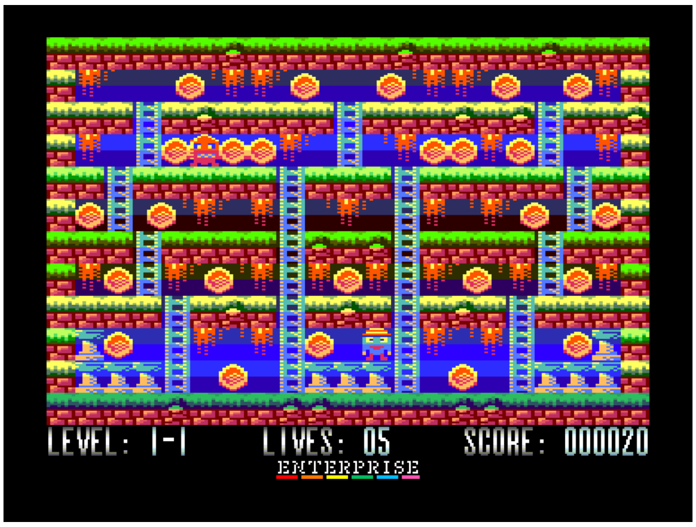
No Enterprise DevCompo needed! It's enough if three main members of the enterpriseforever.com forum unite their forces and an original Enterprise game is created!

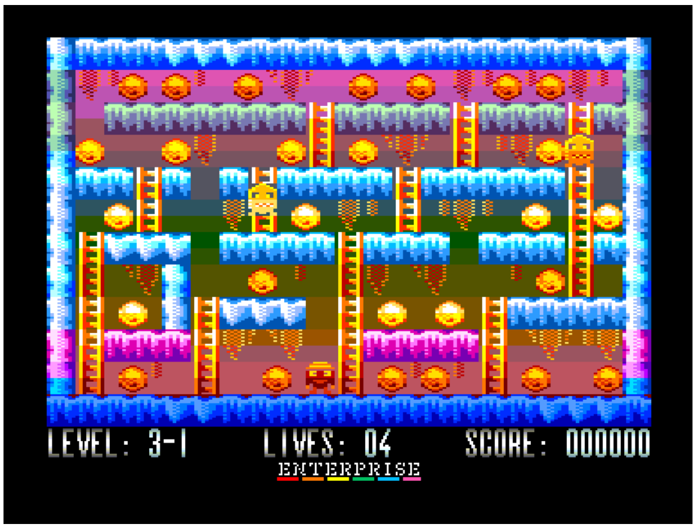
As it started:

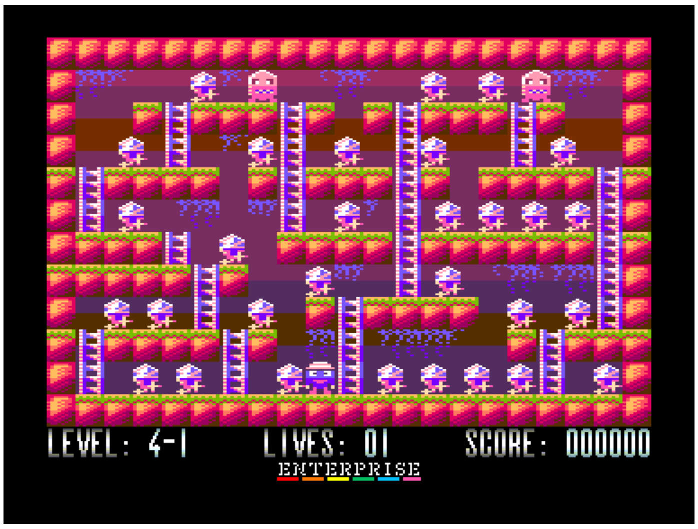
Endi: *"here is only some graphics... What great levels could be made with this, some scroll up and down, it could be with pixels too."*
Geco: *"very nice, something could be made from this." "This will be speccy music at the best, or if Szipucsu feels like writing some music, that will be."*
Szipucsu: *"it can be ok but I can only write music written in basic DATA lines."*

And here it is, a fantastic game has been created in the base of these conversations! Congratulations, guys, we wish you a lot of time, strength and endurance so that a lot more like this could be created!

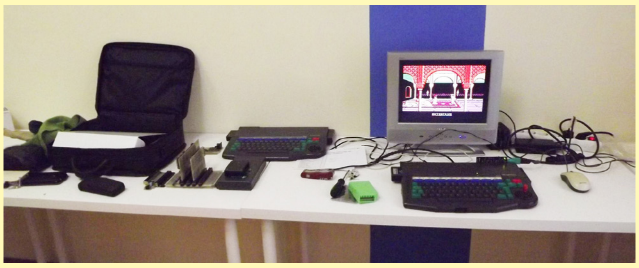


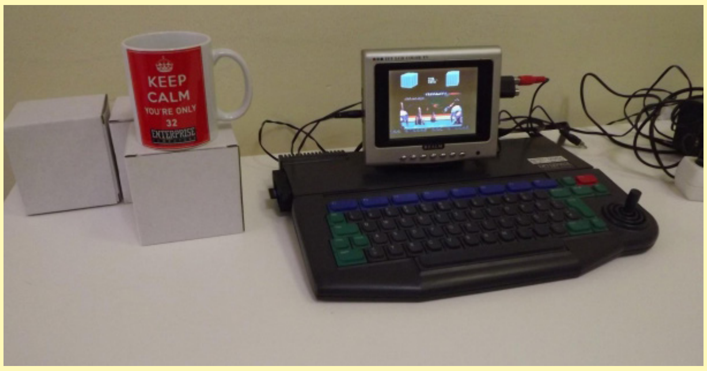






Enterprise club, 20, may 2017

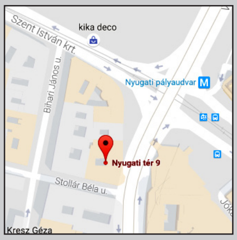




ENTERPRISE KLUB

Six times a year

Location:
Hungary
Budapest (V. ker.)
Nyugati tér 9.
Skála terem
2 pm. – 7 pm.



Information: www.enterpriseklub.hu

If you want to be an editor of ENTERPRESS Magazine, send an article, game description, game info, or anything related to the Enterprise computer!

You can send articles to this email address:

E-mail address:

info@enterpress.news.hu

ENTERPRISE FOREVER

https://enterpriseforever.com

ENTERPRESS Magazine - 2017, 4, July-August
Editor: István Matusa
Editor fellow: Zoltán Németh (Zozosoft)
The staff: geco, Povi, László Kiss, SzörG, szipucsu, lgb
English translate: gflorez, Máté Hajdó, szipucsu
Design and printing preparation: István Matusa
Web: http://enterpress.news.hu
E-mail: info@enterpress.news.hu

You can pay here for the printed and electronic versions here:
http://enterpress.news.hu/shop