ENTERPRESS

Magazine for the Enterprise users

2017, 2-3, March-May

Hisoft Pascal

SIDBASIC

Adventure

External colour input card

Time

- an enemy or a good friend?

Written by István Matusa (Tutus)

Most of us met the Enterprise computer for the first time when we were 18-20. At that time, being so young, almost everyone had more free time for the machine. However, long ago we were just looking and looking forward to get new games, utilities, hardware accessories for the newly bought computer. At that time we didn't know that nothing would happen. But if we knew it the situation would be different now.

The answer is obvious from the title of our article: Time is our biggest enemy! Most of us either have a family or work full time their whole life or both. Thus there is very little time to write programs or develop hardware. So the best solution will remain for which there have been numerous examples on the enterpriseforever.com page: more people write programs with collaboration. This way it also takes quite a long time to complete a work but this is comprehensible.

Enterprise DevCompo #2

The lines of the previous paragraph may fit here too. 80% of the present works are small demos, the rest is conversion. New software wasn't created though it would have been the essence of this competition. If one is organised next year too, the rules might be thought over. As for me, I would declare the competition only for original Enterprise software!

The badly planned club membership

I didn't think that so few people would subscribe for the Enterprise Club and Enterpress Magazine (24 1088k subscriptions, 3 128k subscriptions and 14 free 64k subscriptions arrived)...

Let's explain: 6 club meetings have been organized in 2017. Though I could find a cheap and good place for the club the 6 events cost 72 000 Ft. Not to mention the printing of the Enterpress magazine, maintaining the web-domain, embossed club cards and the other costs. No math knowledge is needed to see that the amount paid by the subscribers won't be enough for the Club events and for the magazine...

This is also my fault as I haven't considered this version: what if there are few subscribers?

Now I would like to ask everyone to push at least once the DONATE button considering their financial power. Or you can transfer to the following account:

http://enterpress.news.hu

"Donate" at the bottom

My editorial may be a bit negative. I feel we are a lot of people for a retro-computer club but at the same time few in number. I really have confidence that the year 2018 will be better than this! I ask everyone having any acquaintance that can be interested in the Enterprise to inform them about this opportunity.

Souvenirs and boxed Bricky Prise

Enterprise souvenirs have been made thanks to our friend József Hevesi. So far only mugs are available but we hope for having a wider range of offer soon. A web shop has been planned however it couldn't be started yet. We hope for its starting in the near future as some need does seem to exist.

The boxed version of the popular, original Enterprise game Bricky Prise has been being made. It is going to be published in an exclusive way on tape and 3.5 floppy disk. You will be informed about the details soon!

Those damned articles!

I was looking at the previous issues of the Enterpress edited by myself, we had the same problem at that time! Namely, there are not enough articles for the magazine. This is also the reason for the late coming out of this issue. Now I had to come out with a double issue so we are in time.

I also would like to ask for your help with this!

We look forward to your writings, articles in any topic in connection with the Enterprise!

If you wish to support the Enterpress magazine to come out you can do it here:

http://enterpress.news.hu

"Donate" at the bottom

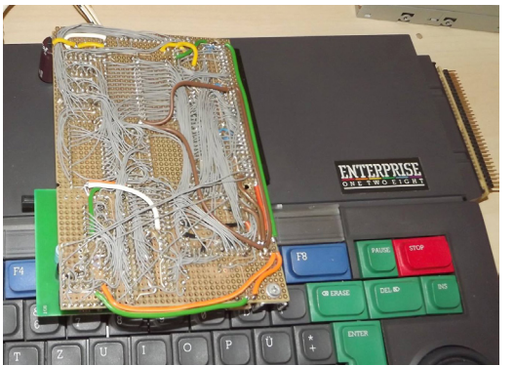
External colour input card

The external colour input card was started with testing the NICK EC3.. EC0 inputs however the imagination (fantasy) of the enterpriseforever.com members ran a little bit over. Of course this is not a problem, and what is more... The original intention of the creators was to connect an electronic device to the Enterprise which is able to display hardware sprites on the screen, with the help of these inputs. However - due to its construction - there are several other possibilities. But the main purpose is to show sprites. The essence of this is the following: a program loads the sprite image into a special memory, then the hardware puts it over the original Enterprise image in the correct position. This position can be modified, so moving the sprite means modifying the coordinates for the card's processor, which is much less work than if it's „copied" (with proper masking) to the image generated by NICK.

On the Enterprise hardware, NICK was designed in the way that the pixel arriving at EC3.. EC0 inputs is allowed to pass through the colouring circuit of the NICK and then the pixel appears on the monitor screen. Nevertheless, this is also possible at the highest resolution because the sprite image from outside can have 16 colours at the maximum resolution, where the original image of the Enterprise could have only up to 2 colours per line.

The original idea was only to display some sprites. But further idea was that not only sprite images could be sent to NICK, you can even get a full screen content from outside. This is significant because such mode(s) can be created with such a resolution that could not be originally displayed by the original NICK (for example the resolution of 640(or more)×288, that NICK can only show in 2 colours, but can be displayed with 16 colours from outside).

Questions and answers:



1. This is good for game developers because sprites do not need to be displayed from a program routine (it is even not always possible) but directly by the hardware. The „precious" CPU time can be used for something else. But that will benefit the players as well because that's how the enhanced effect will appear on their computer screen too. So „this will be good" for all.

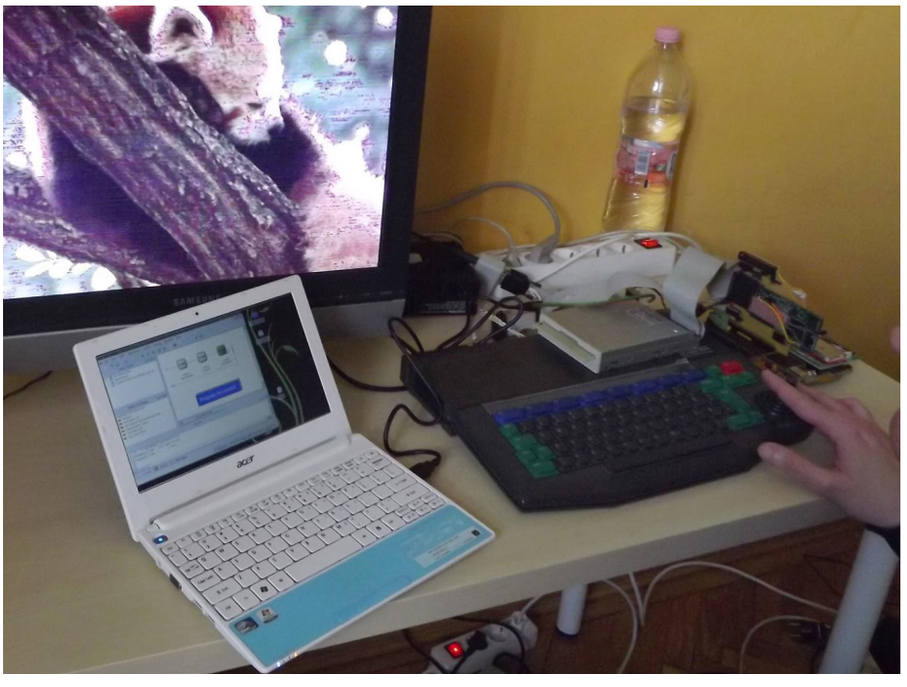
2. There is no video connector on the card, the card itself is the one that generates the data for the EP, and that's what this project is about. So, „the data is sent from the card directly to the EP".

3. Definitely, the card cannot be directly used as an „external video card" or not that way. It can be used as a video card, but the resulting picture will be always limited by the palette and video output of the EP. What is improved over NICK's skills is the number of colours that can be used for higher resolution, and I also have some additional ideas (for example, you can perform super-smooth hardware horizontal scrolling).

This is the current state of the project but this hardware of mine is more complex than my previous hardware of this kind so far, so there is a lot of challenge. I hope the story is now more clear for the Enterprise users.

If you have any questions regarding this subject I will reply them at the „Tesztelés" topic on the enterpriseforever.com forum.

Balagesz



Enterprise accessories

Written by István Matusa (Tutus)

It's 32 years since the Enterprise has been our favourite computer. During so much time both the inside and the outside of the computer could decay. Let's see what we can substitute and what not.

Accessories that can be substituted:

Keyboard membrane: Tamás Kiss (MCU Developments) http://www.mcudevelopments.com/ sells membrane of great quality that can be ordered from his web page or on the Vatera.

Joystick cap: can also be ordered from Tamás Kiss, a great piece, wasn't made by 3D printing, it's cast plastic.

The following accessories can be printed by 3D printer:
- Joystick cap
- back heatsink cover (a question if the material can bear the heat coming from heatsink)
- right side cover
- Cartridge for SD card

Files for printing can be downloaded here:
http://www.thingiverse.com/search?q=Enterprise+128

The rubber soles on the bottom of the computer can be bought both on the page http://www.mouser.com and in Budapest at Conrad specialist's shop. Unbelievable that it's still been produced by the company 3M like at the time when the EP came out.

The inner lying capacitors also got exhausted during the 32 years. Luckily the can also be substituted (http://www.mouser.com)

Accessories that cannot be substituted so far:
- The rubber material around the internal joystick
- The keys
- The rubber inlay between the keyboard and the membrane
- The dark-grey paper that covers the tape out from outside

Enterprise Hardware

I think all of us know the history of the Enterprise. When the Enterprise came out not so much hardware was made however as it is known now there would have been a really great deal of possibilities.
Let's see what could be bought long ago:

- EXDOS card
- Speakeasy
- Mouse
- Printer

Third party expansions -Hungarian developments

'A' Stúdió, even being the representative of Enterprise Computers in Hungary, during all that years they only offered the Spectrum Emulator and the EP-Plus cartridge.

MICROTEAM Ltd. designed an **EXDOS card** with 512 kb RAM expansion and EPROM socket.

The SMD Team designed an **internal 6 MHz turbo charger.**

But was *Gyula Mészáros* who took care of changing the Enterprise Hungarian situation adding what he planned as super add-ons for Enterprise:
- **5 slots bus extension card**
- **1 MB RAM expansion card**
- **6 sockets EPROM card**
- **6 sockets EPROM / SRAM expansion card**
- **Serial Interface Card/ Serial Mouse**
- **Real Time Clock card**

More recently, *Zoltan Németh (Zozo)* designed an **IDE HD control card** for the Enterprise, which works great.

Perhaps the biggest sensation so far is the **SD-Card** reader from *Sárközi Gergely, Sebestyén Pál, and Zoltan Németh (Zozo),* which can be connected to the cartridge port . If you think about it, you can leave aside the EXDOS card, floppy drives, IDE control card and HD... Great development!

Foreign developments

Also in 2015, our Polish friend *Maciej Gruszeczki (Pear)* created the **EterMice card,** which allows the use of a PS / 2 mouse and to connect two external standard joysticks to the machine.

English man *Saint* designed a substitutive **512KB/1MB RAM internal expansion card,** which is a great addition, though unfortunately no longer available.

Maciej Gruszeczki designed an external **EXDOS card** but really (no offense) it was a clone, not the original thing....(joke)

In development

Bruce Tanner is currently working on his **EPNet card,** which we are looking forward to. It can be connected directly to the machine or to the Mészáros Gyula bus extension.

Maciej Gruszeczki (Pear) has designed three external cards that break with the usual busbar extension, which is replaced by a standard connector and the cards can be packed on top of each other. The three cards are: **Flexi Bridge** (this

is connected to the machine), **IDE - Compact Flash card, RAM-Flash-Clock card.** We also look forward to these!

Balagesz plans the external color-sprite card, which he recently brought to the Enterprise Club. At present, this is only a test version, so there is a lot of work to do with it. It is clear that this will be the second biggest hit in EP history (after SD card)! Everyone is looking forward to it!

SzörG (Gergely Sárközi) plans to replace the internal memory expansion card with a **RAM / Flash card** and also plans to create a **turbo card** for Enterprise. An **EP Scart cable** is also planned.

Are they few or many? Some *Gyula Mészáros*'s cards are very hard to find. There was also a design of a new bus extension, which would no longer be an edge connector.

It is always a big question whether it is worth making such a small series of hardware manufacturing? We think the answer is yes!

This is a retro community, and obviously it does not have thousands of members.

What is missing

The **date and clock management** has been solved by Zozo, but there is no more a hardware for it, and this is a big loss (literally not a big card, so it could be designed for another card). It is true that *Gyula Mészáros* designed it, but only 1-2 of them have been found and it is a pity, because it works very well.

Those who still want to use floppy drives, would need a **memory expansion and an EXDOS card** (similar to Microteam's card) which could of course be much more modern.

The **EP Scart connector and cable** can only be obtained from Spain at a fair price and unfortunately its quality is not very good.

Management of the Enterprise memory pages

The EP has a Z80 processor, which allocates memory through a 16 bit bus, this results in only 64 kilobytes. The Z80 has been fused with DAVE, which performs the following tasks:

- 4 x 2 * 6-bit stereo audio hardware base.
- Programmable sound or time base.
- Serving 2 internal and 2 external interrupts.
- Generate high bits of memory addressing.

The EP can handle a 22-bit memory storage corresponding to a 4 megabytes range. Addressing depends on the content of Z80 and DAVE. The bottom 14 bits are of the Z80, and the top 8 are generated by DAVE as follows: It has 4 registers that store 8-bit values. The upper 2 bits of the Z80 address one of the page registers and send the value stored therein.
It is apparent from the figure that an EP 64K/128K and Z80 can address 4 Mega.
However, you should periodically change the contents of the DAVE address register.
This is called pagination.

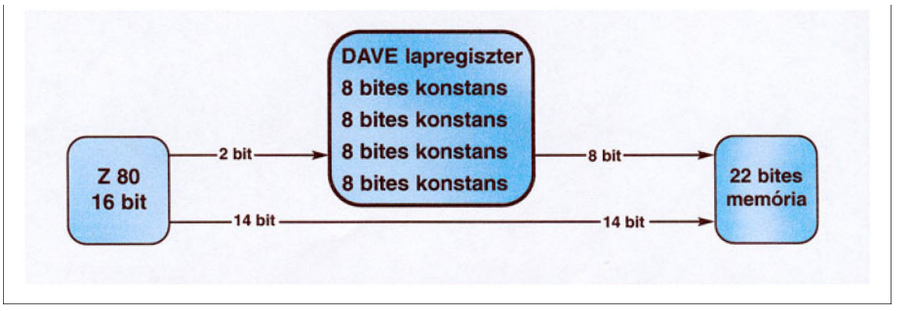
Why we must paginate?

Because the memory exceeds the Z80's manageable 64 kilobytes.
When reading a book we only can see 2 pages at a time.
But you can see the rest by paging.

The EP has 4 pages, each with 14 bits or 16 kilobytes of memory capacity.
On any page, any segment can be entered, so you can apply the same segment to all 4 pages.
The number of segments is 256, of which we use only a few in practice.
Increasing the machine memory can increase their number.

The memory can be of 2 types:
Read-only ROM, and writable/readable RAM.

The ROM memory also retains its contents after turning off the power.



Without it, cold start is unimaginable(impossible?).

ROMs can store not only the resident part of the operating system but also any extensions, programs, and data.

Simplified control systems can even work with ROM-built memory. For more complicated tasks such as screen management, is essential the RAM memory. The EP operating system also requires a minimum of 2 segments of RAM.
What, when, where to paginate?

The stock EP contains the following memory segments:

00 ROM operative system
01 ROM operative system,
Basic routines, WP
04 ROM Basic interpreter
F8 RAM
F9 RAM
FA RAM
FB RAM
FC RAM video RAM
FD RAM video RAM
FE RAM video RAM
FF RAM video RAM

The NICK video chip handles the screen independently of the Z80, and have priority when reading the 64K Nick video RAM. Its addresses are shaped as the FC FD FE FF segments from the Z80 point of view.

In reality, such addresses aren't correct to run a program, therefore once a segment of video RAM is inserted into one of the pages its NICK address has to be converted to Z80 addressing, which is to modify the 2 upper bits of the 16 bit value.

EP cold start and RESET environment:

The NICK register 4 is loaded with 0. These are FIXBIAS, BORDER, LPL and LPH.
The DAVE 16 voice registers, 4 page registers, 1 interrupt register, 3 I / O registers, and 1 memory access control register are also loaded with 0.

The PC register of Z80 is also 0. From this starting point, the zero segment is shown on all four pages.

The Z80 then reads the startup code from the 0000 address on page 0 where the null ROM segment is inserted. Interruption is disabled.

IM register switches to 1 and then skips to the 3rd page where the zero segment is also visible. Paging to Segment 4 will check the „TEST_ROM" startup. If this text is found, the control is passed to 0008. This is how the quick test extensions work.

Running further checks for the hot start mode.

This means that after some initialization, the control is returned to the current running user program.

Running even further the cold start begins. RAM test, and the RAM segments are listed.

At this point, the operating system layout is formed. The zero page has the lowest RAM segment, which is F8 for a 128K stock machine.

Some routines of the operative system are copied to the 0030h-005Ah area on the zero-page. RST 30 will launch EXOS function calls and RST 38 will serve for interruptions.

So, the zero-page should never be patched, except when the interrupts have been disabled and they are needed to be restored!

The content of the first page is virtual, it is the main segment. The second page is the FF RAM segment, that is, the System segment. Here is placed the stack of the operative system, so that during paging you have to make sure that there is no interruption or no operation requiring the stack. On the third page, the zero ROM remains.

The following steps will be:

Enumeration of the Rom segment list.
RAM allocation for the ROM segments.
Enumeration of the peripheral drivers and initialisation.
Initialisation of the ROM extensions, followed by the title screen, that can be disabled.
Then the extensions are scanned with a cold-start action code.
If there is no extension which wants to take control, the Rom segment 1 will accept it, launching the WP application.

There are two types of programs that can run under the operating system: User and System Extensions. Both type of program can create user devices that simplify third-party routines with access tables.

In any case, the EP can only run a single user program. This program can be executed by an EXOS call or an interruption.

Then the operative system will continue calling other extensions and device drivers, following the next that has the higher priority.

Apart from the zero page, you can page segments freely. The system stack must be placed on any address of the third page.

The User can request or release RAM memory through the operating system.
When another User program takes control, the operating system is responsible for initializing memory, devices, and extensions.

There are 2 ways to start an user program, from a file type or from an extension.

File type.

The header begins with 00, 05. This is followed by the two bytes of the loading size. But The system loading routine isn't completely correct. Up to 0100-7FFF it works flawless. But up to 0100 BFFF the memory is not correctly paginated, so the extra data is loaded somewhere and has to be found. I therefore suggest that a program file with 5 header 7FFF should not exceed 7FFF. Additional data can be loaded from another file chunk by the program.

The loader uses a 5-header program to release the user-supplied memory, allocate the memory for loading, and then perform the load.

If a loading error occurs, it sends an error message to the default channel and then calls the extensions with a cold start action code. This initializes the peripherals and extensions by flawless charging, switches the system on by user running and starts the program from 0100.

Another failure is that interrupt is not disabled, even though the user program has not yet built its stack. The first step is LD SP ,? Or the DI command!
For small stack applications, LD SP, 100H is a common practice.

Segments allocated in this case:

0. Zero RAM segment(Lowest RAM segment)
1. RAM-segment containing the program, For shorter programs
Content can be arbitrary.
2. FF System Segment.
3. Zero ROM Segment.

How to start an user application:

In case that a cold-start is needed:

```
LD C, 0
EXOS 0
LD SP ,?
EI
```

Parancs sztringgel indított felhasználói futásnál a felismerés utáni ág.(Not correctly translated)

```
LD C,60H
EXOS 0
LD SP,?
EI
```

Additional steps made by the user application:

- Sets the EXOS variables.
- Opens the vital channels.
- If necessary builds the user interruption.

```
LD A,255
OUT (0B2H),A
LD HL, inter
LD (0BFEDH),HL
```

- If necessary builds a software interruption.

```
LD HL, softint
LD (3DH),HL
```

-If necessary creates a warm reset address.

```
LD A,255
OUT (0B2H),A
LD HL,melegstarl
LD (0BFFRH),HL
```

The user program can not finish running as an extension application.
Exit options:
- Command string transfer.
LD DE, BASIC or WP etc.
EXOS 26
JP ERROR; the string is not recognized by the system
BASIC: DB 5, „BASIC"
WP: DB 2, „WP"
- Call up extensions with a cold start.

```
    LD A, 255
    OUT (0B2H), A
AGAIN: XOR A
    LD (0BF78H), A
    EXOS 26
    JR AGAIN
```

-Cold Reset

```
    LD C, 80H
    EXOS 0
```

An extension routine returns to zero and second page before returning. Be careful about the pagination since:
Page 0 Here is the interrupt entry point.
Page 2 Here is the stack.
Page 3 The routine is running here.
Recommended book:
EXOS 2.1 TECHNICAL INFORMATION

Laszlo Haluska

(Planned No. 1 of Enterpress News 1999, which did not appear)

EXOS Compatible Memory Management - part II.

Written by Zoltán Németh (Zozosoft)

Before going into the details, first a little summary:

We have a 4MB address space, which is divided into 256 16K segments.
Not surprisingly, they are numbered from 0 to 255, 00-FF in Hex.
Basically any one of them may be a RAM or a ROM, or it may be left blank except that the motherboard itself decides other: 00-03 is assigned to the motherboard ROM, and FC-FF is also the motherboard RAM, which has a prominent role as the video memory seen by the Nick chip.
04-07 belongs to the cartridge socket, but here you can place RAM.
In the 128 machines an additional 64K extension board has been installed, which contains F8-FB segments.
What is quite widespread: The MICROTEAM card's 512K expansion occupies the 40-5F area.
EC-FB segments are located on the internal expansion card on a machine with 320k.

It is interesting to note that one of the NASA & GUY demos (I think that one where digital music from Jean Michel Jarre is played) was originally written for this „EC" machine, and I was surprised by the year that the 320K conversion article was not published in Enterpress!.

And in vain, my MICROTEAM card was a 640K machine, but it did not play all the music thanks to the fixed memory programming style... then I rewritten the code using the segments on the MICROTEAM card so I could listen to the whole music.

If you look at RAM, there is a 64K machine with FC-FF. The 128K one has F8-FF.
If we look at the previously mentioned MICROTEAM + EP64, then whe have 40-5F, FC-FF, which is a huge amount, but there is a lack of the F8-FB area that is required by badly coded applications, made by programmers of 128K machines that do not use EXOS Directly.

For the correct programming, forget about these numbers, only FC-FF must be memorized, because of the privileged video memory(and because every Enterprise have that memory range as a minimum).
All you need to know is how many segments you need, the specific number of the segments will be told by EXOS!

Now let's look at the EXOS RAM allocation: two segments have a prominent role, one is FF which is the system segment, and the lowest number of the RAM segments, that is the zero-segment.
Here is the entry point for the EXOS calls or the interrupt program.
And also is where the 5th header programs are loaded from address 100H.

On a 128 machine, it is segment F8. But if, for example, we have a MICROTEAM card, it is 40.

And this is already a problem for many programs (usually bad conversions with a complete modified Spectrum ROM included on the code)...

But on an expanded machine, F8 can be the Zero segment if we are using VENUS. This can also happen with EP-DOS 2.1. But this is a case for an oncoming chapter...

The remaining RAM can be in four groups: system, device, user, free.

If, for example, we will open channels, especially with high demand of video Ram, then EXOS will start expanding downwards and, if the FF segment is over, additional segments may be allocated as system segments.

The total RAM segments required by the various embedded EXOS device drivers are classified in the device category.

A typical example is RAMDISK. The segments occupied by the loaded system extensions are also joined in this category.

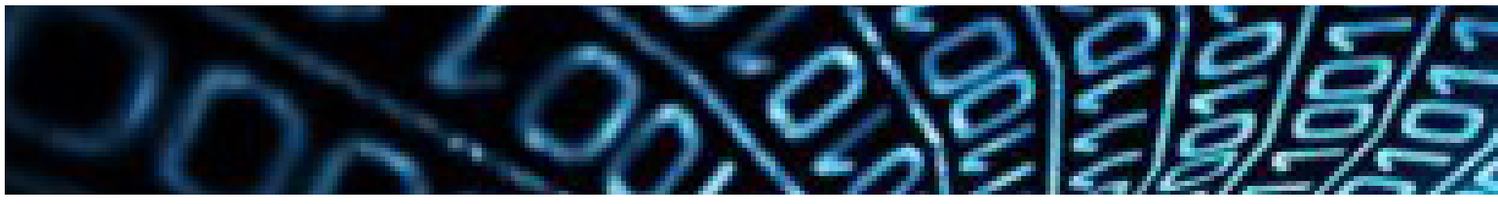
And finally, always there is an active user program, this can be a „new user program" from a system extension or a header 5 files. The segments that they are requesting are seized in the user category.

Which is very important: the user program can only release a user segment, no device nor system segment must be released!

And with this we arrive to a certain faulty method used on Spectrum conversions:

```
1.      LD C,0FAH
2.      EXOS 25
3.      LD A,0FAH
4.      OUT (0B2H),A
5.
```

At first sight, the intent is wise, because not only the segment is paginated, it is first freed.(revise this sentence)
But unfortunately this is a misinterpretation of the EXOS description that was not too correctly translated to Hungarian. Because the EXOS 25 call will be successful only if the segment was previously allocated to us, that is, its USERs.



If FAH is already part of the RAMDISK or if there is a loaded EXOS expansion in it, as it is reserved as a segment for the device, we get an error message so that the segment can not be released. However, the program is still safe to use.

And there is also the case that this segment does not exist in that configuration because it is about an EP64. The right solution here is to ask to allocate a segment from EXOS and use it. If you get an error message because you have run out of free memory, you will have to deal with it in the program mode (eg quit or continue with limited functionality to run the program).

```
1.          EXOS 24
2.          JP NZ,HIBA
3.          LD A,C
4.          OUT (0B2H),A
5.
```

Something more difficult, if you want a specific number of segments: repeat the EXOS 24 call cycle as long as you get the desired segment or run out of memory and then go to the ERROR routine.

But for this method, a slightly modified code should be required in one case: if a video segment is required. Then repeat the call until you get a FC or greater segment.

And you still have to make use of this method if you want to correct an older fixed address program so that the memory used will be properly allocated, as I did yesterday with TUSKER.

This was the original routine from Attus, which is based on SpV method:

```
1.              LD BC,5FAH
2.  SZABAD      PUSH BC
3.              EXOS 25
4.              POP BC
5.              INC C
6.              DJNZ SZABAD
```

This reveals that from FA we need 5 segments, respectively.

Later it is shown that the LPT table was placed at the top of the FF segment.

And then here is the listing:

```
1.  FOGLAL      EXOS 24
2.              JP NZ,HIBA
3.              LD A,C
4.              CP 0F9H
5.              JR NZ,FOGLAL
6.              LD BC,5FAH
7.  EZKELL      PUSH BC
8.              EXOS 24
9.              LD A,C
10.             POP BC
11.             CP C
12.             JP NZ,HIBA
13.             INC C
14.             DJNZ EZKELL
15.             EXOS 24
16.             CP 7FH
17.             JP NZ,HIBA
18.             LD DE,3200
19.             EXOS 23
20.             JP NZ,HIBA
21.             LD L,0F9H
22.  VISSZAAD   LD C,L
23.             EXOS 25
24.             DEC L
25.             JR NZ,VISSZAAD
```

At the beginning, wait until you get the F9 segment. If you run out of memory, you will jump to ERROR.

Then we need 5 segments, which must be FA, FB, FC, FD and FE, otherwise go to ERROR...

We need one more segment, only FF will remain and we will receive the „shared segment" error signal (this is the 7F DTC), otherwise it's an ERROR.

And here's another important aspect: if you are using a shared segment, we need to tell EXOS how long we want to expand, which is done with the EXOS 23 call: the user limit is set. If there is an error signal here, there is not as much free space as we need, so again jump to the ERROR.

If everything is OK, then the code will not be too beautiful, nor too fast, but we will have a way to return the unnecessary segments by a simple method.
That is, trying to release all of them from F9 downwards, so this will only succeed if all the segments have been allocated to us as users. Of course, the completely correct solution is to store the unnecessary segments numbers at the beginning and release them now.

This is needed if our program header 5 no longer fit on the Zero-Page because the additional program segments are being allocated as users and so they would be released by the primitive method, but it should not work. There is no problem with the TUSKER loader, which is good.

This was the procedure on how to correct the fix-segments-use.

Questions and answers:

Povi asked:

Is there no more elegant way to claim a specific segment? Because, in extreme cases, up to 249 segments can be allocated, until we finally get our coveted FC segment. And then the unnecessarily allocated segments will have to be released as well, even if they are needed afterwards.

Answer:

Unfortunately there isn't :-(this is a small flaw in EXOS. It would be good if you could indicate separately that you want a video segment. This is only possible through a driver when you set up a video peripheral type. So it is better to stuck to this method, this is discussed in several books and articles.

Question:

But such programs that are already running out of EXOS 2.0 ... it might be possible to write separate subroutines for EXOS 2.0 or. 2.1. But what if the Super EP is accidentally found?,
what was supposedly hiding Mr. Kopácsy?
(Since then it has been revealed - ed.)

Answer:

Again, these „dunk" programs will not run... So, I think it is better to stay in the regular solution, a few tenths of a second running time may be the worst case, not a big loss on the altar of compatibility.

To be continued!

HiSoft Pascal

Written by Zoltán Povázsay (Povi)

In the first three years of the Enterpress magazine (1990-92), a 12-part series of articles featured the Pascal language from the pen of László Ujlaki. The primary target audience of the publication were Enterprise users who knew well IS-BASIC and were enough „skilled". The speed of BASIC limits the writing of „more serious" programs, and one of the biggest shortcomings of the ZZZIP Basic compiler is that it does not accept floating point numbers.
However, there are few applications on the Enterprise written in Pascal, the ones I know made with Borland Turbo Pascal, running in IS-DOS (CP / M); for example PCBPrise, an electronic PCB design program.

The other Pascal compiler available for the Enterprise is an HiSoft's product. This program is undeniably undervalued, which is partly understandable, as the compiler only implements the standard Pascal, so its features are behind Turbo Pascal's ones: the biggest drawback is the lack of a string type.
However, with my current article, I would like to recommend the use of HiSoft Pascal because although the lack of string type is really annoying, the program still has advantages over Turbo Pascal. What are these?
British HiSoft originally developed the Pascal compiler for Spectrum, but later released it on several Z80-based machines, including CPC and Enterprise. The „most advanced", CP / M version, was the unnoticed HiSoft Pascal80. Despite its Spectrum roots, it is not a Spectrum mere transcript, but a program that maximally (and correctly) uses EXOS and translates the source code into a 5-header executable (Translate command).

Only with this feature, you've already gained one extra point against Turbo Pascal, as this means that you do not need IS-DOS or the compiler to run the compiled programs.

However, it is also surprising that Turbo Pascal - unlike its name says - produces a slower code than HiSoft Pascal. This velocity difference is mainly apparent when calculating floating point numbers, and it is because Turbo Pascal stores the real numbers in 6-bytes while HiSoft does it in only 4 bytes.

Of course, due of the smaller size, the representation of the numbers is less accurate and the numbered range is smaller, but in my opinion, this accuracy is also quite appropriate for us (since we do not want to emulate the first second of the Big Bang) and speed is always the preferable characteristic on this type of machines.

It could have contributed to HiSoft Pascal's unjustly unnoticed role the fact that in the past (in the ,90s) there was no documentation available. Nowadays, with almost anything on the net, when Google can find anything - though I found a number of different HiSoft Pascal versions - the Enterprise version does not seem to exist. But the Enterprise specific commands have been figured out thanks to the partial disassembly of the program.

As mentioned above, the Enterprise version exploits the possibilities of EXOS, but previously there was no way to read about the implementation of a function named EXOS and the associated pre-defined RA, RB, RBC, etc. variables! Also it was not possible to read about the procedures and functions that, although not part of standard Pascal, are available at HiSoft Pascal (eg INCH, MAKESTR, TIN, TOUT etc.). In the forthcoming article series, we will be talking about these undocumented commands, I will show you various tips and tricks that hopefully will persuade more

people to use Pascal!
Although the last official version 1.1 of Hisoft Pascal is a very powerful tool, enhanced version 1.2 from PovySoft contains useful modifications and improvements:

Now the system starts on an 80-character screen mode instead of only 40. Has been corrected a bug that appeared when in a program a row of numbers is followed by 13 spaces. The source code now can be also written in lowercase (for better readability). However, after the input, the commands are stored tokenized in the memory and displayed as capitalized on the program listing. It no more writes spaces after the num-

bers, matching Turbo Pascal. The PAGE procedure has been fixed: CHR (26) is sent to the screen and CHR (12) is sent to the printer. The EXOS calls are now converted to a procedure (see below).
Three new functions and three new procedures have been implemented.

New functions :
Function Swap (x: integer): integer; Replaces the bottom and top bytes of the argument.
Function Hi (x: integer): char; Returns the top byte of the argument.
Function Lo (x: integer): char; Returns the bottom byte of the argument.

New procedures:

Procedures ClrScr; Deletes the screen and places the cursor in the upper left corner.
Procedure GotoXY (x, y: integer); Position the text cursor in the xth column and y in the row of the screen.
Procedure SetVar (ExosVariable: integer; NewValue: integer); Sets the ExosVariable EXOS variable to NewValue.

Example:
SetVar (27, 255); The Border color is changed to white.

Using the EXOS procedure

With the EXOS procedure - however surprising - we can do EXOS calls within Pascal. EXOS calls expect parameters on the A, BC and DE registers, and the results are returned in the same registers. The A-register will returns a Zero status code if the call was successful, otherwise it will return the error value status. HiSoft Pascal has already predefined the RA, RB, RC, RD, and RE char types, and RBC and RDE integer variables: these can be used when calling the EXOS procedure.

Example:

```
program time;
begin
  exos(32);
  write(,Current time is: ,);
  write(ord(rc):2:H, ,:');
  write(ord(rd):2:H, ,:');
  write(ord(re):2:H)
end.
```

Comment:

On previous versions of HiSoft Pascal the Exos command was not a procedure, instead it was implemented as a function, the resulting value stored in A. Since Pascal unfortunately does not allow calling functions, the returned value was not usable somehow, so if we did not need the result, then at least one dummy variable assignment had to be made, for example: tmp: = exos (32) ;.

This has changed on version 1.2, the Exos command has been modified. Now we don't lose information, because the value of register A is automatically added to the built-in RA variable. Therefore, what has been implemented on the EXOS function on the new version is: Now we can Test the RA variable to check whether the EXOS call was successful.

Keyboard and joystick use In HiSoft Pascal:

In the previous issue I've shown how to call Exos Functions from within Pascal, and now let's look at how to read the keyboard and joystick. In the second half of the article, I will show you how to use a machine tool to accelerate our prepared functions a bit. HiSoft Pascal has „stock" implemented function called "inch", which can be used to read the keyboard.

Function inch: char;

The function does not wait for the key to be pressed: if no key is pressed, it returns zero, otherwise it returns the ASCII code of the pressed key.

Example:

```
program TestInCh;
var ch : char;
begin
  writeln(,Press any key...');
  repeat
    ch := inch;
  until ch <> chr(0);
  writeln(,You pressed ,, ch)
end.
```

The inch function is an hybrid solution compared with the more standard ReadKey and KeyPressed functions from Turbo Pascal, so let's see how those functions can be implemented!

HiSoft Pascal uses the KEYBOARD: device, the 105 channel is opened for it and we will use it in our functions:

```
function KeyPressed : boolean;
begin
  ra := chr(105);
  exos(9);
  KeyPressed := rc = chr(0)
end;
function ReadKey : char;
begin
  ra := chr(105);
  exos(5);
  ReadKey := rb
end;
```

Using these functions, we are able to observe the keyboard and even the built-in joystick as the main directions correspond to the following ASCII codes:
Up = chr (176), down = chr (180), left = chr (184), right = chr (188)
However, to read the joystick, a more elaborate method would be a function that behaves like the JOY function of IS-BASIC. It would be possible to read not only the main but also the diagonal directions (for example left-up), and the reading of the external joysticks. Fortunately, the EXOS 11 function does exactly that (too). Let's see its implementation:

```
function ReadJoy(n: integer) : integer;
begin
  ra := chr(105);
  rb := chr(9);
  rc := chr(n);
  exos(11);
  ReadJoy := ord(rc);
end;
```

Since there is no implicit type conversion between integer and char, we are forced to use the chr () and ord () functions. Of course, this is not a problem, as the char type was basically designed to store characters; It was not by accident that the later versions of Pascal introduced the 8-bit bytes and short ones of all types. Due to the lack of these latter types, we can see

the above-mentioned forced solving.
Another interesting thing, which is due to the shortcomings of the HiSoft compiler:
If instead of ra: = chr (105) we write ra: = ‚i' (the lowercase ASCII code 105) , then the compiler produces a faster 7- clock cycles code. However, I do not recommend this solution, because it would be at the expense of readability and maintenance of the source code (not worth 1.75 μs).

The input parameter of the ReadJoy function is the joystick type (0: internal, 1: ext1, 2: ext2), the return value of the read direction:

- 0 = nothing
- 1 = right
- 2 = left
- 4 = down
- 8 = up
- 16 = fire (space).

In the case of a diagonal direction, the sum of the two main directions is the result (for example 5 = right down). The function does not verify the correctness of the argument, it will read the ext2 joystick (EXOS property) for an invalid value. We can therefore see that the behavior of the function is the same as the JOY command of IS-BASIC.

Of course, besides the functions described above, you can also directly read the ports so that you can detect keypresses that do not provide an ASCII code (for example SHIFT, ALT, etc.) on their own. For reading and writing Z80 ports, HiSoft Pascal provides the inp function and the out procedure:

```
function inp(Port: integer) : char;
procedure out(Port: integer; Value: char);
```

Using machine code in HiSoft Pascal:

Unfortunately, in HiSoft Pascal there is no way to write inline assembly inserts (such as on PC Turbo Pascal), but with the INLINE command, you can place machine code routines in the source code.

In order to use the function or process parameters in the machine routine, let's know how the translator stores them.

```
procedure test(i: real; j: integer);
```

In this case, the variable j is located at addresses IX + 2 and IX + 3, and the variable i is located at addresses IX + 4, IX + 5, IX + 6 and IX + 7. The reservation of parameters supplied by value depends on the size of the type (so it takes 4 bytes for our real-type i variable).

The situation is different with reference parameters:

```
procedure test(i: integer; var x: real);
```

In this case, the value of x is not the value (which occupies 4 bytes), but only the memory address that occupies two bytes. So the address of variable x is given by IX + 2 and IX + 3, and the value i is given at addresses IX + 4 and IX + 5.

For functions, the return value will be send back to the memory address above the input parameters:

```
function test(p, q: integer): real;
```

Thus, in this case, the value of q is at IX + 2 and IX + 3, the value of p is located at addresses IX + 4 and IX + 5, and the return value of the function will be stored at IX + 6, IX + 7, IX + 8 and IX + 9.

After that, let's see how to write the KeyPressed, ReadKey, and ReadJoy functions as a machine-code readable routine:

```
Function KeyPressed: boolean;
Function ReadKey: char;
```

The above two functions are „simple case": they do not have input parameters, and the type of return values of both functions is byte:

So we have to place the result at the address IX + 2. For a boolean type implemented in Pascal, you should know that „false" is stored as 0, and „true" is stored as 1 (unlike, for example, with C, where all non-zero values are true) , You will have to pay attention to this when writing your routine.

Let's see the implementation:

```
function KeyPressed : boolean;
begin
  inline(#3E,#69)          {LD A,105};
  inline(#F7,#09)          {EXOS 9};
  inline(#79)              {LD A,C};
  inline(#ED,#44)          {NEG};
  inline(#3C)              {INC A};
  inline(#DD,#77,#02)      {LD (IX+2),A};
end;


function ReadKey : char;
begin
  inline(#3E,#69)          {LD A,105};
  inline(#F7,#05)          {EXOS 5};
  inline(#DD,#70,#02)      {LD (IX+2),B};
end;
```

Let's read the ReadJoy function:

```
function ReadJoy(n: integer) : integer;
```

The n argument of the function is obtained at addresses IX + 2 and IX + 3, in the low-endian form (ie, the lower byte will be at the lower address, ie IX + 2).
The result of the function should be placed on the address IX + 4 and IX + 5, also in a small-endian form. Since the result of the EXOS 11 function call is only one byte, we need to place 0 in address IX + 5. In the example below, let's take advantage of the fact that the value of register A after the call is always 0, so we write it to IX + 5:

```
function ReadJoy(n: integer) : integer;
begin
  inline(#3E,#69)          {LD A,105};
  inline(#06,#09)          {LD B,9};
  inline(#DD,#4E,#02)      {LD C,(IX+2)};
  inline(#F7,#0B)          {EXOS 11};
  inline(#DD,#71,#04)      {LD (IX+4),C};
  inline(#DD,#77,#05)      {LD (IX+5),A};
end;
```

The following sections of this article series, will be about the use of VIDEO: pages, graphics, and file management.

(to be continued...)

Games of the past

Written by:
László Kiss (Lacika)

From the 1960s until the mid 70s mainframe computers overtook the place of second generation computers of the 50s in more and more American universities and research centres.

These third generation computers were using integrated circuits, had operating systems and were able to perform multiple tasks at once. Connecting these kinds of machines was made possible by the ancestor of today's internet, ARPANET. This on one hand had economical reasons (sharing system resources), but also military purpose: if any component of a decentralized network is destroyed by bombing for example the network itself remains operational. But don't think that these computers were only used for running super serious software! Many genre-creating games were designed on them mostly by students of course. Many of these games got lost due to the neglectful behaviour of students or the strict teachers. Many others on the other hand spread from college to college on the network.

I bet you wouldn't even think that some of these software run on Enterprise too in its original or almost original form. Let's take a look at a couple of them!

One of the earliest games designed for mainframe computers is from 1969 and called Sumer. This one later becomes popular as Hamurabi. In this game we play the role of an emperor, therefore buying/selling land on a yearly basis, deciding what and how much to plant and feeding the population. If we are successful our people won't starve and more people will arrive to our city. The program runs on Altair BASIC and on Microsoft BASIC by using IS-DOS.

The game is the ancestor of empire-building games. Many people added features like „taxing" and „building" after the original release. The expanded 1984 Spectrum version titled Feudal Overlord features some character graphics too and runs on Enterprise by using Geco's software emulator.

Another software called Lunar Lander was written by high school student Jim Storer who had been inspired by the moon landing of Apollo-11 on July 20, 1969. The game was originally written in FOCAL language. It challenged the player to land successfully on the Moon. The software informed the player about speed, altitude and fuel level in writing. Then the player typed in how much fuel they wished to use in the next time period. The BASIC version was later created by David Ahl, which many others have modified in many ways since then. We can try out one of these modified versions on Altair BASIC.

The Star Trek universe was also something that inspired people to create computer games. The first TV Series was made between 1966 and 1969 so it was well before the first movie when they made a game out of it. Don Daglow's version in 1972 mixed simulation with a kind of storytelling. Later many more versions were made and we can try some of them on Microsoft BASIC. One of them is from 1971 and its creator is Bill Peterson.

One of the first RTS games was probably Civilwar, which takes the player back to the bloody events of the American Civil War. This software had a very basic gameplay and could be played by two players against each other. After splitting the available resources the players only needed to care about the strategy part. The original one was created in 1968, but the version compatible with Microsoft BASIC is from 1973.

Gregory Yob's 1972 game called Wumpus (or Hunt the Wumpus) can be taken as an early version of text based role playing games. In this software - which we can hardly call a game at all by today's standard - we have to hunt for a monster called Wumpus in a dodecahedron shaped labyrinth. There are three doors opening from every room; the software identifies the rooms by numbers. After the computer informs us about the room we are in it asks the question: „shoot or move?" We have to answer by pressing „S" or „M" and the number of one of the doors. We have to be careful, because if we step into the room of the monster, it wakes up and that means harm to us. Two of the rooms have holes in them and if we fall the game ends. Other two rooms are inhabited by giant bats that grab and fly us to a randomly chosen room (perhaps one with a hole). If there is possible danger behind one of the doors the computer warns us, but it doesn't tell you which room is it. If we

Arpanet

figure out where the Wumpus is we have to shoot into that room, but in case we are wrong the arrow can bounce back and hit us. The game runs on Altair BASIC and Microsoft BASIC too.

One of the programmers constructing the ARPANET network was William Crowther. He and his wife Patricia, who was also a programmer, had a mutual hobby: speleology. Crowther's other passion was D&D (Dungeons and Dragons) role playing games that started their conquest in 1974. When the couple's marriage went sour William decided to write a game software for his daughter which combines his two passions. The software created in 1975-1976 was simply called Adventure, but became also known as Colossal Cave Adventure. The program gave a written description about place and events going on around the player. (The CRT monitors only became widely spread from the mid 70s, thus earlier games obviously provided the players with much less information via printers.) Its most interesting feature was that the player could control the game by typing in verb+noun commands using their own words. The goal of the game was not defeating opponents, but exploring a dangerous cave and finding the hidden treasures.

After a time Crowther got bored with developing the program and left the half-done game on his computer. The software then moved from institution to institution until Don Woods discovered it in the MI laboratory of Stanford University. He liked it so much he decided to continue writing it. He contacted Crowther, who also approved Woods' ideas. The Tolkien fanatic Woods expanded the game with motifs of his favorite novels (The Hobbit, Lord of the Rings), thus the game was soon swarming with trolls, dwarves, dragons and a giant volcano also showed up. Descriptions of places became more realistic and readable; players could feel like they were in an interactive novel. Adventure started a new genre aptly called role playing game. IBM picked up on the opportunity and soon every PC came with a version of Adventure besides the MS-DOS 1.0 operating system. In 1981 the game got officially released being named The Original Adventure. Versions compatible with CP/M operating system were created by Mike Gotez between 1980 and 1982. We are rightly using plural form here, because by then the game had multiple (extended) versions. Different versions can be distinguished from each other by the maximum of achievable point score after entering the SCORE command. Scoring points is possible by finding treasures and taking them back to the house (where the game starts). We can only carry a limited number of items at once, but left behind items stay where we leave them (DROP command). At the end of the game (including the possibility of dying) the game evaluates our performance by calculating our total score. Our final goal of course is achieving the „Adventurer Grandmaster" title by reaching the maximum point score possible. The game was later released for Spectrum, C64, CPC, MSX and Enterprise by Level 9 Computing with the same minimalist design, but with the name changed to Colossal Adventure. This game along with Adventure Quest and Dungeon Adventure constitutes the Middle Earth trilogy.

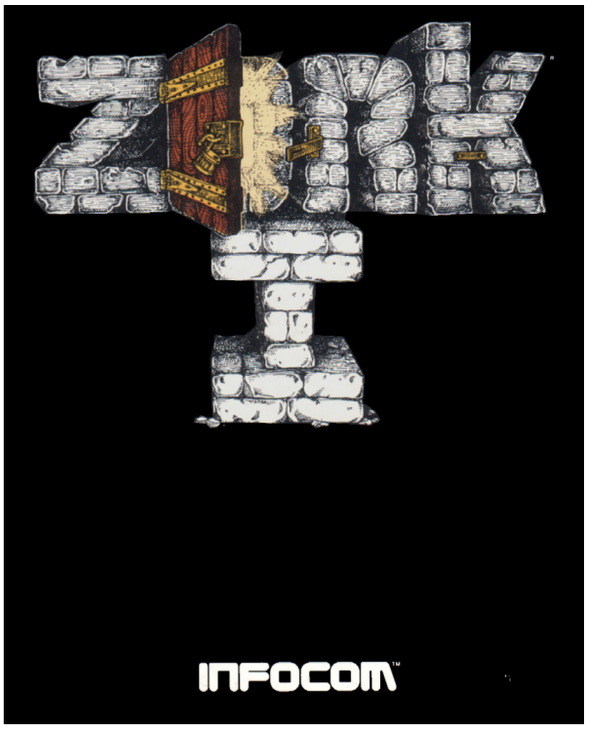
The first role playing game available for public sale however wasn't the Adventure, but Adventureland by Scott Adams from Florida. The game had been written in BASIC language and got released in 1978 for TRS-80 and Apple II computers. Although the story, design and the game's parsing ability was primitive, it was unique at the time and therefore became successful anyhow. After the success Scott Adams and his wife Alexis founded their own company called Adventure International. Adventureland was then followed by twelve other role playing games in future years. Nearly all of them became classics.

The game Adventureland worked with the same recipe as Adventure: our adventurer character had to collect thirteen mythological treasures (Blue Ox, Jeweled Fruit, Pot of Rubies, Diamond Ring, Diamond Bracelet, Magic Mirror, Gold Crown, Thick Persian Rug, Firestone, Golden Net, Golden Fish, Royal Honey, and Dragon Eggs) and thus achieving the maximum of 100 points. In

order to do so he had to tackle logic games, use items, trick other characters. Fighting isn't a main feature in this game either. Of course just like in Adventure it is impossible to carry all of the treasures at once, so once again we need to drop each of them off at one location.

After microcomputers came into general use Adams' games got rewritten in much better quality (not BASIC) by the creator of Mysterious Adventures series Brian Howarth. Some got graphics too. The Spectrum compatible versions of these run well on Geco's software emulator: Pirate Island, Secret Mission, Voodoo Castle, Savage Island 1, Savage Island 2, The Sorcerer of Claymorgue Castle.

One of the first video game companies was founded in June 1979 by college students Dave Lebling, Marc Blank, Joel Berez and their professor Albert Vezza. They named it Infocom. Thanks to the new company more and more people learned about „interactive fiction" which others call text based role playing game. They started developing their first game in 1977 using the PDP-10 mainframe computer of MIT university (home computers didn't even exist back then). This adventure taking place in The Great Underground Empire turned out to be so long they released it in three parts. Zork 1: The Great Underground Adventure was first in 1980, then came Zork 2: The Wizard of Frobozz in 1981 and the final part of the trilogy was Zork 3: The Dungeon Master in 1982.

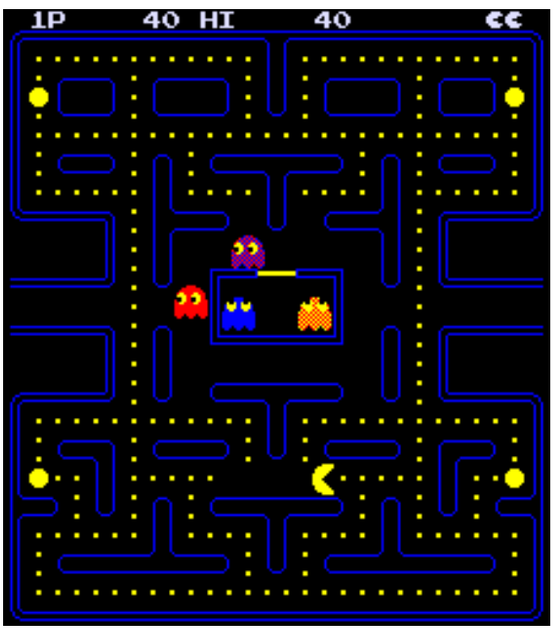


The company had been advertising the concept of „a thousand words tell more than a picture" since foundation and they were very vocal about this. In their opinion wasting the limited memory of computers on images would have been superfluous (of course at the time it was also virtually impossible), because due to the literature quality of texts and a good parser places and characters are brought to life by the player's imagination. Of course they meant every word they said. Mark Blank and Dave Lebling when writing Zork were aiming to create the best parser ever. According to them „a role playing game is only as good as its parser". They achieved their goal. The parser of Infocom games was famously smarter than that of any other RPG at the time or even later. It's demoralizing even for the biggest fans of the genre if one has to waste time guessing what phrase is accepted by the software. The games of Infocom therefore didn't stand in the way of the fans' creativity.

Text based games made an interesting programming innovation possible: Joel Berez and Marc Blank created a virtual machine called Z-machine that enabled easier compatibility with different platforms. All the games of the company were written in Z-code and therefore developers could make them compatible with any platform just by writing a Z-machine for that platform. Thanks to this Infocom games could quickly be released for 8-bit computers coming into general use in the early 80s. CP/M compatible version also exists which is good news for Enterprise owners too.

The Zork trilogy cannot deny that it was inspired by Adventure: the setting for our adventures is The Great Underground Empire which we can enter through a trapdoor found in a remote little house by the woods. We can control our actions by the traditional method. However, this game understands more than simple verb+noun collocations and can be commanded by using more complicated sentences (eg.: PUT TORCH AND DIAMOND IN BASKET) too.

To see some animation on the screen we have to change genre and play with one of the most famous arcade games, PacMan. Although there are many PacMan clones, it's worth to check out the



PacMan emulator for SAM Coupé and Spectrum written by Simon Owen in 2012. The SAM Coupé version was converted to CPC and Geco Ep by SyX. The word „emulator" is used in the strict sense here as the software has been made by using the original arcade ROM (the PacMan arcade machines were using Z80 CPU too)! This means that the game is the exact copy of the arcade version. The colors, figures, sound and gameplay are all providing the same experience as the original.

Boot „sectorology”

Written by
Zoltán Németh
(Zozosoft)

The EXDOS boot sector looks like this:

00-02 EBH, FEH, 90H IBM (jump to the boot code)
03-10 „EXDOS 1.0” system (format software)
11-12 512 byte / sector
13 sector/cluster
14-15 number of boot sectors (usually 1)
16 number of FAT copies (usually 2)
17-18 number of main directory entries
19-20 number of disk sectors
21 formatting type (F8h-FFh)
22-23 copy of FAT number of sectors
24-25 sectors / track
26-27 number of heads
28-29 number of hidden sectors
From here only present on EXDOS (or VT-DOS) disks:
30 C9H RET, entry point for the MSX-DOS BOOT program
31-63 0 null filled
64-69 „VOL_ID” identifying string
70 0 UNDEL signaling byte
71-74 32 bits Disk ID
75-99 0 null filled
100-511 E5H empty space

The 03-29 range is in line with the FAT standard, but now I would be interested in scraping this: At 00-02 there is an x86 jump instruction that EXDOS does not need, but for compatibility reasons it does because some Microsoft operating systems - unlike their own standard, check their existence. This fails with Ataris because the early versions of TOS didn't write this, so PCs did not read the Atari formatted disks.

At 30, MSX owners thought that in case of an EP disc forgotten in a random drive, the boot experiment would not cause a start, so a RET command was put in this entry point. 64-74 bytes are EXDOS's own extras: In 64-69 bytes the „VOL_ID” identifies that this is an EP disk and contains the following 2 data. At 70, a flag byte indicates that the disc contains deleted but recoverable files. 71-74 bytes have a random 32-bit disk identifier, which is useful for disk exchange control.

In more than 25 years since the creation of EXDOS, new standards have added new data to the boot sector:

28-31 number of hidden sectors (32 bit)
32-35 number of disk sectors (32 bit)
36 physical drive number (PC) (00h floppy, 80h HDD)
37 CHKDSK mark bytes(Windows NT based systems)
38 expanded boot sector flag (29h) this indicates that the following parameters exist
39-42 32 bits Disk ID
43-53 11 characters disk name
54-61 7 character file system ID „FAT12 „ or „FAT16 „
510-511 55h,AAh boot sector mark

These novelties will also be incorporated into EP formatting programs, in the interest of better compatibility with today's systems. Fortunately, they are all implemented in the empty space left by EXDOS, so it's only necessary to extend the existing EP boot sector. Only RET instructions for MSX are a threat to expansion, but they are only for Vista, with more than 65535 partitions, which is perhaps a viable problem.

As you can see just a few years after EXDOS, Microsoft has also incorporated the 32-bit disk ID. We can also do it by copying the 32-bit ID used by EXDOS to its PC location as well as the 29h for 38 bytes indicate whether if such a value is present. With the File System Identification adjusted to „FAT12”.

To place the disc name in the boot sector would actually be a smart idea instead to search for it in the main directory. However, I tried both MSDOS 6.2 and XP and, regardless of whether you are using or not the disk name in the main directory, If it is not present, the name copy in the boot sector will not be used.. So with a relaxed heart, we can ignore this field. Then, when using the VOL command, the name will be entered there too. Lastly, place the 55h, AAh bytes at the end of the boot sector. It is not completely necessary, but who knows what will check other more lenient PC systems (as it was already a fact with the jumping instruction or the formatting program name).

And there is one more thing at the beginning of the disc EBh. FEh, 90h is a self-leaping JMP instruction on a PC, which means that if you burn an EP disk in your

drive and boot your PC, it will freeze well ... ultimately, this solution will not harm the computer, only annoying when you reset. There is a simple solution that does not waste much space: after the EP disc ID, for example, 75-76 addresses must be written with CDh and 19h, and then the jumping instruction is changed to EBh, 49h, 90h at the beginning of the disc. The new instruction which has been embedded is an INT 19h that restarts the boot, so it reads the floppy again and again, but when we notice the disc forgotten inside and take it out, the computer start-ups normally, it does not have to be re-seted. If there isn't problem with wasting more bytes in our EP program, it is also possible to use a longer code to disregard the disk, filling the boot sector (also 4Bh):

```
db 31h,0C0h         ;xor   ax, ax
db 8Eh,0D8h         ;mov   ds, ax
db 8Eh,0C0h         ;mov   es, ax
db 0FCh             ;cld
db 0B9h             ;mov   cx,
dw 100h             ;      100h
db 0BEh             ;mov   si,
dw  7C00h           ;      7C00h
db 0BFh             ;mov   di,
dw  8000h           ;      8000h
db 0F3h,0A5h        ;rep movsw
db 0EAh             ;jmp   far ptr 800h:62h
dw  62h
dw 800h
db 0B8h             ;mov   ax,
dw 201h             ;      201h
db 0BBh             ;mov   bx,
dw  7C00h           ;      7C00h
db 0BAh             ;mov   dx,
dw  80h             ;      80h
db 0B9h             ;mov   cx,
dw   1              ;      1
db 0CDh,13h         ;int   13h
                    ; DISK - READ SECTORS
                      INTO MEMORY
db  72h,5           ;jb   +5
db 0EAh             ;jmp   far ptr   0:7C00h
dw 7C00h
dw   0
db 0CDh,19h         ;int   19h   ; DISK BOOT
```

The year 2012 EXDOS 1.4 and EPDOS 1.7, and FAFO 2.5 already write modernized boot sectors when formatting.

F I L E

We have not used this small system extension from HSOFT for a long time but, it is a very practical program! Probably this happened in part, because its excessively concise documentation did not reveal how it could be used, for example in BASIC. Its original function was to call back the name of the selected file name from a program, that is, from any program you can choose comfortably, for example, a file to open. Later, ZozoSoft has developed it into truly universal, so it has become easy to use it as a START menu program.

After you call it, a file selection menu appears on the screen with the contents of the current drive. Handling is absolutely straightforward: You can move between the files and subdirectories with the up and down positions of the built-in joystick. SHIFT + up / down - scroll, ALT + up / down - skip to top / end of list, ENTER or SPACE: Select file. (The subdirectories are also managed by the program.) We can change them by pressing STOP. (Not selecting a file).? You can switch the drive by pressing the desired drive letter. The program originally only handled A-E drives, but if running on newer versions of EXDOS it will ask for existing drives.

After the program is invoked - originally - it places the name of the selected file (and path - line) on a 256-byte buffer at page 0. The program should be called in this form as follows:

```
FILE word$(buffer)path /options
```

In the path you can specify a filter name (for example * .com). The options can be: H - display hidden files, S - display system files.
Since this is the most elaborate solution for file selection (absolutely foolproof and user-friendly), it is also worth using it in BASIC as an example:

```
100 ALLOCATE 256
110 CODE A=" "
120 LET C$="FILE "&CHR$(MOD(A,256))&CHR$(INT(A/256))
130 EXT C$
140 LET L=PEEK(A)
150 LET N$=""
160 FOR I=1 TO L
170   LET N$=N$&CHR$(PEEK(A+I))
180 NEXT
190 PRINT "file="""&N$&""""
```

The file name of the selected file is sent to the N$ variable. For example, if you want to show only .com files, the 120 line has to be modified as follows:
120 LET C$="FILE „&CHR$(MOD(A,256))&CHR$(INT(A/256))&"*.COM"

If we want to be absolutely precise, we have to call the extension within the control of an interruption handler(HANDLER), if not the system will not generate an error?, then you can use traditional INPUTs. It is important to know, that if you press STOP back from the expansion, It also interrupts the BASIC program, so It should also be taken care of properly.

The original function of the program was supplemented by ZozoSoft: When called without parameters, then it launches the program (EXOS module loading function).
If it is not the case, You can try the EPDOS START command, so you can start BASIC programs(if there is EPDOS in the system).
The File program launcher has a separate bootable START program that can be put on disks. Compared with the former program provider it is EXOS compatible (Also on EP64), and handles HDD and SD.
The control was enhanced by István adding EnterMice mouse handling. The scroll wheel also works, left button select, right button CD .. (moves up one directory).

SIDBASIC

Written by
Persa Noel
(Geco)

SIDBASIC , the most recently version of the SID player, now reproduces converted SID files on a stock Enterprise 128 without the need of any hardware expansion.

The software is EXOS compatible, if the machine has been installed with a storage medium (disk controller, SD card), the M64 files can be selected from a file manager, but still the cassette-only configuration can load the selected M64 file from the tape deck.

Playback on a 128KB machine is allowed by unpacking 8Kb blocks during playback.

Limitations: Maximum Loadable M64 File Size is of 24320 bytes (5f00h), No SYNC, Overhead, and Low Pass Filter Emulation, also SID digital effects can not be played.

Combined waveforms do not work properly due to real hardware differences, in this case, simply noise> saw>triangle>rectangle conversion takes preference.

Control

1 - Play music on Dave chip
2 - Music playback on an external 8-bit DAC card if connected to the machine.
3 - raster bar on (works only at 50Hz speed)
4 - raster bar off
Esc - file selection

Some information about the M64 files

Description of the format: uses 16384 bytes each block, each block stores 655 bytes of one of the 25 SID registers, the first are the 655 bytes of register 0, the next 655 bytes for the register 1, and so on. The last 9 bytes are usually 0, except:

- in the last block, the number of interruptions actually used is 3FFE-3FFF, which can be less than 655 - in the first block with 3FFD the breaking frequency - 50 Hz can be found (so 50-305 Hz is possible), 3FFB -3FFC is the exact number of SID cycles between interruptions at CIA timing (CIA counter value + 1). For video interrupt, it is 0 and the default is 312 * 63 (PAL 50 Hz) or 262 * 65 (NTSC 60 Hz). The Sid.com command only takes note of the rounded Hz value.

The Initially unused bit 7 of channel 3 register (PWM top 4 bit) has a special function: it indicates that the envelope curve must be restarted by the GATE bit during a 1-> 0-> 1 transition period.

Thanks to **István Varga (IstvanV)** for his ideas, and for the following routines and applications without which SIDBASIC would not have been possible:

Routines

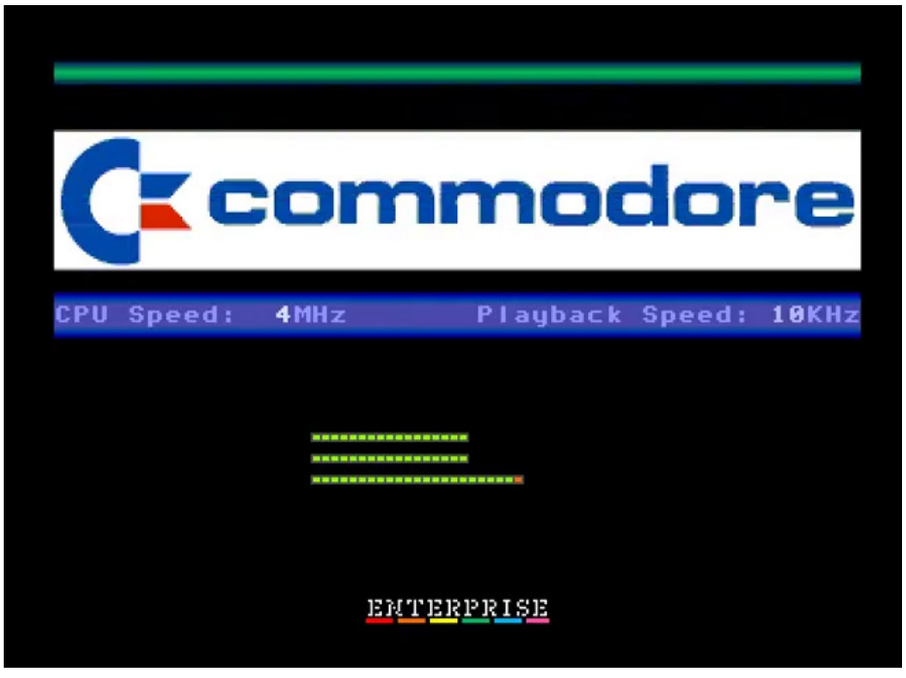
The 8 KB block unpacking routine interrupted by the Digi player routine is ideal for generating SID voice tags.

Applications

Epcompress - Enterprise packing program
Epimgconv - Enterprise Image Converter Program
Sid_dump - „raw" SID registry data recovery program from SID files
Sid_conv - „raw" SID registry data conversion program into M64 format

Use

sid_dump.exe INFILE OUTFILE [LENGTH1 [LENGTH2...]]
sid_dump.exe INFILE OUTFILE [SONGLENGTHS FILENAME]



sid_conv INFILE OUTFILE [INTFREQ [BLKSIZE [NOADSRBUG]]] INTFREQ (default: -1):

INTFREQ(Interrupt frequency), if the value is <=0 then it is read from input file, otherwise the value (Hz) is used given in command line. In case of -2 doubles the frequency of envelope emulation, which increase the size of output file with about 40-70%.

BLKSIZE (power of 2 in range 256-16384, default: 8192): size of a block of compression, size of dictionary is double of block size (-blocksize BLKSIZE -maxoffs BLKSIZE*2). 0 or negative value also sets the default 8K.

NOADSRBUG (0 or 1 ,default: 0): if this value is not 0 then it resets the counter of SID envelope emulation 15 bit timer at rising edge of GATE bit. Real Hardware works differently, so in this case the conversion can be vorse also.

The sid_dump command runs a PSID file in a minimum emulated 6502 environment, and its output is a simple „raw" format storing values written to SID registers at 50 Hz (or other frequency) interruptions.

Such a file can be played back with sid.com (below) if it has a SID card (that is, currently only on an emulator :)). PSID -> RAW inverting is also possible with SIDPLAY and sidrecn. lua scripts if sid_dump does not work for some reason.

Examples of SID music can be downloaded here:

http://www.hvsc.c64.org/#download

Example conversion

(Epcompress only needs sid.com, sid_conv.exe also accepts uncompressed format):

..\sidconv\sid_dump.exe MUSICIANS/T/Tel_Jeroen/Cybernoid_II.sid cybnoid2.raw DOCUMENTS/Songlengths.txt MUSICIANS/T/Tel_Jeroen/Cybernoid_II.sid
Name: Cybernoid II
Author: Jeroen Tel
Released: 1988 Hewson
Video standard: PAL
SID model: MOS6581
IRQ frequency: 50.12 Hz
Done converting track 1: 17343 frames
IRQ frequency: 50.12 Hz
Done converting track 2: 451 frames

(The next step is to play SID.COM) epcompress -raw -m0 -9 -blocksize 16384 -maxoffs 32768 cybnoid2.raw cybnoid2.raw

Compressing data
100%

* sid.com (3.37 kB. 106x13 - viewed 2 times.)
* sid.s (10.09 kB - downloaded 1 times.)
* decompress_m0_16K.s (10.27 kB - downloaded 1 times.)
* file.s (31.21 kB - downloaded 1 times.)
* mouse.s (1.47 kB - downloaded 1 times.)

The next step is to create a format supported by SIDBASIC.COM and playable without SID card, which is possible with sid_conv. Description of this format:

- at the beginning there is a 16-byte EXOS header: 00h, 4Fh, compressed data size L, compressed data size H, IRQ frequency L, IRQ frequency H, number of interruptions (24 bit, bottom 8 bit first), 7 unused (Min-dig 0) bytes
- the EXOS header follows the compressed data epcompress -raw -m2 -blocksize 8192 -maxoffs 16384 format
- each 8K block contains up to 682 interrupt-length recordings per channel with 4 converted „registers" per channel. Organizing the data within the block is similar to the RAW format. At the end of the last block (1FFE-1FFF) 682 - the number of interruptions actually used in the block
- channel registers 0 and 1 in 16 bits SID frequency, 4 bits to the right waveform waveform
- register 2 is the ring modulation (bit 7), waveform (bit 5-6, 00 = triangle, 01 = saw, 10 = square, 11 = noise), and current volume (0-4 bit) included. For 0 volumes, the waveform is always square
- register 3 is the quadrature fill factor (top 8 bit), only for non-zero square squelch, otherwise 0

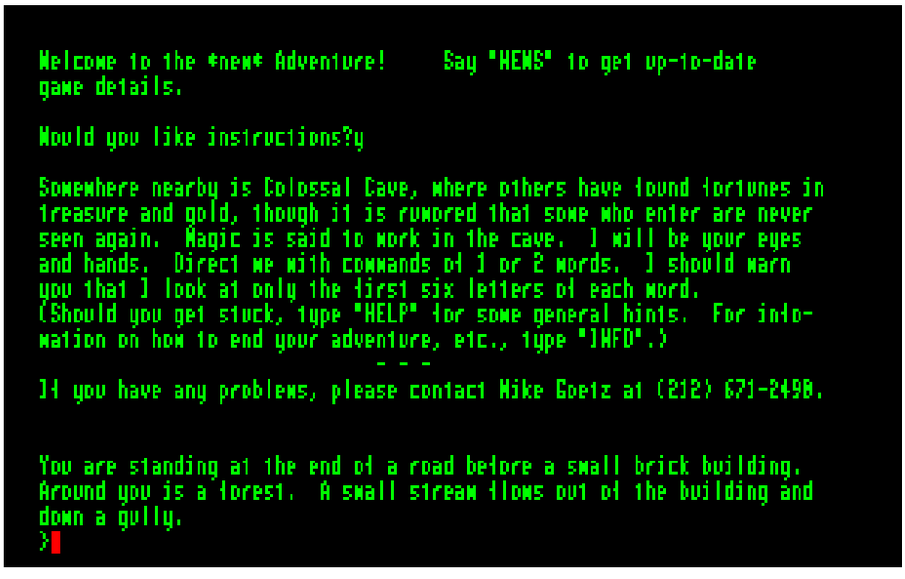
Conversion example

.. \ sidconv \ sid_conv.exe cybnoid2.raw cybnoid2.m64

Converting file...
100%

Compressing date
100%

ADVENTURE



Considered an Internet ancestor, in 1969 was born the ARPANET (Advanced Research Projects Agency) network (Originally a military computer network linking US universities and research institutes) by a group of building up programmers, included among them William Crowther. With his wife, Patrice, who was also a programmer, most of their free time was devoted to speleology. Crowther's other passion was D & D (Dungeons and Dragons) the role games, which started their conquest saga in 1974. When Crowther's private life became on a crisis, William decided to write to her daughter a game program that combined his two passions. The program, produced in 1975-1976, received the naive simple title of Adventure, but it is also known as Colossal Cave Adventure. The program provided a narrative description of the scene and the events surrounding the player. The special feature is that the player guide the gameplay with commands consisting of verb and noun that imitate the natural language. The aim of the game is not to defeat opponents, but to map a dangerous cave, finding hidden treasures. With time Crowther grew bored of the program development and left it semi-finished on his computer, which then moved from institute to institute. Don Woods, at Stanford University's AI(artificial intelligence) Laboratory, found the game and was so pleased with it that he was determined to continue his development. He contacted Crowther who approved Woods' ideas. Being Woods a fan of Tolkien novels (The Hobbit, The Lord of the Rings), then the trolls, the dwarves, the dragons, and the huge volcano appeared on the enhanced game. The sites were more vivid, with more readable descriptions, the player felt immersed in an interactive novel. Adventure has created a new genre that has been named adventure games. Then IBM caught the program and soon released an Adventure version for each PC inside MS-DOS 1.0. The game was officially released in 1981 as The Original Adventure. The version for the CP / M operating system was written by Mike Goetz between 1980 and 1982. Multiplayer was justified because by then existed - due to the countless rapes - several (extended) versions of the game. The game was later released by Level 9 Computing for Spectrum, C64, CPC, MSX and Enterprise, in the same „minimalist" version, Colossal Adventure, which makes Adventure Quest and Dungeon Adventure a Middle Earth trilogy.

There are four versions available in the downloadable package:

A02 - 350 points can be collected in this version. This version also has a FORTRAN source code.
B00 - 550 points can be collected in this version. The labyrinth is nearly twice as big and has new monsters.
B01 - The same 550-bit version as the B00, but it can also be run on Z80 and 8080 processors.
B02 - With some new locations, this version has 580 points.

Commands for managing the game:

SAVE - writes the actual gameplay and exits the program. There is no way to enter a name, only a saved game per disc.
RESTORE - loads a previously saved game.
QUIT - exits the program.
INFO - gives a brief description of the program management (versions B ... only).
NEWS version information (only in B ... versions).
SCORE - checks our current score.
INVENTORY - gives a list of the objects carried by the player (we can not get any weight!).
HELP - returns a couple of tips for playing.
FAST - we get only a solid, concise description of the sites (only in B ... versions).
BRIEF - a full description of a given location will only be obtained when we first go there (version B ... only).

FULL - we always get a full description of the location (only in B ... versions).
LOOK - we will get the description of the site again.

Of course, we get points for finding the treasures, or if we bring them back to the house (before the adventure begins) - since at the same time there are as many objects as we can.?. Objects released with the DROP command will remain in the given location and can be taken later. At the end of the game (including the chance of dying), the program evaluates our performance as a function of our score. Of course, our main aim is to achieve the „Adventurer Grandmaster" title by reaching the maximum score. The most important tip for the game is to run a RAMDISK on a computer with memory expansion, so avoiding constant disk scanning runs much faster! Of course, before you turn-off your computer, do not forget to copy the saved game to a „real" disk.

GraCha

- or the graphic character editor for the EP

I have created games for the EP many times, mostly using graphics-pixel mode. So I thought that I should try something to bring out from the EP the underused modes I call Graphic-Character. There are several other modes, unfortunately, virtually only one of them is usable, in which we have 2x4 colours in one character, on a Hires 16 resolution mode. Some character codes use the first 4 colours, while others use the second 4.(The 2 colours per character from a total 4 colour pairs, higher resolution character mode, is also beautiful anyway, but only a couple of programs use it, although I think that well and beautifully only Geco's Panic-man has achieved it: http://www.ep128.hu/Ep_Games/Leiras/Panic_Man.htm). So I decided to write an editor because there are no tools available for EP that would allow this mode to be exploited. It was my goal too, that the end result can be used in BASIC. Although the editor is written in BASIC language itself(but it can only be properly used in the emulator, due to its low speed), the finished artwork can be saved as a BASIC program (levels, colour rows, fonts).
Since my time is not infinite, this was also a limiting factor, was for that reason I tried to write a program which, In addition to the low cost(?), it could be used efficiently.
The most difficult task was to figure out how to present this character mode the best way to use it. The 4 pixel width is small, so the editor is based on 2x2 character blocks, from which you can build a track and edit 2x2 characters at once. Of course, 1 character editing is also possible. The 2x4 colour is such a limiting factor that there is a separate palette editor where we can edit small colour groups of 4 colours (but there are many built-in), and colour them on the screen by line. The row colouring, of course, is poked, because BASIC and EXOS don't admit this, i.e. the palette can not be different per row.(they don't support the 2x4-colour mode, only the 1x4 colour one, so the video mode is also poked.) So we can easily colour our levels?(of course not only the levels? can be modified, but whatever else aspect is suitable of modification on this kind of character mode.)

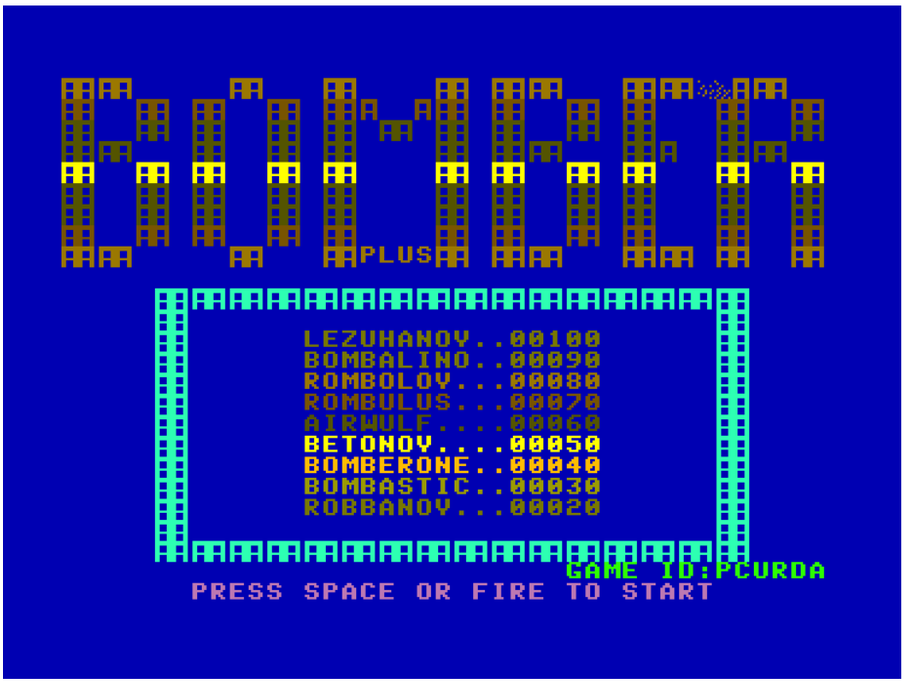
The drawing session approx. looks like this: we draw characters(tiles-blocks), then we build a level (more precisely from the 2x2 blocks) and embed it. Of course, the order is not tied, there may be another character, and sometimes you have to re-colour the level again.

There is a feature for background colouring that draws random gradients. Again and again we invite you to use this function and soon find the right background colours. There is one extra function in the row that copies the 0-colour to the 5 one. The purpose of this is to have the same background for the characters on the two 4 colour groups. But, of course, it is better to work in drawing characters of one of the colour groups that fill them completely,, because we have more colours, no need to waste one for the background. This may be a little confusing, but whoever uses the system will soon come up with how to use the program.

Endre Baráth

A missed bomb is a good opportunity

Written by
Tamás Bodnár
(Szipucsu)



Inside the Enterprise demo tape, a game called the Bomber can be loaded and played. It's like the bomber games seen on other computers, they basically do more or less the same. This one does not exploit the capabilities of the Enterprise except for the stereo sound.

The aim of the programs of the demo cassette was, beyond the possibility of using them,
that the user could list, modify, try, what he wanted to change them. Programmers probably didn't deliberately take the most out of these programs, though the officially „released" programs would have been filled with many visual and sound effects, since there was not too much to buy and much less, a program that really utilized the machine's capabilities.

The first problem that the user finds after loading the bomber game is that the key-click is not switched off. We can easily overtake this ourselves, but after some game play we will discover more bugs. After a lot of practice at last we can finally succeed knocking down all the houses. Then, at the end of the game, while we will be enjoying pressing almost all the time the space bar and the bombs will miss the buildings one after the other, the falling sound will not be silenced even if the bomb has hit the ground. If we toss a lot of bombs down, the sound buffer will become full, and the game will stop until it is released. This can easily avoided by putting the CLEAR QUEUE n command in the correct location, or by inserting an INTERRUPT into the SOUND parameters of the bombing release.

During the great excitement you might not be able to realize that, if we have reached the top of a house, and we drop the bomb, the nose of the plane (first character) disappears covered by the bomb, then a space is overwritten, and as long as the house collapses, the plane will remain „naked". Not only in such an exciting situation, but when every bomb is dropped,
It happens that the flying nose disappears for a moment, which is not very aesthetic. This is likely to help, If the bombing cycle begins to count down one character less from where the plane is...

If a taller building is hit and completely collapsed, this will take a while. Occasionally, the SOUND instruction on the flying engine sound will be stopped all this time. It is also easy to fix only increasing the number after DURATION.

It is unknown if the programmers have deliberately avoided a RANDOMIZE command at the beginning of the program. This can be intentional, so every time you start, the program will adjust the height of the houses the same way. This can be good for those who would like to practice, but the game could have been more varied if there was a different track every start.

It may be a little surprising that the play field is filled writing to channel 0 (default EDITOR: channel). It could have been better to write to channel 102 (VIDEO: channel), so at the beginning of the game, drawing houses could have been faster. The design of houses could be accelerated in a different way, disabling the wait-state (OUT 191.12) and disabling interruptions could have dramatically increased the speed. There are oth-

er many „advantages" to using the 0 channel, so that if a novice user stops the game with the STOP, he or she can draw on the track either from homes or planes and after CONTINUE it may be surprising that his work is completely ignored. So, in vain, if he delete the space from the top of the houses, they will stay there, and the planes that we draw do not even go into the bombing.

However, if channel 102 is used for drawing, it would be a bit faster, and so it would have been easier to use another colour. Just because the height of the houses is stored in the program array (not read from the screen), more colours could be used.

But even using the 0 channel it could have been possible to use another colour to draw houses.

Not to mention that the program could have been done, with almost insignificant extra work, to run on an attribute screen so it could have used up to 16 colors on the screen at once, which would have made a different impression for example, in Centrum Shops for those who saw the program.

Sounds are also very simple. As for the possibilities, they have practically not been used in the program It only uses the enveloping curve for the bomb's fall, and even for the sound of the flying, but it did not need it. Perhaps it is understandable that they have been left open to the user to experimentally modify the sounds. Dave's options would have allowed more varied explosion sounds and bombshells on the 3 + 1 channel, including the envelope, filters, and ring modulation.

There is no need for a game to have a music score, which is not here. However, taking advantage of these possibilities could have been used to make music in the program using very powerful sounds, which would also have raised the first impression at Centrum Stores.

All in all, the program included some minor bugs, and programmers were remarkably minimalist. It is almost the same program as the bomber game of any other modern computer, although it could provide much more. Perhaps they have left the opportunity for the user to fix the bugs and develop the game further? Perhaps the character defining program was also placed on the bomber (as it might be on other machines) so that we can modify the shape of the flying ship, whether it is a boat, a chicken, a seaweed(?), or a laser guided bomb, and the houses covered with clay, or whatever we want. But we can not save it and we can not get it back.

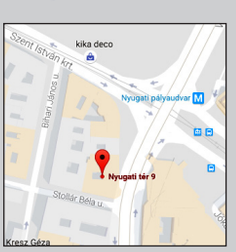
It is true that here, but not in the printed newspaper, in the electronic version you will see all the pictures in color. (Ed.)

Enterprise club, 15. april 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, 2-3, March-May
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