ENTERPRESS

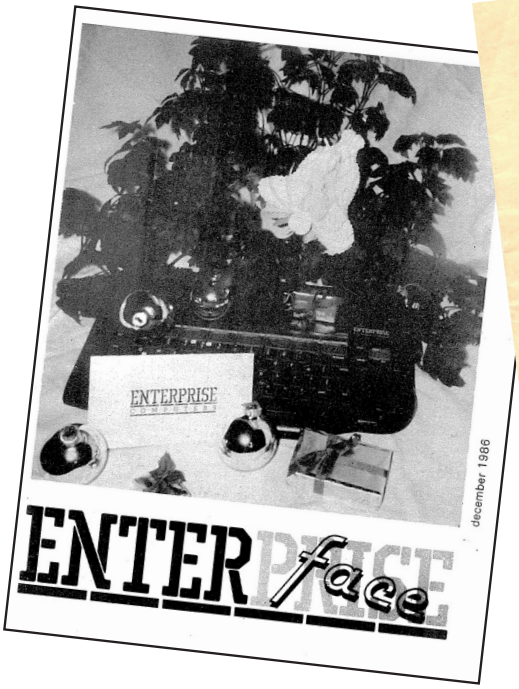
Magazine for the Enterprise users

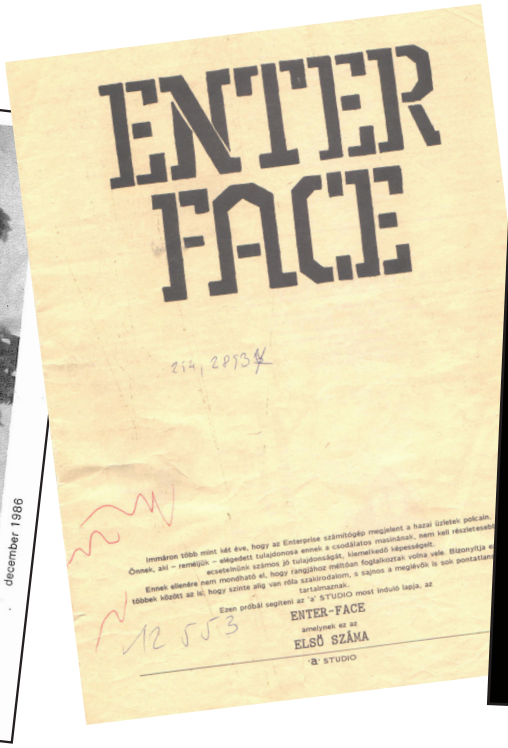
2017/5-6. September–December

The Golden Baton

File management in Pascal

Enterprise publications in the world 1.

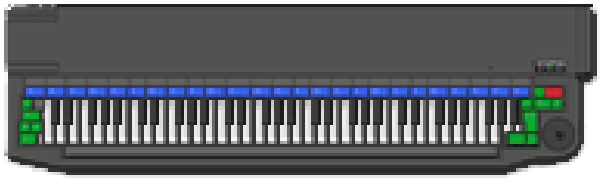




Enterprise MIDIDISP

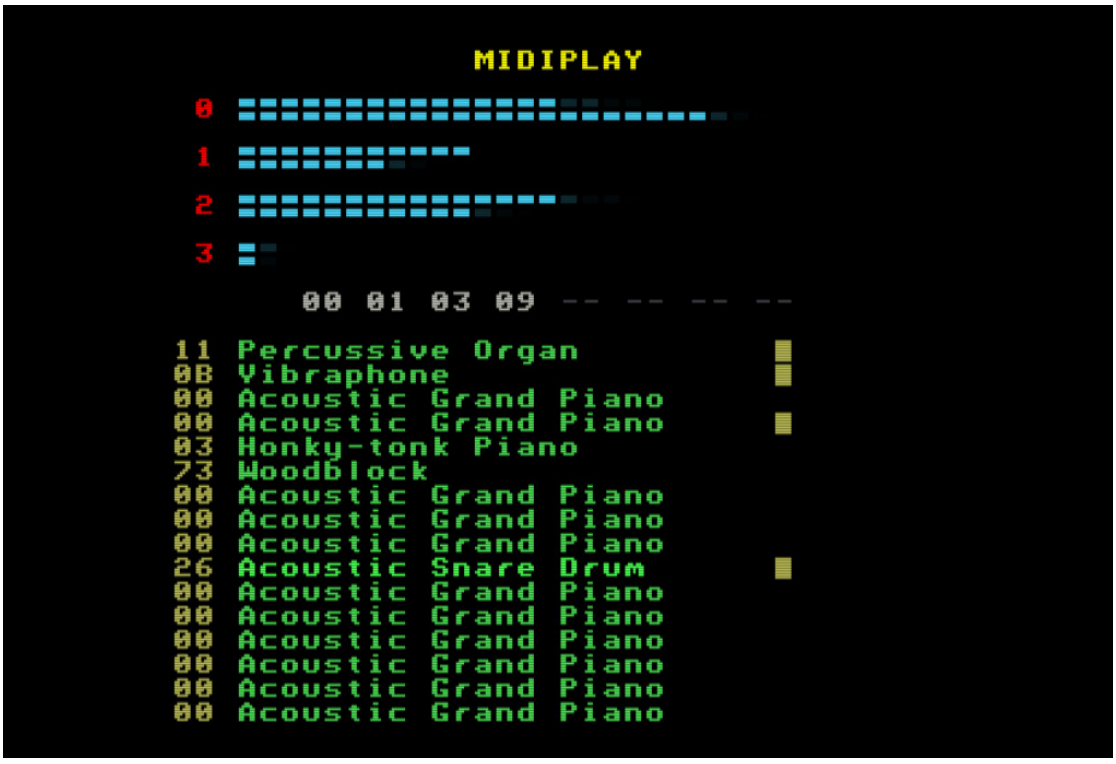
by IstvanV

Written by: Bodnár Tamás (Szipucsu)



IstvanV has created a new version of Midiplay, the MidiDisp version. The difference with the previous version is that it now shows the volume of the 4 channels as well as the Midi instruments used at the bottom.

Midiplay was a great sensation this year. It can be used on the emulator or on an Enterprise machine. You can also connect to an emulator using an external Midi keyboard (we tried it successfully on one of the Enterprise Club Days!). We look forward to having real Midi hardware for Enterprise.



Z80 reset

Written by: Németh Zoltán (Zozosoft)

I managed to solve a bug that caused me a headache for several months :-)

It happened when I was developing and testing EXOS 2.4. It worked fine on the emulator, but when I tried it on the real machine I get consternated: It got freezed whenever the reset button was pressed, and no further investigation about reset helped me to discover what was causing it. Sometimes it occured even right after it was turned on.

Finally, returning to my backups and comparing with the versions, I found where the error occurred. I had introduced an unconditional jump without first checking a given condition. :-) Ok, I had made a mistake but, why didn't the error happen in the emulator?

I was searching the Web at what would happen on reset to the Z80 registers, when I found a description where, based on the information of a mysterious „Matt", was written that at reset AF and SP will always be loaded with FFFFh.

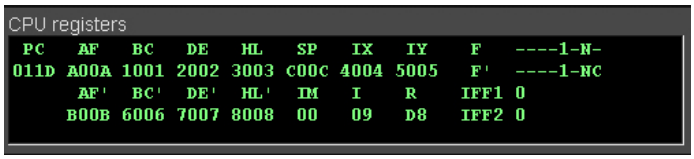
> *Matt has done some excellent research on this. He found that AF and SP are always set to FFFFh after a reset, and all other registers are undefined (different depending on how long the CPU has been powered or, different for different Z80 chips).*

I thought that the ep128emu also operate based on this description (aditionally also IX and IY registers are set to FFFFh, and I haven't found where they where written :oops:). So, it was understandable, why the error did not emerge on the emulator.

From here, however, the idea that this description seemed to be false came straight! And I had to figure out what it was really doing.

I created a tiny program inserted on the system-board Rom that shows the value of the registers, then loads them with values and waits for the reset button to be pressed on an infinite loop. Then, I could see how the reset would affect the loaded values.

Conclusion:



Several different Z80s have been tried, but, as you can see, it has nothing to do with that description, maybe sometimes it's accidental. The registers are completely randomly loaded, except for the PC and I, R which will be zeroed as described in the Zilog description.
Probably due to some manufacturing technology, the Zilog and ST chips prefer bits to 1 and NEC better to 0. The most interesting is the piece made by NDK, which loves AAh values very much.
However, after a lot of resets this is the result:

That is, all registry values are retained except for PC and I, R which will be zeroed.

This was the case with Ep128emu:

If I look well:

```
void Z80::reset()
 {
   R.PC.L = 0;
   R.I = 0;
   R.IM = 0;
   R.IFF1 = 0;
   R.IFF2 = 0;
   R.RBit7 = 0;
   R.R = 0;
   R.AF.W = 0xFFFF;
   R.SP.W = 0xFFFF;
   R.IX.W = 0xFFFF;
   R.IY.W = 0xFFFF;
   R.Flags &= ~
      (Z80_EXECUTING_HALT_FLAG |
       Z80_CHECK_INTERRUPT_FLAG |
            Z80_EXECUTE_INTERRUPT_HANDLER_FLAG |
Z80_INTERRUPT_FLAG |
       Z80_NMI_FLAG | Z80_SET_PC_FLAG);
   newPCAddress = -1;
 }
```

This could be fixed by deleting the four FFFF assignments.

EXOS compatible memory management - part IV.

Written by: Németh Zoltán (Zozosoft)

Question by Povi:

```
                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
```

Something is not clear to me here. If I understand well the code, then, in the case of a LPT memory request, if we get a shared segment, the LPT will be on segment 255. But why? Why is LD C, 255?

Answer from Zozo:

The previous EXOS 23 call spoils the value of C, and we restore its value.

Question by Povi:

I understand, but a shared segment can only be 255?

Answer from Zozo:

Your doubt is understandable, the answer is yes too :)
Since we are using program header 5, in this case all channels are closed, so it means that you will not need more the EXOS environment. If we say we would act as an extension,
when an open video call may occur, EXOS will be very limited. (Iview also closes all channels when loading modules, it is not EXOS who manages the image file).

Question by Ferro73:

I have read, but I have not found an opportunity, although I have seen some of its forms.

My idea:

```
ld a,255
out (0b2h),a
ld hl,(0bf91h)     ;exos  lower limit
ld a,(0bf9eh)
cp 0
jp z,nincsoszt
cp 255
jp nz,nemrencers
ld bc,hl  ;ex hl,bc ; push hl  pop bc
ld hl,0
ld de,8000h
ldir
ld a,255
out (0b0h),a
```

Because I didn't find any indication of how much and what portion of EXOS to use from segment 255. I am thinking of tape: disk and other temporary storages.

The LPT table would fit somewhere on the rest of the space and then we can release the FC segment.

Since there is a ROM between 0-3FFF on ZX machines, it will not interfere with system memory (FF, FD, FC, FE)
If I remember well, at the end of the ROM 0 disassembly it is explained how to use the system segment area. Do you have this book for someone? I couldn't find the full eBook.

Answer from Zozo:

> I have read, but I have not found an opportunity, although I have seen some of its forms.

The idea is good, but there are several errors in the practical implementation.

> *ld a,255*
> *out (0b2h),a*
> *ld hl,(0bf91h)* ;exos lower limit
> *ld a,(0bf9eh)*

Eg. on the other side it is EXOS 2.0, so it is not worth to take its spirit directly as EXOS :)

> *ld bc,hl ;ex hl,bc ; push hl pop bc*
> *ld hl,0*
> *ld de,8000h*
> *ldir*

It fails to check, how we can control it?

> *ld a,255*
> *out (0b0h),a*

Because I didn't find any indication of how much and what portion of EXOS to use from segment 255. I am thinking of tape: disk and other temporary storages.

> It's constantly changing! This is why you have to allocate it as a shared segment and then EXOS will tell you how much you can use (set user boundary).

> Since there is a ROM between 0-3FFF on ZX machines, it will not interfere with system memory (FF, FD, FC, FE)

That's right, that's what I've done in my Spectrum transcript since 3 years.

Here the point is highlighted:

```
PRGSIZE: EQU 0B00H  ; our program
                    ; should choose the size,
                    ; but it should be rounded to 10H
         LD SP,100H
         LD HL,HIBA
         LD (0BFF8H),HL
         LD A,12
         OUT (191),A
         LD BC,100H+26
         LD D,0
         EXOS 16
         LD BC,100H+28
         LD D,255
         EXOS 16
         LD BC,100H+27
         LD D,0
         EXOS 16
         HALT
         LD HL,(0BFF4H)
         SET 6,H
         LD B,4
ELPT     SRL H
         RR L
         DJNZ ELPT
         LD A,L
         LD (ELPTL),A
         LD A,H
         OR 0C0H
         LD (ELPTH),A
         LD HL,(0BFF4H)
         LD DE,STATUS
         LD BC,8
         LDIR
         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
         EXOS 24
         LD A,C
         LD (P2S),A
         JP NZ,HIBA
         EXOS 24
         JR Z,OKEP3
         CP 7FH
         JP NZ,HIBA
         LD A,C
         LD (LPTS),A
         OUT (0B3H),A
         CP 0FCH
         JP C,HIBA
         LD DE,PRGSIZE+200*16
         EXOS 23
         JP NZ,HIBA
         DI
         IN A,(0B0H)
         LD (P0S),A
         LD (P3S),A
         LD DE,PRGSIZE
         LD (VIDCIM2),DE
         LD HL,0
         LD DE,0C000H
         LD BC,PRGSIZE-1
         LDIR
         IN A,(0B3H)
         OUT (0B0H),A
         LD (0BFFCH),A
         LD A,(P3S)
         OUT (0B3H),A
         LD A,(LPTS)
         JR NEMHIBA1
OKEP3    LD A,C
         LD (P3S),A
         OUT (0B3H),A
         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    CALL LPT
            LD A,(VIDS)
            OUT (0B1H),A
            LD A,(P2S)
            OUT (0B2H),A
            XOR A
            LD DE,(VIDCIM2)
            LD HL,VIDCIM2+1
            RRD
            RLCA
            RLCA
            RLCA
            RLCA
            LD (LPTL),A
            OUT (82H),A
            OR 0C0H
            RRD
            LD (LPTH),A
            OUT (83H),A
            LD (VIDCIM2),DE
```

Creating a LPT, which I mentioned below, changes compared to the original procedure because it is not at a fixed location, so you need to handle the changing video addresses.

```
LPT:    LD DE,(VIDCIM2)
        LD HL,0
        RRA
        RR H
        RRA
        RR H
        ADD HL,DE
        LD (VIDCIM2),HL
LPT2:   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
L1      EX AF,AF'
        EXX
        LD HL,LINE
        LD BC,16
        LDIR
        EXX
        LD (HL),E
        INC HL
        LD A,D
        RRA
        RRA
        AND 3
        OR 18H
        OR (IX+1)
        LD (HL),A
        INC HL
        LD (HL),E
        INC HL
        LD (HL),D
        ADD HL,BC
        INC D
        LD A,D
        AND 7
        JR NZ,L2
        LD A,E
        ADD A,32
        LD E,A
        CCF
        SBC A,A
        AND 0F8H
        ADD A,D
        LD D,A
L2      EX AF,AF'
        DEC A
        JR NZ,L1
        EXX
        LD HL,SYNC
        LD BC,HOSSZ
        LDIR
        RET
LINE    DB 255,14H,15,2FH,0,0,0,0
TABL    DB 0,36,121,88,130,182,219,63
SYNC    DB 0F5H,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0
STATUS  DB 247,8,11,73H,0B8H,
          0FEH,0E9H, 1,0,216, 216,0,0,0,0,0
        DB 217,12H,63,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB 253,16,63,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB 252,16,6,63,0,0,0,0,0,0,0,0,0,0,0,0
        DB 255,90H,63,32,0,0,0,0,0,0,0,0,0,0,0,0
        DB 252,12H,6,63,0,0,0,0,0,0,0,0,0,0,0,0
        DB 207,13H,63,0,0,0,0,0,0,0,0,0,0,0,0,0
HOSSZ   EQU $-SYNC
```

The end

File management in Pascal

Written by: Povázsay Zoltán (Povi)

HiSoft Pascal does not recognize file types. But there are two non-standard procedures for managing files, which we can use(with some limitations) to perform simple file operations:

Using the TOUT method, you can save any area of memory to a mass storage device:

tout(filename: string; address, count: integer);

The procedure writes the byte count and the starting address on the file header. The procedure automatically creates the file (if you open an existing file, it will be overwritten) and then closes it at the end of the operation.

You can retrieve the saved file using the TIN procedure:

tin(filename: string; address: integer);

The procedure loads the filename file to the specified memory address. But the number of bytes you want to load cannot be specified, the entire contents of the file will be loaded at the address.

Both procedures use channel 122, be careful to not open channel 122 before calling the procedures (in plain English: do not use this channel)!

Let's see an example of how to use the procedures!
Suppose there is an array in which we store the names and scores of the top ten players:

```
const fn = ‚HISCORE';

type
  TSCORE = record
    name  : packed array[1..3] of char;
    score : integer;
  end;

var
  scores : array[1..10] of TSCORE;
```

The scores array can be written to the disk as follows:
tout (fn, addr (scores), size (scores));

And when we start our game, we can load the saved scores as follows:
tin(fn, addr(scores));

What are the disadvantages of these two procedures? One minor problem is that they are slow: the files are written and read character by character (Using exos 7 or exos 5); while using block operations would be much faster (exos 8 and exos 6).

But the bigger problem is the lack of error handling! In the example above (reading scores from disk), if the file HISCORE does not exist on the disc (which is a common occurrence when the game is first launched), our game would stop with an error message! It would be good if we could handle the incorrect file operations that might occur (such as opening a non-existent file for reading, or writing to a read-only disk, etc.) ourselves.

Another limitation in the procedures is that we can't position the file pointer (we only can read or write from the beginning, there is no possibility to add data at the end of the file)

To resolve these issues, look on the following magazine number for how to create your own file management procedures using Exos functions.

Implementing string management functions

Written by: Kiss László (Lacika)

In Pascal, therefore, a string of characters always stores the same number of characters, and this is settled when you specify the type of index when defining the type. If you want, for example, to use a string whose length varies during the program run, the following defining type is recommended:

```
CONST MAXLGTH=78;
TYPE STR=RECORD
       LGTH:INTEGER;
       TEXT:ARRAY[1..MAXLGTH] OF CHAR;
END;
```

The MAXLGTH constant specifies the maximum length of the string used. We store in the LGTH field of the record that at the moment the first text element of the text field stores the actual text.

The NEV variable type can be conveniently written in the usual way:

```
WRITELN(NEV.TEXT);
```

When reading the value of a variable we also have to specify the length:

```
PROCEDURE READSTR(VAR ST:STR);
VAR I:INTEGER;
BEGIN
  READLN;READ(ST.TEXT);
  IF (ST.TEXT[MAXLGTH]<>CHR(0)) THEN ST.LGTH:=-
MAXLGTH
  ELSE BEGIN
    I:=0;
    REPEAT
      I:=I+1
    UNTIL ST.TEXT[I]=CHR(0);
    ST.LGTH:=I-1
  END
END;
```

If you want to read a text string named NEV in a STR variable, you simply have to call the procedure:
READSTR(NEV);

Thus, is very easy to do an assignment between two identical STR type variables:
NEV2:=NEV;
You can also create string management procedures and functions, the easiest of which is to query the length of the text string stored in the variable:

```
FUNCTION LENGTH(ST:STR):INTEGER;
BEGIN
  LENGTH:=ST.LGTH
END;
```

The value of a STR variable can be deleted by the procedure below:

```
PROCEDURE ERASESTR(VAR ST:STR);
VAR I:INTEGER;
BEGIN
  FOR I:=1 TO MAXLGTH DO
    ST.TEXT[I]:=CHR(0);
  ST.LGTH:=0
END;
```

The content of two STR variables can be concatenated with the CONCAT method, the result will go to the variable specified as the first parameter.

```
PROCEDURE CONCAT(VAR ST1,ST2:STR);
VAR I:INTEGER;
BEGIN
  I:=1;
  WHILE (ST1.LGTH<MAXLGTH) AND (I<=ST2.LGTH)
DO BEGIN
    ST1.TEXT[ST1.LGTH+1]:=ST2.TEXT[I];
    ST1.LGTH:=ST1.LGTH+1;I:=I+1
  END
END;
```

Using the COPYSTR method, you can copy the specified portion of a string variable to another string variable.

Template: COPYSTR(VAR1,n1,n2,VAR2)

The character on the n1. position will be copied to n2. position and the generated result stored in VAR2, its previous content overwritten.

```
PROCEDURE COPYSTR(ST1:STR;POS1,POS2:INTE-
GER;VAR ST2:STR);
VAR I:INTEGER;
BEGIN
  ERASESTR(ST2);
  IF POS1<1 THEN POS1:=1;
  IF POS2>ST1.LGTH THEN POS2:=ST1.LGTH;
  FOR I:=POS1 TO POS2 DO
    ST2.TEXT[I]:=ST1.TEXT[I];
  ST2.LGTH:=POS2-POS1+1;
  IF ST2.LGTH<0 THEN ST2.LGTH:=0
END;
```

With DELSTR, you can delete the specified part from the content of a string variable. Template: DELSTR (VAR, n1, n2). The deletion is done from the n1 to the n2 positions. The COPYSTR and DELSTR procedures are „dummyproof", ie they control the specified parameters.

Using the DELSTR procedure, you can easily create the LTRIM method, which deletes spaces at the beginning of a string.

The RTRIM method, which deletes the spaces at the end of the string, can be created on a similar way:
Az UCASE eljárás egy string-változót nagybetűsít (a kisbetűket nagybetűkre cseréli, minden más karaktert változatlanul hagy):

The UCASE method capitalizes a string variable (replaces lowercase letters with uppercase letters, all other characters remain unchanged):

```
PROCEDURE LCASE(VAR ST:STR);
VAR I:INTEGER;
BEGIN
  FOR I:=1 TO ST.LGTH DO
    IF ST.TEXT[I] IN [‚A'..'Z'] THEN
      ST.TEXT[I]:=CHR(ORD(ST.TEXT[I])+32)
END;
```

The POS node searches for the position of the first occurrence of the text stored in the ST2 variable in the text string stored in the ST1 variable. If the text you are looking for is not found, the returned value of the function is zero.

```
FUNCTION POS(ST1,ST2:STR):INTEGER;
VAR I,J:INTEGER;
BEGIN
  I:=1;J:=1;
  REPEAT
    IF ST1.TEXT[I]=ST2.TEXT[J] THEN J:=J+1 ELSE J:=1;
    I:=I+1;
  UNTIL (I>ST1.LGTH) OR (J>ST2.LGTH);
  IF J=ST2.LGTH+1 THEN POS:=I-J+1 ELSE POS:=0
END;
```

Eg, with the following function, any string, except a null one(empty), can be converted to an INTEGER type number. For the sake of simplicity, the largest convertible number is 32759.

```
FUNCTION VAL(ST:STR):INTEGER;
VAR I,SZ:INTEGER;
    N:BOOLEAN;
BEGIN
  I:=0;SZ:=0;N:=FALSE;
  REPEAT
    I:=I+1;
    IF ST.TEXT[I]='-' THEN N:=TRUE;
  UNTIL (ST.TEXT[I] IN [‚0'..'9']) OR (I=ST.LGTH);
  IF ST.TEXT[I] IN [‚0'..'9'] THEN BEGIN
    SZ:=ORD(ST.TEXT[I])-48;
    FOR I:=I+1 TO ST.LGTH DO
      IF (ST.TEXT[I] IN [‚0'..'9']) AND (SZ<3276) THEN
        SZ:=SZ*10+(ORD(ST.TEXT[I])-48)
  END;
  IF N THEN VAL:=-SZ ELSE VAL:=SZ
END;
```

System segment memory map

- for non-EXOS compatible programming

EXOS 2.0 Address	EXOS 2.1 Address	Name
(BF97h)...(BF99h)-1	(BF91h)...(BF93h)-1	Channel area
(BF99h)...(BF9Bh)-1	(BF93h)...(BF95h)-1	Device descriptors
(BF9Bh)...(BF9Eh)-1	(BF95h)...(BF97h)-1	RAM areas allocated to the ROM extensions
(BF9Eh)...(BFA0h)	(BF97h)...(BF9Ch)	Extension ROM list
(BFA0h)+1...(BF9Eh)...ABD5h	(BF9Ch)+1...(BF9Ah)...ABD0h	RAM segment list
ABD6h...	ABD1h...	EXOS work area
...B21Bh	...B216h	Z80 Stack under EXOS operations
B21Ch...B234h	B217h...B22Fh	EXOS paging routines
B235h...B256h	B230h...B251h	„Written by: Mrl BT NMV GNH CGE AEL"
		Devices work area
B680h...BAFFh	B480h...B8FFh	Character Font (128 characters)
(BFF4h)BB00h...BD1Fh	(BFF4h) B900h...BB1Fh	Line Parameter Table
		EXOS work area
BEBCh...BEE3h	BEB8h...BEDFh	Status Line
		EXOS work area
BF1Dh...	BF18h...	Default Device
		EXOS work area
BF76h	BF72h	Time: Second (BCD)
BF77h	BF73h	Time: Minute (BCD)
BF78h	BF74h	Time: Hour (BCD)
BF79h	BF75h	Date: Day (BCD)
BF7Ah	BF76h	Date: Month (BCD)
BF7Bh	BF77h	Date: Year (BCD), started at 1980
		EXOS work area
BF82h	BF7Eh	ROM CRC for secret protection routine
		EXOS work area
BF97/8h	BF93/4h	Bottom of the Device descriptors
BF99/Ah	BF95/6h	Bottom of the RAM area allocated to the ROM extensions
BF9B/Ch	BF97/8h	End Pointer of the ROM list
BF9Dh	BF99h	
BF9E/Fh	BF9A/Bh	Pointer of the RAM list
BFA0/1h	BF9C/Dh	Start Pointer of the ROM list
BFA2h	BF9Eh	Segment number of the SHARED segment (0 if none)
BFA3h	BF9Fh	Number of free segments
BFA4h	BFA0h	Number of USER allocated segments
BFA5h	BFA1h	Number of DEVICE allocated segments
BFA6h	BFA2h	Number of SYSTEM segments
BFA7h	BFA3h	Number of working segments
BFA8h	BFA4h	Number of defective segments
BFBE/F/C0h	BFBA/B/Ch	First pointer of the Channel chain
BFC1/2/3h	BFBD/E/Fh	First pointer of the Device chain
BFC4/5/6h	BFC0/1/2h	First pointer of the System Extension chain
BFC9h	BFC5h	IRQ_ENABLE_STATE
BFCAh	BFC6h	

BFCBh	BFC7h	CODE_SOFT_IRQ
BFCCh	BFC8h	DEF_TYPE
BFCDh	BFC9h	DEF_CHAN
BFCEh	BFCAh	TIMER
BFCFh	BFCBh	LOCK_KEY
BFD0h	BFCCh	CLICK_KEY
BFD1h	BFCDh	STOP_IRQ
BFD2h	BFCEh	KEY_IRQ
BFD3h	BFCFh	RATE_KEY
BFD4h	BFD0h	DELAY_KEY
BFD5h	BFD1h	TAPE_SND
BFD6h	BFD2h	WAIT_SND
BFD7h	BFD3h	MUTE_SND
BFD8h	BFD4h	BUF_SND
BFD9h	BFD5h	BAUD_SER
BFDAh	BFD6h	FORM_SER
BFDBh	BFD7h	ADDR_NET
BFDCh	BFD8h	NET_IRQ
BFDDh	BFD9h	CHAN_NET
BFDEh	BFDAh	MACH_NET
BFDFh	BFDBh	MODE_VID
BFE0h	BFDCh	COLR_VID
BFE1h	BFDDh	X_SIZ_VID
BFE2h	BFDEh	Y_SIZ_VID
BFE3h	BFDFh	ST_FLAG
BFE4h	BFE0h	BORD_VID
BFE5h	BFE1h	BIAS_VID
BFE6h	BFE2h	VID_EDIT
BFE7h	BFE3h	KEY_EDIT
BFE8h	BFE4h	BUF_EDIT
BFE9h	BFE5h	FLG_EDIT
BFEAh	BFE6h	SP_TAPE
BFEBh	BFE7h	PROTECT
BFECh	BFE8h	LV_TAPE
BFEDh	BFE9h	REM1
BFEEh	BFEAh	REM2
Not exist	BFEBh	SPRITE
Not exist	BFECh	RANDOM_IRQ
Not exist	BFED/Eh	USER_ISR
BFEFh		CRDISP_FLAG
BFF0/1h		SECOND_COUNTER
BFF2h		FLAG_SOFT_IRQ
BFF3h		PORTB5
BFF4/5h		LP_POINTER
BFF6/7h		ST_POINTER
BFF8/9h		RST_ADDR
BFFA/Bh		STACK_LIMIT
BFFCh		USR_P0
BFFDh		USR_P1
BFFEh		USR_P2
BFFFh		USR_P3

The Golden Baton

1983 - Digital Fantasia - Text adventure

Written by: Kiss László (Lacika)

On the 2017 / 2-3. issue of this magazine, in the „game summary", published in a short half-sentence, we recalled the Mysterious Adventures series of Brian Howarth. This unpublished tutorial has been now released for the trivial reason that Geco's software can emulate these games perfectly (including saving / loading the game)!

When Brian Howarth left his telecom engineering career in 1981, BASIC listings of Dungeons & Dragons role playing games and Adventure could be typed from the contemporary computer newspapers. This prepared him to move from writing small programs to writing complete text adventure games. However, the contemporary BASIC games did not really appeal him, so he wrote his primitive text-only games in machine code on his TRS-80 microcomputer. His solution was called Interpreter (similar to Infocom's „Virtual Computer"), making it possible to create 11 adventure games in just two years: the game was in a data file, the Interpreter engine did not change.

After the release of more advanced home graphics computers, his games were released in the Mysterious Adventures series (Atari, Acorn Electron, BBC Micro, Commodore 64, Commodore Plus / 4). Spectrum versions arrived in 1983. Then these games were quite special: they were made entirely in machine code - that was not at all general at that time - and the Spectrum and C64 versions also provided graphics for all locations (most of the adventure games of the time had no graphics at all...). But this had a price because when there were a lot of rooms in the game, the graphics could not be stored inside the program, and they have to use „schematic drawing" lines made by an algorithm on the program, followed by solid colour shapes - coloured by a filling out routine - quite slow. Partly due to the large space requirements of the images, the vocabulary had to be significantly reduced and the text descriptions of the sites were limited to verbs. The games only understand the verb + noun combination of words - it was quite usual then. However, it could be frustrating to play them because of the extremely short vocabulary, almost without synonyms, as there is only one word for every object or verb in the game. As a result, more time is spent searching for words known by the program than for actual play. That means that even non-English speakers can play them, but unfortunately, it undermines the gameplay of games that contain interesting stories and imaginative puzzles. An interesting, though less useful, feature is that all games in the series support the Currah Micro-Speech Speaker system, which reads aloud what is happening.

However, the 11 games released in the Mysterious Adventures series came out at the precise time, so they have become classic despite all their errors.

The Arrow of Death, which appeared in two parts, takes place in the same world as the Golden Baton, in which we have to kill the evil Xerdon with a magic arrow.

Escape from Pulsar 7 is a sci-fi adventure game inspired by the Alien movie.

Feasibility Experiment is a treasure hunt game like Adventureland.

In Time Machine, we are going to rescue a scientist, Dr. Potter, in space and time.

In Circus we have to escape from a haunted circus.

The Wizard Akyrz is the villain of this title, after kidnapping the daughter of our king and stealing the values of the royal treasure.

Perseus and Andromeda introduces us into Greek mythology - with the history the two classic characters.

In the Ten Little Indians game, we have to find an extremely valuable piece of art on the property of a recently deceased art collector.

In the surreal Waxworks, revived wax figures of the Story try to chase us.

The first game of the series is Golden Baton, where the plot is around a golden magic wand. This priceless magic object holds the balance be-

tween good and evil in the kingdom of Ferrenuil. But one day it was stolen from the king's palace. The imbalance between good and evil will manifest itself in droughts and plagues. Our job is to find and acquire again the gold magic wand.
For those who want to try the game alone, the management of the program will be firstly summarized.

We can navigate with the following commands:

GO NORTH, N
GO WEST, W
GO SOUTH, S
GO EAST, E
GO DOWN, D
GO UP, U
After the GO command, a location name must be added (which refers to the description in the program).

Commands for managing the game:

SAVE - save game status,
QUIT (abbreviated as Q) - Exit the game. The program waits for a confirmation and then asks if we want to start a new game. Whenever a new game is started - including after the program is loaded - it asks if we want to load a game. There is no other way to load a previously saved game state (no LOAD command).
INVENTORY (abbreviated as I): list of objects carried by the player.
SCORE: The only „custom" command in the game. All what it answers is that this is not a football match...
HELP: usually it only returns trivial things: try to explore the area, let's examine everything. But sometimes it really helps!
Press ENTER to turn the graphics display on and off. With graphics turned off, we get a short text description of the site and the objects in it. Since this description is blocked by the image, you should constantly switch between the graphics and the description. However, drawing the already ugly graphics („schematic" on the name of kindness) slows down the game so much that it is better to cancel them.

Additional vocabulary for this program:

AKRYZ	FEED	LIZARD	RUNES
ARCH	FILL	LOOK	SAIL
AROUND	GET	MATCHES	SALT
ATTACK	GIVE	MIRROR	SAY
BARREL	HAMMER	OPEN	SEARCH
BATON	HELMET	PARCHMENT	SLUGS
BLOW	HIT	PATH	STABLES
BRIARS	HOLD	PILE	STAFF
BURN	HOLE	POLISH	STRAW
CABIN	HOLLOW	PORTCULLIS	STREAM
CAVE	HORN	POUR	SWIM
CHOP	JUMP	PUT	SWORD
CLIMB	KEY	QUARTZ	TAKE
CLOAK	KILL	RAFT	THROW
CRAB	KNIFE	READ	TREE
DOOR	LAKE	REMOVE	UNLIGHT
DROP	LAMP	RING	WAVE
EAT	LEAVES	ROPE	WEAR
EXAMINE	LIGHT	RUB	WOLF

Words can be abbreviated, just typing their first four characters. Note that we only can carry five objects at the same time.

One possible solution is:

We begin our adventure in a dense enchanted forest (Figure 1). A worn-out cloak and a pile of dried leaves are lying there. The cloak can come in handy, the leaves don't seem to be too interesting, but it can't be a problem if we look at it (GET CLOAK, WEAR CLOAK, EXAMINE LEAVES). We were not wrong, we found a bright sword under the pile. (GET SWORD, or TAKE SWORD)! To the south (S), where dense thorns close the way, we came to a dead end. If we look closely (LOOK BRIARS), the program confirms that they are really sharp. But we have a sword, we can hit them a little (CHOP BRIARS), we also find a rope between the spikes (GET ROPE).
Let's go north(N, N, N).We get to a clearing where a wolf looks at us "straight at the eyes". We can see a path, but the wolf does not let us pass ... Feed it (FEED WOLF) we can't, but still carry a sword...(ATTACK WOLF or KILL WOLF). We will not need the sword anymore (DROP SWORD). We have not reached the path, but let's go backwards (S, W). We are now under an old, huge oak tree. Examining the tree doesn't help us, but if we throw the rope to the tree as a loop it gets stuck in one of his branches and we can climb on it (THROW ROPE, CLIMB TREE).On the tree trunk we find a cavity that hides a ring (LOOK HOLLOW, GET RING). When we examine the ring (EXAMINE RING), we find extrange rune symbols on it. It does not seem dangerous if we wear the ring on one finger. (WEAR RING). We descend to the ground and then recover the rope with a firm pull. (D, GET ROPE). Now we can return to the path (E, N, GO PATH). We're on a forest road, and see a stick in front of our feet (GET STAFF). If we look at it closely (EXAMINE STAFF), there are engraved rune symbols on it, like on the ring.
The trail leads north (N, N, N), and a moat prevents us from passing. Of course there is also a castle, we should get ini. THROW ROPE or GO MOAT don't seem to work(from the second command we are answered „Sorry?"), only (SWIM) allows us to advance. A grilled trapdoor appears on the pool so, the "wacky" adventurer will attempt everything to open the grid, which will not happen, of course, and he will leave the game after an hour of trying. Later he will find that the proper tool has gotten wet inside the INVENTORY . Not too logically, we can throw the rope onto the castle wall from our position on the water, so we can climb the castle battlements (Pic. 2) (THROW ROPE, CLIMB ROPE). To the south we can now walk out of the castle through the grilled trapdoor(that will not let us go), but we choose the other option: down the fortress (D). A scary black armored knight is watching there to only let pass through the arched passage who wears a cloak, which we have already put over (GO ARCH).
Our joy doesn't last for long, a massive closed door blocks our way. We can't go back south, but at least have the east and west ways. To the East (E) we get to a barn without horses but we can see an ivory hunting horn and a rusty

helmet. So far, if we had tried to read the rune symbols on the bot, we would only have known that they were illegible. However, only if we wear the helmet (GET HELMET, WEAR HELMET), we can read (READ STAFF or READ RUNES), the magic word „AKRYZ". Although there is not much (no)logic in the story, the „wacky" player will carry the seemingly useful helmet until the end of the game, but in vain, later we will not be able to use on anything ... (REMOVE HELMET, DROP HELMET).

We can only go backwards (W). More creative may have been trying to „grouch" the ring that looks like a magic ring, instead of rubbing it. Those who haven't done that, are now trying (RUB RING, or POLISH RING). In the ring, a genie „dwels", which gifts us helpful things. Now he has dried our match. This is definitely useful, it is not hard to guess that we will need it on a near future.. Let's see if the genie has more to offer(RUB RING): he give us a key (if the match is not wet, the key is given first). Unfortunately, the genie's repertoire has been exhausted, and nothing happens for the third time (REMOVE RING, DROP RING). The key, however, we will need to use it at just this door (GET KEY, UNLOCK DOOR, DROP KEY, OPEN DOOR). As we'll just use the key on the door of this castle, there is no need of a cloak (REMOVE CLOAK, DROP CLOAK).

The "wacky" adventurer would enter enthusiastically the door, where a wizard named Gorgon would turn him into stone with his „rigid" gaze. However, the thoughtful adventurer will first try the (HELP) command, with the achievement that the program will exceptionally help us: „a mirror would not hurt". Instead, we look at what is in the west (W). There is a shed with scattered hay and an oil lamp (GET LAMP). Let's see what the hay hides (LOOK STRAW): a hole! Of course it is dark, light the lamp (LIGHT LAMP) - which of course is only possible with a dry match - and we can go down (GO HOLE). We got to a chamber where a door opens (GO DOOR) into a cozy torture chamber (TORTURE-CHAMBER). To the north (N) we get to the servants' room (picture 3) (who was the decorator in this castle ???), where a mirror is pierced (GET MIRROR).

To the south of the torture chamber (S, S), we find the wizard's workroom, where our eyes are caught by a glowing quartz crystal lamp. The term "lamp" shall be taken to mean:

if we try to pick it up we will not be able, because it's too hot (LOOK QUARTZ). (HELP) suggests to try with magic movements. Now we have to use our other magic tool, the stick, and there are no words in the vocabulary ... (WAVE STAFF). The program reacts: „Something is happening!" Dirt... The quartz crystal is still hot so we have to spell the magic word (SAY AKYRZ). The quartz crystal floating between the white flames levitates to the table (DROP STAFF, GET QUARTZ). To the East (E) in a dark room we meet a grotesque lizard. If we accept the previous advice of the program to examine everything (EXAM LIZARD), it says us that the lizard is not very friendly. At least this can be also inferred from the fact that without any word we are killed on any action. The (HELP) command, however, helps us again: the use of the quartz crystal can be useful here (WAVE QUARTZ): a dazzling ray of light destroys the lizard! When we observe its carcass (EXAMINE LIZARD), we find a jewel-decorated knife. Do not hesitate to loot a corpse (DROP QUARTZ, GET KNIFE)! In the torture chamber (W, N), we've probably been attracted by a big hammer (GET HAMMER).

Now with the mirror, we can go back to the big door (W, U, UNLIGHT LAMP, E). How do we defend ourselves from the „frosty" gaze of the wizard? We know that the mirror should be used, hold it in front of us, and let's go (HOLD MIRROR, GO DOOR). The room is full of statues... In addition, Gorgon, with his own gaze reflected on the mirror, turned the mirror himself into a stone. The danger is therefore over (DROP MIRROR) and we can read the parchment found here (GET PARCHMENT or GET PARC, READ PARCHMENT). Only short fragments can be read: „Boat on the lake ... blow the horn ... throw ..." We've already seen a horn, so it's worth taking to us (DROP PARCHMENT, S, E, GET HORN). With this, we have completed the exploration (looting) of the castle, we can leave (W, S, U, S, S).

Let's go back south along the path up to the intersection where we have crossed several times (Figure 4) (S, S, S, S, S). From here to the west (W) is the oak, which we climbed up, to north (N) a clearing with a small cottage (Figure 5).Of course we go inside (GO CABIN). We'll find an oil-soaked cloth here, if we have walked in the dark of the castle and the lamp went out of oil, we can refill it there (FILL LAMP). More interesting is the hole in the floor, where a staircase leads down to a room, and the barrel (EXAMINE BARREL) in the middle turns out to contain salt. Surely the salt will be good for something, it's definitely dark there too, but we manage to take full hands of it (LIGHT LAMP, DROP MATCH, GET SALT, D).Heading south (S), there is an open door in our way. The (HELP) command advises you not to smile, crushing the padlock with this hammer will not cause any more problems (SMASH PADLOCK, - or HIT PADLOCK -, DROP HAMMER, GO DOOR). We got to a storage room, where we find a little raft. We read about a lake, certainly a boat (GET RAFT) will come in handy. We can only go backwards, north of the tunnel (N, N), where we see a large green slug crawling.

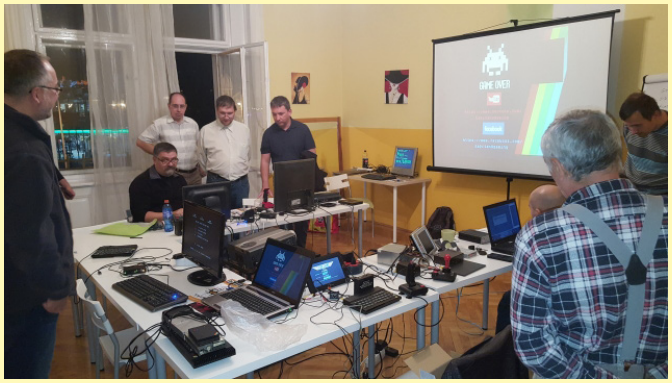
Yuck! Salt and Slugs? Perhaps the relation (GIVE SALT or DROP SALT) is obvious. Take the dead slugs (GET SLUGS). Going north (N, N) we found a huge crab stopping our way, not letting us to pass over, but behind it is the cave exit to the lake (Figure 6). Maybe he's hungry (HELP), but he can't be „beaten" ... Maybe he loves the salty snail! (If the snails are not picked up, (HELP) will tell you that the crab loves salted swine. Snail-chips ...) (FEED CRAB or GIVE SLUGS, but also DROP SLUGS will work). The satiated crag retires aside.

Now it only lefts us to follow the parchment readable text:

sail (GO LAKE or SAIL LAKE), then blow the horn (BLOW HORN). Now it will happen an unexpected turn: a huge hand appears that squeezes the ominous magic wand. We can't catch the wand out of the hand because it moves restless... When we cut it with the knife

then the wand is drop down (THROW KNIFE, GET BATON).

After getting the magic wand, the program congratulates us on having completed the task.



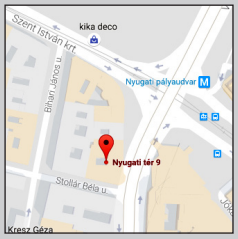
Enterprise club,

28, october 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 - 2017/5-6, September-December
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