ENTERPRESS

Magazin az ENTERPRISE felhasználóknak

2022/3-6. május - december

SymbiFace 3 újdonságok

Bobby Bearing

XORKA logikai játék a xor elvén

Pear-féle EXDOS v2.0 compact floppyvezérlő tuningja

Erős utolsó negyedév

Írta: Matusa István (Tutus)

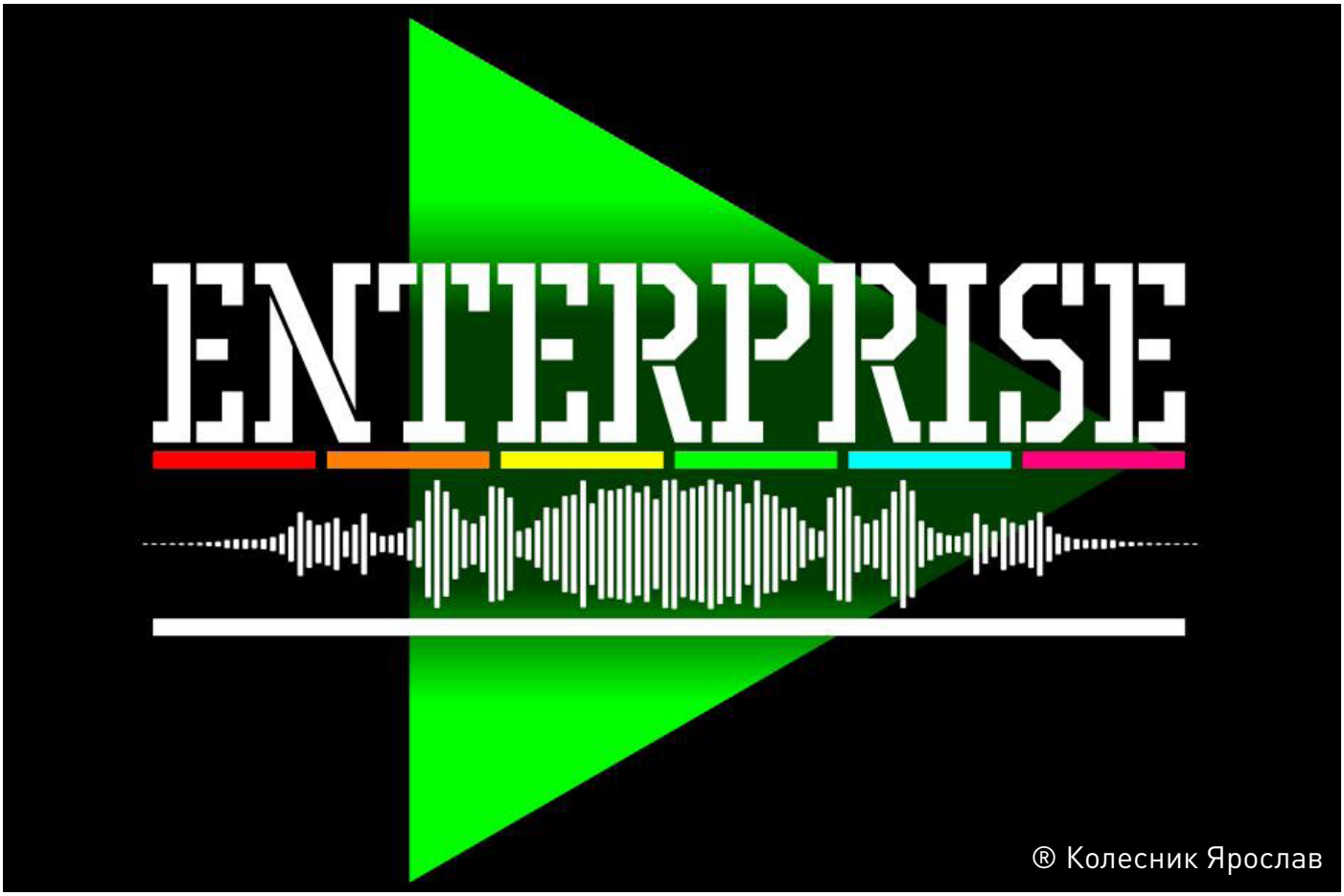
Végre kicsit fellélegezhetett mindenki, így mi és Klubunk is a hosszú Covid időszak után. Ismét tudtunk klubtalálkozókat szervezni, jó volt újra látni a régi arcokat, jókat beszélgetni kedvenc gépünkről vagy éppen valami más témáról.

Megpróbáltam rendbe tenni valahogy a Covid időszak alatt szétkuszálódott előfizetési rendszert, így év végére szerintem sikerült ez az akció. A lényeg, hogy megpróbáljuk, hogy év közben ne lehessen előfizetni, hiszen az előfizetés egy adott, teljes évre szól (ez az Enterpress Magazin szempontjából fontos).

Kis közösségünk 2022 utolsó negyedévét egy óriási hajrával zárta! Olyannal, hogy én csak pislogtam mindenfelé: itt is egy új fejlesztés, ott is... Nagy öröm számunkra ez, idei utolsó (nagy) összevont számunkban ezekről be is számolunk. De csak címszavakban: SF3 kártyára több program és driver is készült, melyet **Hansnak, Geconak, Zozonak** köszönhetünk (SF3 fájl kezelés, FTP, RTC). SFMSD: EXDOS bővítés, hogy a SymbiFace 3-ra dugott USB meghajtót tudja kezelni az EXDOS, ezt is **Zozonak** köszönhetjük.
Vaczkó Károly klubtagunk hardver fejlesztései: L2 panel, REPower modul, Joystick illesztő, RGB adapter. Készülőben a szuper turbó kártya.
EP128emu Libretro core verzió **Zöldalmától,** Raspberry Pi-re.
Pear-féle EXDOS kártya tuningja **Dr.OG**-től
MiSTer és Neptuno Enterprise FPGA projektek **Kyp**-től.

Valamint az év végi utolsó klubnapunk előtt kitaláltunk egy olyan projektet, mely reményeink szerint jelentősen megváltoztat majd mindent az Enterprise számítógép körül hazánkban. Ez még legyen meglepetés, hamarosan beszámolunk majd erről is! :)

Enterprise Forever!



Zzzzip

A BASIC compiler - 2. rész

Peter Hiner - 1986.

RESTORE

Ha adott egy sorszám (mint pl. a RESTORE 200 -ban), ez egy DATA-t tartalmazó sor kell, hogy legyen. Máskülönben a ZZZIP hibaüzenetet ad.

RUN

Ezzel a paranccsal egy lefordított program újraindítható (az elejétől vagy egy kijelölt sorszámtól). Egy másik program (BASIC vagy lefordított) betöltésére és futtatására is használható, de egyik programból a másikba történő paraméterátadás már nem lehetséges.

STOP

Programban szereplő parancsként a STOP az END-hez hasonlóan kezelődik, s a BASIC-hez való kilépést eredményezi. A STOP billentyű egy lefordított program megszakítására is használható, akárcsak a BASIC-ben, de a lefordított program futása a megszakítási ponttól már nem folytatható a CONTINUE segítségével. Újra kell indítani élőiről, a START utasítással.

EXCEPTION handling (megszakítás kezelése)

Az EXCEPTION szó már önmagában is azt jelenti, hogy valami hiba következett be, s ezért nehéz ezt lekezelni egy lefordított programban. Ennek ellenére a ZZZIP megkísérli egy jól használható EXCEPTION kezelő biztosítását. Nem valószínű, hogy a fordítás során problémák lennének, mivel a ZZZIP minden kapcsolódó parancsot elfogad, beleértve a CONTINUE és a RETRY parancsokat is. A lehetséges problémák futás során jelentkeznek. A CAUSE EXCEPTION parancs nyomán előállt helyzetek nem okozhatnak zavart, de ha egy másik forrásból, mint például egy EXOS-beli hibaellenőrzésből bekövetkező EXCEPTION után próbáljuk használni a CONTINUE parancsot, az eredményt illetően semmi biztosíték nincs! A ZZZIP részéről két Különleges korlátozás áll fenn. Csak egyszintű EXCEPTION kezelés biztosított, más szóval a WHEN EXCEPTION parancsok egymásba skatulyázása nem megengedett és bármely második megszakítás egy hibaüzenetet, valamint a BASIC-be történő visszatérést eredményez. Kis mértékben a RETRY parancs is különbözik - a lefordított program ahelyett, hogy abba a sorba menne vissza, amelyikben az EXCEPTION előfordult, a legutóbbi WHEN EXCEPTION parancshoz fog visszatérni. Természetesen megfelelő elrendezéssel elérhetjük a kívánt visszatérést, mégpedig úgy, hogy a WHEN EXCEPTION sort közvetlenül az elé a sor elé tesszük, amelybe vissza akarunk térni (például, egy INPUT sor elé).

CHAIN, IMAGE, PRINT USING, TRACE, TYPE

Ezeket a parancsokat a ZZZIP nem fogadja el és használatuk a fordítás alatt hibaüzenetet eredményez.

6. beépített függvények

A ZZZIP megtart minden beépített függvényt, habár néhány esetben korlátozza használatukat, ezekről külön említést teszünk. Talán meglepő lehet, hogy a ZZZIP megtartja a SIN és COS trigonometrikus függvényeket, hisz ezek valójában nem kompatibilisek az egész értékű matematikával. Ezeket használni is nehezebb, addig ne is foglalkozzunk velük, amíg nincs elegendő tapasztalatunk a ZZZIP működtetésében.

BIN(X)

Azokra az X értékekre, amelyek nem haladják meg a 32767 értéket, a függvény a szokásos módon használható a zárójelben lévő változó vagy konstans értékű operandussal. Nagyobb értékek esetén az operandus csak konstans lehet pl. BIN(11010101), s ezt az esetet a ZZZIP megkülönböztetetten kezeli.

CEIL(X), INT(X), IP(X), ROUND(X.N), TRUNCATE(X,N)

Mivel az egész értékű matematika nem veszi figyelembe a törteket, ezek a függvények gyakorlatilag nem csinálnak semmit, egyszerűen az X értékét vissza.

EPS(X)

Ez a függvény mindig az 1 értéket adja vissza. Az 1 az egész értékű matematikában az értékváltás legkisebb egysége.

EXLINE

Egy lefordított programban a végrehajtás alatt álló sor számának a rekordja nem őrződik meg, s ezért ez a függvény mindig a 0 értéket adja vissza.

FP(X)

Mivel az egész értékű matematika nem veszi figyelembe a törteket, ez a függvény mindig a 0 értéket adja vissza.

FREE

A függvény a lefordított program által felhasználható byte-ok számát adja vissza, ahogy ezt el is várnánk. Ellenben, ha ez az érték meghaladja a 32767-et, negatív számként fog megjelenni.

INF

A függvény mindig a 32767 értéket adja vissza. Ez az előjeles, 16 bites bináris jelölésben a legmagasabb érték.
PI

A függvény mindig a 3 értéket adja vissza.

RGB

Ez a függvény szokásos értelmezésében egy 0 és 1 közé eső értéksorozatot vár, mellyel az elsődleges színeket definiálhatjuk. Kivételes eset, hogy a ZZZIP elfogad ilyen értékeket, feltéve, hogy azok decimális formában lévő konstansok. Tehát az RGB(0,.4,1) elfogadható, ellenben az RGB(0,3/7,7/7) vagy az RGB(X,Y,Z) már nem. Megjegyezzük, hogy a ZZZIP csak az első számjegyet használja a tizedespont után, s ezért az RGB(0,.4,.99)-et úgy tekinti, mint az RGB(0,.4,.9)-et.

RND és RND(X)

Az RND(X) pontosan úgy működik, mint a BASIC-ben. Az RND önmagában mindig a 0 értéket adja vissza. Az RND*X különleges eset, amikor is a visszaadott érték megegyezik az RND(X) által szolgáltatott értékkel.

VERNUM és VER$

A VERNUM függvény mindig, az 1 értéket adja vissza azért, hogy az EXOS 2.0 és 2.1 változatait ne lehessen összetéveszteni (az egész értékű matematikában a 2.0 és 2.1 mindegyike 2 lenne). A VER$ függvény azonban egy olyan karaktersorozatot ad vissza, amely az éppen használatban lévő EXOS változatát jelzi.

Trigonometrikus és Logaritmikus függvények

Az olyan függvények, mint a COS és a SIN, valójában nem kompatibilisek az egész értékű matematikával, mert értékeik mindig a 0 és 1 között helyezkednek el. Azért, hogy működjenek ezek a függvények, arányosító tényezőket fogunk használni. Ezeket a COS esetében részletezzük, bemutatva az elvet, majd megadunk egy olyan részletes táblázatot, mely minden érintett függvényt tartalmaz. Hogy használható eredményeket kaphassunk, a COS(X) által visszaadott érték automatikusan arányosítódik (beszorzódik) az 1000-es tényezővel azért, hogy egy 0 és 1000 közé eső egész értékké váljon (a 0 és 1 közé eső érték helyett). A program további részében ez az eredmény már használható, de ne felejtsük el, hogy egy arra alkalmas helyen osztanunk is kell 1000-rel, hogy pontos eredményt kaphassunk. Természetesen az 1000-rel való osztás helyét gondosan kell megválasztanunk, hogy elkerüljük az eredmény nullára való csökkentését, ez a hely a COS(X) értékének egy más értékkel való beszorzása után legyen. Ezt a külön lépést, (az 1000-rel osztást) a BASIC program szokásos vizsgálata után helyezzük el a programban, de még a fordítás előtt. Ha a BASIC-ben is így vizsgálnánk, téves eredményt adna. Ha a szögek meghatározásakor a DEGREES-t használjuk, a COS(X)-ben lévő X operandus megegyezik az eredeti BASIC programban szerepelő értékével. Ellenben, ha a RADIANS-t használjuk, az X-et arányosítani (szorozni) kell 1000-rel a COS(X) használata előtt. Ismét ügyelnünk kell arra, hogy ezt a helyet gondosan válasszuk meg a programban, annak érdekében, hogy értelmes értéket kapjunk. Nem jó, ha hagyjuk, hogy az X értéke 1 alá essen és azután szorozzuk be 1000-rel, hisz ebben az esetben az X értékére eredményként mindig 0 adódik!

Függvények	Bejövő arányosító	Kimenő arányosító
ACOS ASIN ATN	1000	1 (fok) vagy 1000 (radián)
COS COT CSC SEC SIN TAN	1 (fok) vagy 1000 (radián)	1000
DEG	1000	1
RAD	1	1000
EXP	1000	1
LOG LOG2 LOG10	1	1000
COSH SINH TANH	1000	1000
ANGLE(X,Y)	1	1 (fok) vagy 1000 (radián)

Megjegyezzük, hogy egyes trigonometrikus és logaritmikus függvények 32767-nél nagyobb értéket adhatnak (különösen, ha az arányosításnál megszorzódnak 1000-rel). Ezekben az esetekben a kijövő érték automatikusan 32767-re korlátozódik. Ugyanez vonatkozik a hatvány függvényre (^).

7. A lefordított programok másolása

A lefordított programok csupán a SAVE parancs használatával nem másolhatók szalagra. További másolatok készítésének egy módja az lenne, ha a BASIC programot ismételten lefordítanánk. Egy másik módszer, amely csak egy szalagos készülék használatát igényli, azt a lehetőséget használja ki, amelyet az egyes lefordított programokat megelőző BASIC betöltő program tartalmaz. A Basic képernyőeditorból indulva, a bent lévő programok törlése után, folytassa a következőkkel:
Helyezze be a lefordított programot tartalmazó szalagot és készüljön fel a betöltésre.
Gépelje be a LOAD „XXX”-et (ahol XXX a program neve).
Helyezze be a célszalagot (azt, amelyikre a másolatot szeretné kapni) és készüljön fel a kimentésre.
Gépelje be a SAVE „XXX”-et.
Helyezze be a lefordított programot tartalmazó szalagot és készüljön fel a betöltésre.
Gépelje be a RUN 100-at.
A program jelezni fogja, hogy mikor kell betennie a célszalagot és előkészülnie a kimentésre.
Ekkor nyomja le az ENTER billentyűt és a lefordított program kiíródik a célszalagra.

Szükség lehet arra, hogy a program felismerje magáról, hogy eredeti basic programként, vagy Zzzippel fordított formájában fut-e. Mivel a Zzzip nem kezel törteket, erre a legegyszerűbb megoldás a PI értékének megvizsgálása. Ha a ZZZIP változó értéke 1, akkor a lefordított program fut:

```
LET ZZZIP=1
IF PI>3 THEN LET ZZZIP=0
```

Így későbbi programrészünkben átugorhatjuk a lassító részeket, ha a ZZZIP értéke nulla.

A Zzzip néha furcsán működik. Előfordulhat például, hogy a megjegyzések (REM vagy !) utáni részek is kiakasztják. Az Entersnake játékban ehhez hasonló sorok voltak:

```
2340 CASE "x" !extra life
```

A Zzzip fordítási hibával leállt. Ha a ! utáni részt töröltük, már működött a fordítás.
Ha DATA sorok végére teszünk megjegyzést, azt a Zzzip sztringként kezeli:

```
100 DATA 0,0,0,0 ! comment
```

Ennek a végéről a „0 ! comment" a Zzzip számára egy sztring. Ezért a legjobb a ! jeles megjegyzéseket mellőzni, vagy fordítás előtt törölni. Erre a legjobb módszer, ha a basic programot kilistázzuk egy PC-s txt fájlba, töröljük a megfelelő részeket, és külön fájlként elmentjük. Így később visszatérhetünk a kommentelt programunkhoz. Azonban a txt fájlt közvetlenül nem tölthetjük be a Zzzipbe, mivel az csak a tokenizált utasításokat ismeri fel! Előbb basicbe kell betölteni a load paranccsal, a basic minden további nékül beolvassa, és innen kell a save-vel elmenteni:

```
200 DATA „,",0,1
```

A basic így értelmezi: <vessző>,0,1
A Zzzip így értelmezi: <idézőjel>,<idézőjel>,0,1

Ha DATA sorokba vesszőt teszünk idézőjelbe, a Zzzip azt vesszővel elválasztott két idézőjelnek fogja értelmezni. Így vesszőt, mint olyat nem tudunk így beolvasni, ezt máshogy kell a programban megoldanunk.
A túl bonyolult műveletek is okozhatnak problémát. Egy ilyen sor is kifektette a Zzzipet:

```
1460 FOR WADE=1 TO(100*SPEED-DIF*((120+
PLUSIDO)-IDO))*ZIP
```

Nem is az adott sorban jelezte a hibát, hanem teljesen másik sor fordításakor jött elő, és nem valós sorszámot írt hibás sornak. Ha a műveleteket szétszedtük több sorba, már ment. Olykor nem érthető, mi okozza pontosan a hibát a basic program lefordításakor. Egyszer egy eljárásnak EP nevet adtam, a fordítás így leállt. Az eljárást átneveztem EPTXT-re, így már jól működött. Nem valószínű, hogy volt EP nevű változó a programban, mert az a basic program futtatásakor is hibát okozott volna. Olyan is volt, amikor egy MIN(x,y) függvényt tettem a programba, és az már nem működött, de az eredeti basic programban rendben volt.

A Zzzippel lefordított program basic betöltőjét nem lehet sokáig tovább bővíteni. Ha az elejére beteszünk jó sok sort, amiben például PRINT utasítások vannak, már nem indul el a Zzzippel lefordított program, bár betöltődik, és a rendszer eléggé instabil lesz. Ez kár, mert a basic betöltőhöz hozzáírogatva lehet feliratokat, rajzokat tenni a képernyőre, de ezt nem szabad túlzásba vinni.
A törteket a Zzzip kerekíti. Ha esetleg envelope numbert alarunk törtet megadni, akkor tehetjük az envelope numbert utólag a lefordított program betöltőjébe is. Másik megoldás, ha egy külön basic programból escape szekvenciaként elmentjük az összes szükséges envelope-t, és a Zzzippel fordított program ezt egy külön fájlból visszatölti. Így az envelope nem foglalja a helyet a lefordított programban. Az escape szekvencia adatait tehetjük a programon belülre, data sorok után, és onnan is beolvashatjuk, így nem kell külön fájlból betölteni.
Fontos figyelni arra is, hogy amikor a Zzzippel fordítunk, milyen EXOS van a gépben. Ugyanis az EXOS 2.3 magyar verziójával hiba csúszhat a fordításba, és lehet, hogy a lefordított program nem fog megfelelően működni. Ez a villogó pixelek problémája, ami az EXOS 2.31-es verzióban már ki lett javítva.
Talán többeknek feltűnt, hogy a Zzzip által kimentett basic betöltőben vannak olyan részek, melyekre nincs szükségünk. A betöltő első sora rögtön a 200-as sorra ugrasztja a programot, és a Prepare to SAVE program felirat soha nem jelenik meg. Ez azért van, hogy ha a RUN 100 paranccsal indítjuk a betöltőt, akkor magát az egész lefordított programot le tudjuk másolni. Természetesen a SAVE paranccsal csak a betöltőt lehet kimenteni, az egész lefordított programot nem, így ezzel a módszerrel akartak a felhasználókon segíteni, akik több példányban akarták tárolni lefordított programjukat. Természetesen másolóprogramokkal, vagy egyszerűen a COPY #x TO #y utasítással is meg lehetett oldani ezt a problémát. A mai világban erre pláne nincs szükség, így megszállottak ki is törölhetik a lefordított programból ezt a részt, de csak óvatosan, hogy maga a lefordított program azért betölthető maradjon!

```
        IS-BASIC      program    0
LIST
    10 GOSUB 200
    20 CALL USR(6051,0)
    30 END
   100 GOSUB 200
   110 PRINT "Prepare to SAVE program"
   120 INPUT PROMPT "and then press
       ENTER key":B$
   130 OPEN £106:A$ ACCESS OUTPUT
   140 CALL USR(6025,0)
   150 END
   200 FOR I=0 TO 143
   210    READ X
   220    POKE 5932+I,X
   230 NEXT
   240 READ A$
   250 OPEN £106:A$
   260 CALL USR(5932,0)
   270 RETURN
   280 DATA 42,26,2,34,254,23,17,0,23,
       1,16,0,62,106,247,6,183,223,17,
       0,24,42,4,23,25,229,17,0,184,
       183,237,82,225,56,4,33,31,36,
       231,34,32,2,34,26,2,17,28,0,25,
```

Érdekesség, hogy a NOPE'K programmal kilistázhatatlanná tett basic programok is lefordíthatók a Zzzippel, csak fordítás közben furcsa sorszámokat fogunk látni.

Hang és zene átírása az Enterprise és a TVC basic-je között

Írta: Bodnár Tamás (Szipucsu)

Néha váratlan szituációkba keveredhet az ember, amikor Enterprise-ról TVC-re, vagy TVC-ről Enterprise-ra kell átírnia olyan basic programot, melyben hang és zene is van. Ilyenkor nem szabad kétségbe esnünk! Először is inkább nézzük meg az eltéréseket a két gép hangzásbeli képességei között:

1. Az ENTERPRISE 3 négyszögjelcsatornát és 1 zajcsatornát képes egyszerre megszólaltatni, és a csatornák között különböző effektek (gyűrűmoduláció, alul- és felülátereszto szűrő) is használhatók. TVC-n egyetlen csatorna van.
2. Enterprise-on háromféle torzítás használható, TVC-n ilyenre nincs lehetőség, bár bizonyos trükkel elérhetünk torzított hanghoz hasonló hatást TVC-n is, lásd lejjebb.
3. Az Enterprise sztereo hangot képes kiadni, a TVC monót.
4. Az Enterprise-on a hangerő finomabban állítható: 64 hangerőfokozat van (ami 256 értékre van elosztva, így egymás melletti 4 érték ugyanazt a hangerőt jelöli), TVC-n 16 hangerőfokozat írja le a hang erősségét. Enterprise-on a 64 fokozat külön-külön értendő a jobb és a bal csatornára, és mind a 4 Dave-csatornán külön-külön megadható az értéke.
5. Enterprise-on a hangpuffer képes a memóriába letárolni a hangokat, amiket meg kell szólaltatni, így előre kiadott parancsok alapján szólhat a háttérben hang, miközben a gép például rajzolgat. TVC-n mindig a megszólaltatás pillanatában kell megadni a parancsot ehhez, így például rajzolgatás közben SOUND utasításokat is be kell tenni, azok nem adhatók meg a rajzolás előtt. A TVC hangpufferébe mindössze egyetlen hang fér, és az kitart addig, ameddig a hang időtartamában meg van adva, így kb. max 5 másodpercig szólhat egyetlen hang, míg a gép például rajzol a képernyőre.
6. Enterprise-on a hangerő és a hangmagasság egy hangon belül burkológörbékkel szabályozható, TVC-n erre nincs lehetőség, bár másfajta módszerrel elérhető ehhez hasonló hatás, például FOR-NEXT ciklussal.
7. Enterprise-on többféle hangmagasság érhető el. Elérhetők mélyebb hangok, mint TVC-n, ezen kívül a magasabb hangok között is finomabb hangmagasság-különbségek adhatók meg, és magas hangokból is több van.
8. A TVC basicjében alapból rövidebb hangokat is meg lehet adni, mint Enterprise-on, és ezeket az igen rövid hangokat gyorsan lehet váltogatni egymás után, ami Enterprise-on nem oldható meg alapesetben. A TVC-nek ez a funkciója próbál kárpótolni minket azokért, amik Enterprise-on nem érhetők el.
9. TVC-n az egyetlen utasítással megszólaltatható leghosszabb hang kicsivel több, mint 5 másodperc, míg Enterprise-on egyetlen utasítással több percnyi hosszúságú hang is megszólaltatható. Ennek a gyakorlatban nincs igazán haszna, jelentősége.

Hogyan festenek ezek a különbségek a gyakorlatban?

A hangkeltés a két gépen igen hasonló. Mindkét gép basic-je a SOUND utasítást használja. A paraméterek elnevezése is hasonló, de működésük már többé-kevésbé eltér. A TVC sound paraméterei:

PITCH, DURATION, VOLUME, ; (pontosvessző)

Az Enterprise sound paraméterei:

PITCH, DURATION, LEFT, RIGHT, INTERRUPT, SOURCE, SYNC, ENVELOPE, STYLE

A pitch és a duration a két gépen ugyanazt a hangtulajdonságot adja meg: pitch a hangmagasságot, duration a hanghosszt. A volume a hangerőt jelöli, ennek EP-n a left és a right felel meg együtt, külön a két csatornára. A ; (pontosvessző) azt jelenti, hogy ne legyen interrupt, ne szakadjon félbe az előző hang, míg az EP-s interrupt pont ennek az ellenkezője, tehát azt kell külön megadni, ha az előző hangot meg akarjuk szakítani. A source és a sync a több hangcsatorna kezeléséhez kell EP-n, az envelope-pal burkolót adhatunk meg, melyet előre már definiálnunk kell, a style segítségével pedig torzítások, valamint gyűrűmoduláció és szűrők érhetők el.

DURATION

A két gépen leginkább hasonlóan paraméterezhető paraméter a DURATION. Mindkét gépen 1/50 másodpercben adja meg a hang hosszát, így az itt megadott érték elvileg ugyanazt jelenti mindkét gépen. TVC-n maximum 255 lehet az értéke, Enterprise-on több ezer is lehet az értéke, de erre általában nincs szükség. Viszont a DURATION 1 érték TVC-n rövidebb hangot eredményez, mint EP-n, így több, eltérő hangmagasságú DURATION 1 érték egymás után TVC-n sajátos hatású, amit viszont EP-n nem tudunk egykönnyen elérni. Az ilyen rövid hangokból előálló hangzás többek között hangeffektekhez lehet jó TVC-n, és némileg hasonló hatás is elérhető így, mint Enterprise-on torzításokkal.
Fontos különbség, hogy TVC-n a DURATION 0 egy 0 hosszúságú hangot eredményez, magyarán nem szól semmi. Enterprise-on a DURATION 0 viszont az elérhető leghosszabb hangot eredményezi. (Erről már volt szó egy 2018-es számban, A sound utasítás rejtelmei című cikksorozatunkban.)

PITCH

Mindkét gépen a hangmagasság paraméterezhető vele, azonban teljesen máshogy. A legmélyebb elérhető hangot mindkét gépen a SOUND PITCH 0-val érhetjük el, ami EP-n mélyebb, mint TVC-n. A legmagasabb hang EP-n SOUND PITCH 127, TVC-n pedig SOUND PITCH 4094. (TVC-n a 4095 használható szünetnek, ilyenkor egyáltalán nincs hang. Enterprise-on kifejezetten ilyen érték nincs, bár a 127 és annál valamivel kisebb számok is olyan magas hangot jelölnek, amit gyakorlatilag nem hallani.) Enterprise-on tört PITCH értékek is megadhatók, TVC-n nem, de Enterprise-on sem fog minden tizeddel nagyobb szám a nála kisebbnél magasabb hangot adni, sőt a magas hangok tartományában több egész érték is ugyanazt a hangot jelöli EP-n.

Enterprise-on a normál C hangnak PITCH 37 felel meg, TVC-n 3349. EP-n eggyel növelve a PITCH értéket egy félhanggal magasabb hangot kapunk minden esetben, de tört értékek is megadhatók. TVC-n bonyolult képlettel adható meg a fél hanggal magasabb hangot jelölő szám. Zene átírásánál gyakorlatilag az EP-s PITCH értékekhez hozzá kell rendelni a TVC-s PITCH értékeket, ami nem bonyolult, de még egyszerűbb midiben szerkeszteni majd konvertálni zenét mindkét gépre.

Burkológörbék (envelope)

Az ENVELOPE NUMBER-t nem ismeri a TVC.

Mivel hangpuffer TVC-n gyakorlatilag nincs, a CLEAR SOUND-nak TVC-n nincs is igazán értelme, bár az utolsó hang tovább szól, miközben már tovább futhat a program. Ez megszakítható egy tetszőleges SOUND utasítással, így a CLEAR SOUND-nak leginkább megfelelő utasítás TVC-n egy SOUND DURATION 0 lehet.
- A SOUND DURATION 0 TVC-n tehát megszakítja az éppen szóló hangot. Tehát egy SOUND utasítás nem várja meg az előző SOUND lecsengését, hanem megszakítja azt. Hogy ne szakítsa meg, pontosvesszőt kell használni, például: SOUND; PITCH 3349,DURATION 25. Így, ha nem adunk meg pontosvesszőt, az olyan, mintha EP-n az INTERRUPT paramétert is megadnánk a SOUND-nál.

- A hangerő: TVC-n a VOLUME paraméterrel adható meg 0 és 15 közötti számmal: 0, ha egyáltalán nincs hang, 15 a leghangosabb. Enterprise-on a LEFT és a RIGHT paraméterrel adható meg külön-külön a jobb és a bal csatorna hangereje, ez 0 és 255 közötti szám lehet.

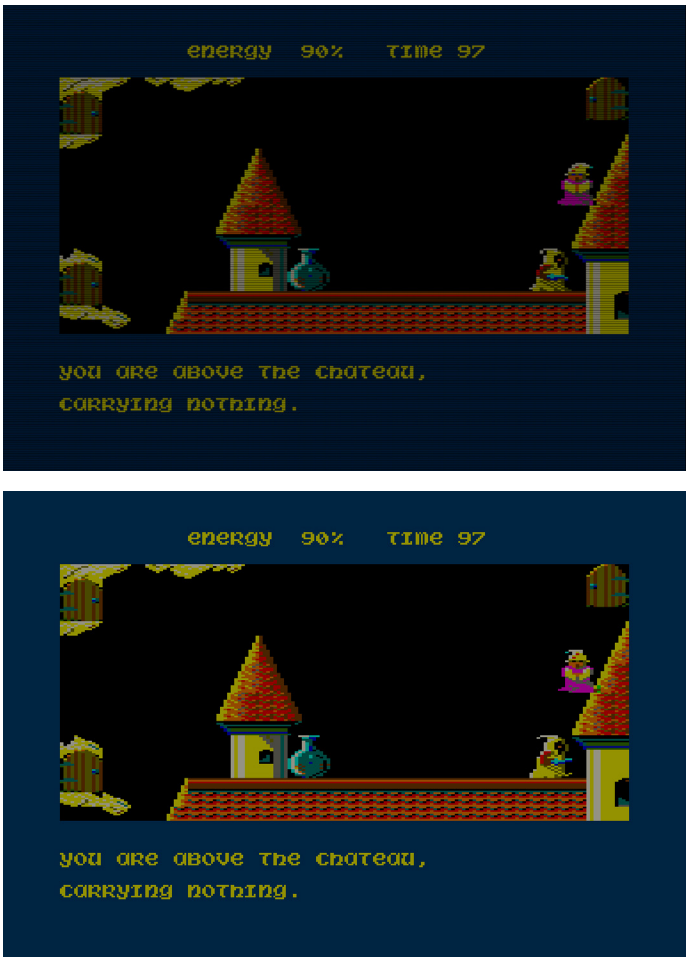
Ha egyik gépről a másikra átírunk egy basic programot, akkor a hangot legcélszerűbb elölről írni. Zenénél az egyes zenei hangoknak megfelelő pitch értékeket kell kicserélni (Enterprise-on könnyen kiszámolható, hogy egy adott szám milyen zenei hangot jelöl, míg a TVC gépkönyvében megtalálhatók az egyes zenei hangokhoz tartozó pitch értékek), hangeffekteknél pedig az adott gép képességeihez lehet igazítani a hangzást, például EP-re átírásnál torzítást, gyűrűmodulációt, sztereó hangot érdemes megszólaltatni, TVC-n pedig az egymás utáni rövid hangokkal és FOR-NEXT ciklusokkal lehet trükközni. Ha Enterprise-ról TVC-re kétszólamú zenét írunk át, DURATION 2 értékekkel felváltva meg lehet szólaltatni nagyon gyorsan egymás után a két szólam hangjait, ami egyetlen csatornán, jellegzetes hangzással játssza a két szólamot egyszerre. Az egyik szólam számára DURATION 1 is megadható, így sajátosabb lehet a hangzás.

Érdekesség: A hangmagasság és hanghossz paraméterezésére **Bruce Tanner** alkotta meg a PITCH és DURATION elnevezéseket. Az volt a cél, hogy a basic minél emberközelibb legyen. Valóban többet mond mindenkinek ez, mint más gépeken a hangkeltéshez használatos utasítások paraméterezése: a paramétereket egyszerűen vesszővel kellett elválasztani egymástól a CPC és a C16 ill. Plus/4-es gépeken, például SOUND 2,300,50. A spectrumos BEEP után is egymás után vesszővel elválasztva adhattuk meg a hanghosszt és a hangmagasságot, míg C64-en a POKE utasítással kellett a zenészeknek zöld ágra vergődniük.

ep128emu
Libretro core

A régi játékok újrafelfedezése, a retro gaming az utóbbi években több irányból is felélénkült. Egyrészt a hitelkártya méretű számítógépek olcsón biztosítanak platformot a különféle emulátoroknak, másrészt pedig megjelentek olyan projektek, amelyek egységes keretet adnak ennek a múltidézésnek. Ha valamelyik elterjedt játékkonzolt szeretnénk kipróbálni a múlt évezredből, csak annyi a dolgunk, hogy beszerzünk az internetről egy SD kártya image-et, elindítjuk vele - teszem azt – a tévére kötött Raspberry Pi-t, és egyetlen kontrollert kézbe fogva, az összes kiadott játékból válogathatunk, egy gombnyomásra indulnak.

Ebből a hullámból az Enterprise eddig kimaradt. Létezik hozzá jó minőségű emulátor (ep128emu), ami elfut ilyen környezetben, elvileg nincs akadálya, hogy akár ebben a formájában bekerüljön ezekbe a válogatásokba. De gondolkozhatunk ennél kicsit nagyobb léptékben is: a libretro projekt egy jól definiált környezetet (API-t) biztosít arra, hogy az emulátoroknak ne kelljen foglalkozni pl. az irányítás testreszabásával, telepítéssel, de még a képernyőfelbontással se. Ha ezt el lehet érni ep128emu-val, akkor lenne egy újabb, könnyen kezelhető opció Enterprise emulálásra, amiben van pár vonzó újdonság is. Innen indult a gondolatmenet még 2021-ben, és 2022 újévre fogalmazódott meg, hogy ennek én idén nekiállok. Szoftverfejlesztést láttam már, az ep128emu C++ nyelvezetében ugyan nincs rutinom, de azért megtanulható. Libretro oldalról van pár elérhető példaprogram, azok nem bonyolultak.



1. kép (shaderek)

Ahogy az várható volt, a legelső elképzelés, miszerint egyesével emelem át a szükséges részeket az új rendszerbe, rögtön életképtelennek bizonyult. Ahhoz, hogy egyáltalán forduljon a kód, a teljes emulációt át kellett venni, és ott vágni el, ahol az ablakkezelő rendszerhez csatlakozik. Ez eltartott pár hétig (természetesen hobbiként, nem főállásban), de január vége felé már képes volt legalább képet kiadni magából, és pár billentyűt érzékelni. Kicsit fura képet ugyan, de én örültem neki, mert látszott, hogy tud ez működni, és az ep128emu kódja is áttekinthetőnek bizonyult, meglettek a logikus kapcsolódási pontok. Innentől sorban jöttek a még nem működő funkciók. Hang - elvileg nem bonyolult, de a gyakorlatban igen érzékeny az időzítésre (nem is vagyok meggyőződve, hogy az 1.0-ban mindig jól működik). Interlace mód - jó sokat nézegettem, hogy mit csinál ez a fél sor vsync, de a végére csak meglett. Save state - az ep128emu snapshotok pont jók erre, csak nem fájlba kell irányítani. TVC emu - itt derült ki, hogy a színkezelés még nincs rendben, amikor megjelent az indítókép, csak épp pirosban. RPI kompatibilitás - megy az elfogadható sebességgel, ha a megjelenítőt kiszervezem külön szálba, és visszarakom bele az előzetes palettakonverziót, amit nem hoztam át, mert bonyolultnak tűnt addig. Március közepén tettem fel először kipróbálásra az Enterpriseforever fórumba, és április vége felé ért el egy stabil állapotot. Menet közben az eredeti emulátor részbe is került pár apróbb javítás, mint a hangszóró lekapcsolás lekövetése, vagy támogatás a 6 joystickhoz.

Pár szót arról, hogy miben is más ez, mint az ep128emu-t futtatni:
Megjelenítés: az emulátor alapesetben a szokásos PAL méretű képet adja tovább a libretronak (768x288 vagy interlace módban 768x576). Járulékos előnyként, mivel a

2. kép (teljes képernyő kitöltése)

valódi kimenő felbontással nem kell foglalkozni, be tudott kerülni egy közepesen intelligens nagyító funkció is, ami gombnyomásra levágja az egyszínű keretet, hogy a tényleges játék tartalom minél jobban kitöltse a képernyőt. *(Lásd. 2. kép).* Ezt a kimenetet aztán fel lehet skálázni, ahova csak szeretnénk (hello 4K), és ami külön jó benne: rá lehet pakolni az ún. pixel shadereket. Ezek olyan utófeldolgozók, amik a nyers képet mindenféle szempont szerint újraszámolják, leggyakrabban azért, hogy az eredeti CRT megjelenést közelítsék. Aki fogékony erre, rengeteg ilyen shadert talál, és belemerülhet az elektronsugár meg a maszk paramétereibe. *(Lásd. 1. kép).*

Irányítás: a libretro meghatároz egy saját kontroller típust, praktikus, ha erre le van képezve minden szükséges funkció. Van egy alapértelmezett kiosztás, de ez nagyon játékfüggő, hogy mennyire elég, elsősorban ezért kellett, hogy játékonként külön konfigurációs fájlt lehessen használni. A billentyűzet is használható, de jelenleg nincs mód változtatni a kiosztáson. Az Ext 1 joystick elérhető a numerikus billentyűzetről. Egy egyszerű autofire is rendelkezésre áll.

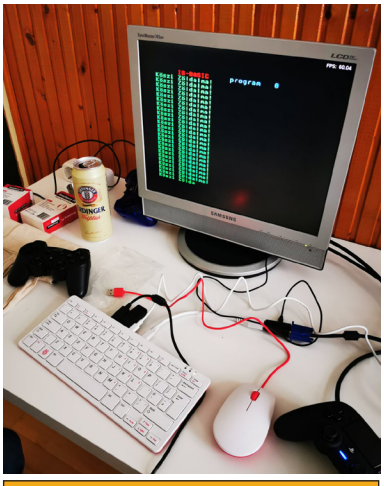
Programbetöltés: a libretro logikája nagymértékben a mainál pár generációval korábbi konzolokra alapoz, tehát nem elindítja az ember az emulált gépet, és utána választja ki a programot, hanem rögtön úgy indítja, hogy választ hozzá tartalmat. Ez lehet a szokásos lemezkép vagy magnós fájl, de hála az epfileionak (és a tvcfileionak), akár a konkrét programfájlt is kiválaszthatjuk. Az ep128emu-core memórieteszt után rögtön be is tölti és elindítja a legtöbbet. A magnós fájlokat sem annyira fájdalmas végigvárni, ha használja az ember a „fast-forward" funkciót, ami az ep128emu alt+w gyorsításának megfelelője. Itt érdemes kitérni arra, hogy a tartalom típusától függően, más-más emulátor konfiguráció indulhat. A DTF formátumú fájloknál egy Zozotools-szal ellátott Enterprise, TVC-s formátumú lemezeknél és .cas fájloknál a tvcemu, CPC lemezeknél és .cdt magnófájloknál a cpcemu, .tzx és nem Enterprise formátumú .tap fájloknál pedig a ZX emulátor. Ezek is használhatóak, de a CPC és a ZX résszel sokat nem foglalkoztam, ezek emulálhatók más core-ok felhasználásával is.

De a környezet nemcsak előnyökkel jár, hanem kompromisszumokkal is. Leegyszerűsítve, minden olyan funkció, ami ep128emu-ban külön ablakban nyílik, itt nem elérhető. Nincs debugger, nem tudunk hardver konfigurációt állítani, lemezt vagy szalagot cserélni menet közben. Ezek egy részét bele lehetne épp építeni, de az egyszerű használhatóság fontosabb volt most, meglátjuk hosszabb távon, van-e ezekre igény.

Ezen sorok írásakor a legegyszerűbben még mindig úgy lehet hozzájutni az ep128emu-core-hoz, hogy a githubról letölti az ember a megfelelőt, hozzá egy ROM csomagot (amit itt inkább BIOS néven emlegetnek), és egy telepített retroarch környezetbe beépíti. Ez remélhetőleg rövid távon annyiban fog javulni, hogy retroarch-on belül letölthető lesz az ep128emu-core. A ROM csomag az ismert szerzői jogi kérdések miatt nem tud ide bekerülni, így itt marad a saját beszerzés.

További tervek? Éppen lehetnek. Vannak játékok, amik még nem tudnak futni, tipikusan olyan magyar szöveges játékok, amik különféle ROM bővítőket igényelnek. Más gépek felé is lehetne terjeszkedni, a Primot vagy más hazai Z80 alapú gépet is hozzá lehetne csapni. Egy virtuális billentyűzet se lenne rossz. Mindezekhez képest viszont nekem hasznosabbnak tűnik inkább a tartalmat rendezni - de erről talán majd máskor.

- Zöldalma -

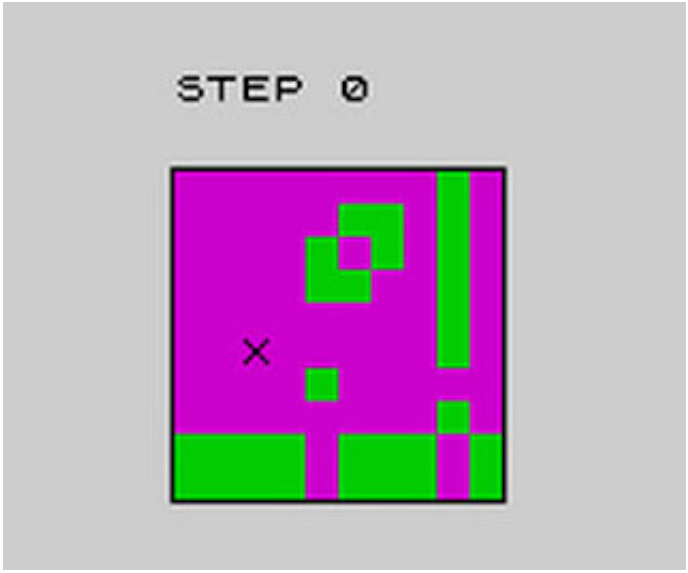


LETÖLTÉS:
https://github.com/libretro/ep128emu-core

Xorka - logikai játék a xor elvén - Enterprise változat elkészítése

Írta: Bodnár Tamás (Szipucsu)

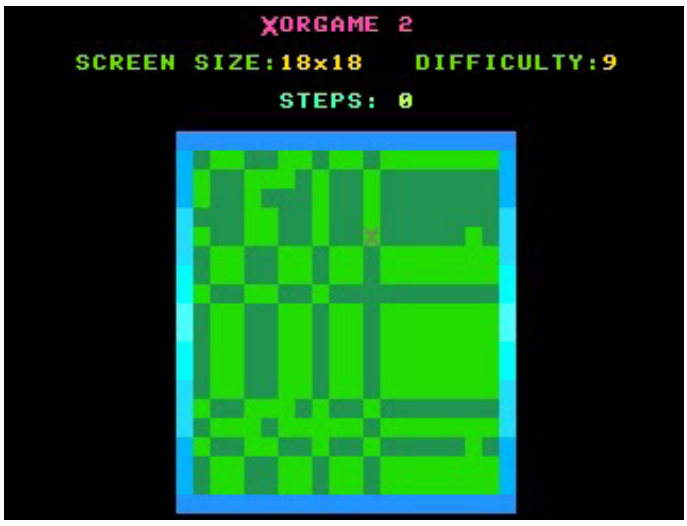
Spectrum számítógépre készítette el Alexander Zavgorodniy 2015-ben basic nyelven ezt a játékot, a Spectrum Computing oldalán is megtalálható.



A játék egy 10x10-es felületen játszódik, ahol a célunk a színes (a képen zöld) mezők eltüntetése. A pályán egy kurzort (X alakú) mozgathatunk. A tűzgomb megnyomására a gép egy vízszintes és egy függőleges vonalat rajzol úgy, hogy azok a kurzor pozíciójában metszik egymást. Pontosabban a vonalak kirajzolása közben a pályán oda, ahol nincs pályaelem, pályaelemet tesz, ahol pedig már van valami, onnan letörli. Tehát a xor elvén az adott vízszintes és függőleges pozícióban átállítja a 0 jelű elemeket 1 jelűvé, az 1 jelűeket pedig 0 jelűvé. A játék indításakor kirajzol véletlenszerűen vízszintes és függőleges vonalakat, ebből kell nekünk visszaállítani az üres pályát.

Az ep128.hu oldalon ennél tömörebben és érthetőbben van leírva a játék lényege: Feladatunk a megadott méretű játéktér megtisztítása a sötét mezőktől. Ennek egyetlen lehetséges módja, hogy a mozgatható célkereszt oszlopában és sorában a tűz gomb megnyomásával ellentétes színűre változtatjuk az összes mezőt (vagyis amelyik mező eddig sötét volt, most világos lesz, a világos mezőkből pedig sötét).

Az Enterprise változat a Xorgame 2 nevet kapta. Az első rész, a Xorgame Endi és IstvánV közös, gépi kódú programja, melyet Endi PC-re írt meg, majd István ebből elkészítette az Enterprise változatot. A Xorgame kapcsán botlottunk bele a Xorka nevű játékba, így merült fel, hogy ezt is meg lehetne írni Enterprise-ra.



A program működése:

A program az elején létrehoz egy kétdimenziós tömböt, ebben fogja tárolni az egész pályát. Játék közben ennek a tömbnek a megfelelő elemét fogja majd vizsgálni a program, illetve ennek a tömbnek az elemeit fogja módosítani xor végrehajtásakor (tűzgomb játék közben). Ahogy a képernyőre kiír ill. onnan letöröl pályaelemeket, azzal párhuzamosan a tömb tartalmát is módosítja. A tömb helyett kiolvashatná éppen a képernyőről is, hogy az adott pozícióban 0 vagy 1 van (vagyis van-e pályaelem vagy nincs), azonban a játék attribútum képernyőn fut, hogy több szín lehessen, grafikus képernyőről pedig nem tudunk karak-

tereket kiolvasni. (Elvileg megoldható lenne az is, hogy ne a karaktereket, hanem a színüket olvassa ki a program, ezzel még nem próbálkoztam.)

A játéknak két fő része van:

1. a kurzor (X jel) mozgatása a pályán
2. a xor művelet végrehajtása a tűzgombra (ez a XOR nevű eljárás a DEF XOR blokkban, amit a CALL XOR paranccsal hívunk meg mindig, amikor a játékos megnyomja a tűzgombot).

A pálya kirajzolása nem külön rész, hanem a program véletlenszerűen kiválaszt kurzorpozíciókat, és ott végrehajtja a xor műveletet (meghívja a XOR eljárást a CALL XOR-ral), ezzel rak a pályára vonalakat.

1. Az X mozgatása a pályán: figyeli a program, hogy a botkormány el van-e döntve valamelyik irányba. Ha igen, akkor megvizsgálja, hogy az adott irányba lehet-e tovább lépni, nem vagyunk-e már a pálya szélénél. Ha még lehet lépni, akkor a kurzort letörli a helyéről, az adott koordinátájához hozzáad egyet, majd kirajzolja az új helyén. Közben arra is figyelni kell, hogy milyen mezőre lép a kurzor, oda, ahol van pályaelem, vagy oda, ahol üres a pálya. Bárhova rakjuk ki a kurzort, az letörli, ami éppen ott van, tehát a pályaelem helyére és az üres helyre kirakva is teljesen ugyanúgy fog kinézni. Így nem lehetne eldönteni, a kurzor helyén pályaelem van-e vagy sem. Ezért az X kirakása előtt előbb meg kell vizsgálni, hogy az adott pozícióban mi van, és ha pályaelem, akkor inverz X-et kell kirakni. Az inverz X karakter definiálásáról a program elején gondoskodunk.
Akár meg lehetett volna csinálni, hogy az X villogjon, így látszana az is, ami alatta van. De azt is meg lehetett volna csinálni, hogy villogjon is, és a pályaelemhez is hasonuljon.

Fontos még, hogy a kurzor mozgatásakor, amikor a régi pozíciójából letörli és az új pozíciójába kirakja, akkor a régi pozíciójából ne csak letörölje, hanem az ott lévő pályaelemet rakja ki maga után. Ezt a tömbből kell kiolvasni. Ha nem tömböt használnánk, hanem a képernyőről olvasná ki, milyen karakter van adott pozícióban, akkor adott pozícióba lépés előtt egy stringben el kellene tárolnia az adott pozícióban lévő karaktert, utána kéne rálépnie, majd onnan ellépéskor kiírni a kurzor régi helyére, amit kiolvasott onnan odalépéskor.

2. A XOR művelet végrehajtása nem bonyolult. Két FOR ciklus kell hozzá. Az első a pálya felső szélétől a az alsó széléig, a második a pálya jobb szélétől a bal széléig rajzol vagy pályaelemet minden pozícióba, vagy vonalat, attól függően, hogy mi van ott: megnézi a program minden pozícióban a tömb adott elemét, és ha az 1 (pályaelem), akkor átírja 0-ra. Ha pedig 0, akkor átírja 1-re.

Amikor a keret kirajzolása után a pályát felépíti a gép véletlenszerű pozíciókba xorolással, akkor az 1-esek lerakásakor megnöveli 1-gyel a SUM változó értékét, mely kezdetben 0. Ha pedig egy 1-et átváltoztat 0-ra, akkor csökkenti 1-gyel a SUM értékét. Így tehát azt tárolja a SUM változóban, hogy jelenleg hány 1 (pályaelem) van a képernyőn. Erre azért van szükség, hogy játék során, ha sikerül az összes pályaelemet lepucolnunk, akkor a játszma befejeződhessen, és kiírja a gép a képernyőre a gratulációját. Figyelni kell arra is, hogy a XOR műveleteket végző eljárást a pálya kirajzolásakor, vagy pedig játék közben hívjuk-e meg. Erre azért van szükség, mert a pálya kirajzolásakor is előállhat az az állapot, hogy eltűnik, minden 1-es a pályáról, mert egymás után kétszer hajt végre a kirajzoló ugyanabban a pozícióban xor műveletet, így egyszer kirajzolja az 1-eseket, utána pedig letörli, és ebben a helyzetben a játék gratulációját fejezi ki, és vége a játéknak. Először bele is estem ebbe a hibába, hogy erre nem figyeltem, utólag tettem bele annak figyelését, hogy játékból vagy pedig a pálya kirajzolásából hívjuk-e meg a XOR eljárást, az előbbi esetben a GAME változó értéke 1, utóbbi esetben 0.

XOR művelet legelején a kurzor pozíciójában lévő karaktert átváltoztatja az ellentétére. Az eredeti program írója úgy oldotta meg, hogy ahol a két vonal metszi egymást, ott az maradjon a pályaelem, ami előtte volt, így én is ezt az elvet követtem. Elméletileg az se lenne baj, ha a vonalak metszéspontjában ellenkezőjére váltana a karakter, mert az egyik vonal felülírja azt, a másik vonal pedig ismét felülírja.

Enterprise az FPGA rendszeren

Idén **Kyp** publikálta az Enterprise FPGA core csomagokat. Jó hír, hogy Rampa kifejlesztette a A Dave chip hang részét és a WD1770 emulációt is. Nem kis munkájuk volt ebben. Szerencsére az Enterprise Fórumon kérdéseikre szinte azonnal kaptak választ és segítséget a magyar felhasználóktól és **Bruce Tanner**-től valamint **Gflorez**-től aki **Ron**-nal együtt béta tesztelő. A verzió még nem végleges, az EXDOS mag a következő verzióba kerül majd.

Az újdonság az, hogy a következő Core verzió kezeli majd a floppy képeket a következő verzióban.

Az eddig elkészült verziók innen tölthetők le:
http://retrowiki.es/viewtopic.php?f=107&t=200037978&p=200154146#p200154146

MiSTer:
enterprise_20220501.rbf.zip

NeptUNO:
enterprise_20220501.np1.zip

Unamiga Reloaded:
enterprise_20220501.ua2.zip

Tutus

SymbiFace 3 újdonságok

Legyünk őszinték! Az SymbiFace 3 kártya megjelenésekor, vásárlásakor kicsit felemás érzésünk volt. Sokat tud ez a kártya, de drivereket, programokat nem igen kaptunk ehhez, kicsit olyan érzésünk volt, mintha ezt a felhasználónak kellene megoldania. Nos, nem mindenki programozó. Így néhány egyszerű beállítás után a legtöbb felhasználó RAM és ROM bővítőként tudta használni, kaptunk hozzá egy MP3 lejátszót és chat-et.
Itt egyből felmerült az, ha már van chat, akkor nyilván az internet kapcsolatot használ, esetleg ki lehet használni ezt a kártyát jobban ezen a téren.

Idén az SF3 kártyával is sokan foglalkoztak, így több fejlesztés is elkészült.

Fájlkezelés

Az SF3 kártyán lévő USB pen drive-on található programok valamint a lemezműveletek eddig nem voltak elérhetőek. Ennek megoldására a Geco néhány egyszerű fájlkezelő parancsot implementált az SF3BOOT.ROM-ban, amelyek ugyanazokat a műveleteket hajtják végre, mint az EXDOS.ROM-ban ismertek:

:sf3 cd, :sf3 dir, :sf3 md, :sf3 rd, :sf3 del

Teszteltünk sok programot és a rendszer nagyon megbízható... kivétel: SymbOS.
(halkan hozzáteszem, hogy az idei új SymbOS kiadás szinte használhatatlan Enterprise gépen. Nem nagyon értjük ezt, hiszen pontosan Hansék és Prodatron között legalább a SymbOS–SF3 kapcsolatnak kellene jól működnie, de ez sem...)

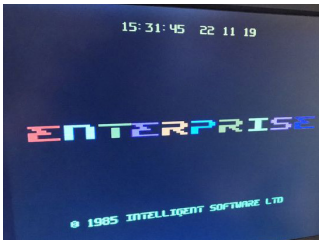
FTP

Ezt a fejlesztést Hansnak köszönhetjük. Ki gondolná, hogy itt van előttem egy 37 éves 8 bites gép és ezen most nekiállunk FTP-zni?

A :def_dev_ftpd parancsot kiadva engedélyezzük az FTP kapcsolatot. Az FTP kapcsolat beállításait az SF3_EP.INI fájlban kell megadni a következőképpen:

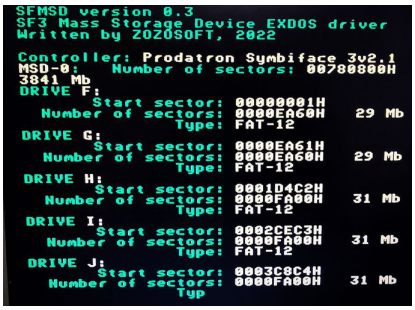
FTP_ENA = ON
FTP_PORT = 21
FTP_IP = FTP szerver IP címe
FTP_USER = FTP szerver felhasználónév
FTP_PASS = FTP szerver jelszó
FTP_PATH = FTP szerver útvonal
Innentől már ugyanúgy használhatjuk a fájlkezelésnél már említett parancsokat, pl. :sf3 dir. Valamint az FTP szerverről programokat, játékokat is be tudunk tölteni!

RTC



Szintén az SF3 leírásában ott volt az dátum/óra megjelenítési funkció, de azt, hogy hogyan jelenítjük meg, azt nem tudhattuk. Végül az SF3BOOT.ROM-ba Geco oldotta ezt meg a következőképpen: ha van Zozotools ROM a rendszerben (mely tartalmazza az óra megjelenítését és az RTC kezelését), akkor a Zozotools erre vonatkozó parancsai megjelenítik a dátumot és pontos időt, valamint kezelik az SF3 RTC-jét.

SFMSD



Zozo írt egy SFMSD.ROM-ot.
Ha egy EXDOS.ROM-mal együtt beállítjuk az SF3 ROM konfigurációba, akkor a korábbi EP-s IDE vagy SD megoldásokhoz hasonlóan érhetjük el az SF3-ra csatlakoztatott USB meghajtót. Ehhez az SD-hez hasonló módon írjuk ki a szokásos VHD fájlt az USB-re.
Ha szét van partícionálva az USB, akkor maga az SF3 az első partíciót látja. Ez viszont minden állítással szemben, nem csak FAT32 lehet, hanem FAT12 vagy FAT16 is. Működnek rajta a Geco féle SF3 fájlkezelő parancsok.
Az FDISK legfrissebb 0.8 változata tartalmazza már az SFMSD kezelését is. Az SFMSD helyes működéséhez legalább 2022 november 21-ei DFU-ra kell frissíteni az SF3-at.

Tutus

ENTERPRISE
COMPUTERS

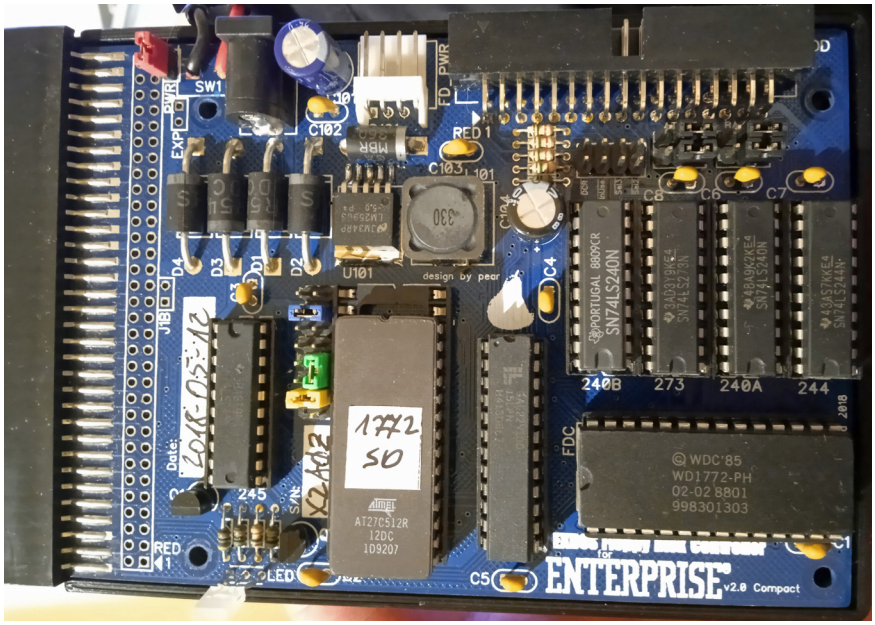
Az Enterprise számítógéppel kapcsolatos dolgok egy helyen!

?

HAMAROSAN!

Enteprise 128-hoz készült Pear-féle EXDOS v2.0 compact floppyvezérlő tuningja

Írta: Dr. Orvos Gábor (Dr.OG)



Még 2018-ban jutottam hozzá egy ilyen kártyához, amit rögvest be is üzemeltem, és elégedetten használtam, de csak rövid ideig. Nem sokkal később sikerült ugyanis szereznem egy SD-kártya olvasót, azóta vele volt (többé-kevésbé) rendszeres használatban a gép. Néhány hete azonban ismét kezembe került ez a floppyillesztő, és elgondolkodtam rajta, mihez is kezdjek vele.

Akár el is adhattam volna, de a termék Wikijét (https://wiki.enterpriseforever.com/index.php?title=EXDOS_by_pear) alaposan áttanulmányozva döbbentem rá, hogy korábban a kártya képességeinek csak a minimumát használtam ki. A benne lévő 64kB-os EPROM-nak (gondolom kompatibilitási okokból) ugyanis csak az első 32kB-ja tartalmazott adatot (EXDOS 1.4 és IS-DOS 1.0). A ROM-ot nagyobbra (256kB, vagy akár 512kB - lásd később) cserélve azt teleírhatjuk hasznos felhasználói- (pl. EPDOS, ASMON, HEASS, különféle programnyelvek) vagy szórakoztató játékprogramokkal (pl. Bricky Prise, Buzzsaw, Wriggler), melyek kis keresés után ROM formátumban is fellelhetők a https://www.enterpriseforever.com/ -on. Az így megpakolt ROM-ot azonban a rendszer a gyári EXOS 2.1 esetén sajnos nem fogja látni, jobban mondva csak az első 32kB-ot észleli. Erre megoldást jelent a korábban már említett SD-kártyaolvasó használata, ezen már a Zozó-féle memória gyorsteszt kapott helyet, mely feltérképezi az összes rendelkezésre álló ROM-ot és RAM-ot. A másik megoldás a gép alaplapjának módosítása, és a 2.4-es EXOS beégetése egy 64kB-os EPROM-ba, erről részletes leírás itt található: http://www.ep128.hu/Ep_Util/Exos.htm

Amit a ROM állományokról tudni érdemes:

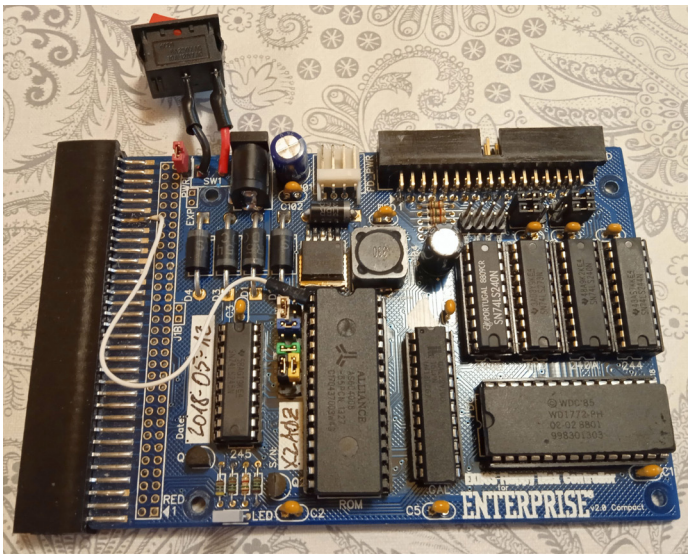
*) A nem teljes 16kB vagy többszörösei méretű fájlokat hexaeditorral ki kell egészíteni (én FF hexa karaktereket használtam), hogy pontosan 16.384, 32.768 vagy 65.536 bájt méretűek legyenek.

*) Bizonyos játékok EXOS-kompatibilisek, és más bővítményekhez hasonlóan futtathatók ROM-ból. Ilyen a régiek közül pl. a PASIANS, a CYRUS, de Zozó is csinált pár ilyet, pl. WRIG, EAT, BZS.

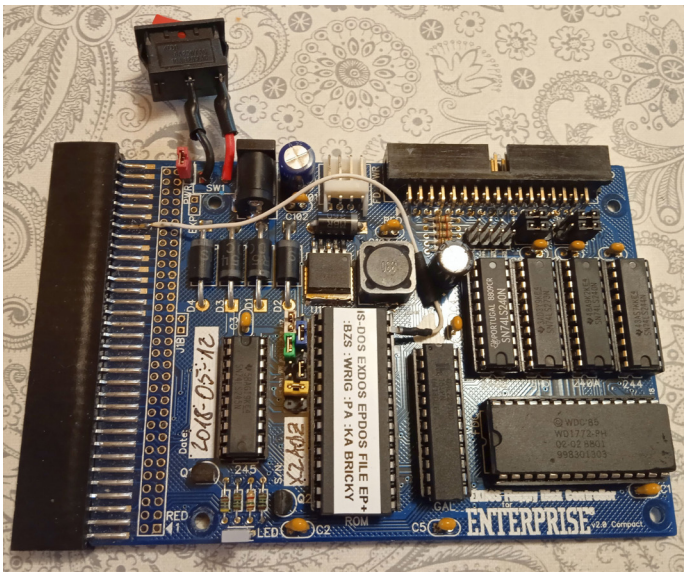
*) Mások nem jelennek meg a :HELP listáján, de LOAD „FILENAME:"-vel betölthetők és futtathatók, ilyen pl. a HAMIKA, BRUCE, NODES.

) Vannak olyan játékok, amik direkt cartridge ROM-ként történő használatra alakítottak át, pl. BRICKY, TCAVE. Ezek esetén a :HELP listájában ROMFS szerepel, melynek tartalma a :ROMDIR paranccsal listázható, ezt követően a LOAD „ROM:" indítja az első file-t, a LOAD „ROM:FILENAME" pedig a listáról választott ‚FILENAME' nevű fájlt. Vagy a ROMFS első fájlja az ENTERPRISE logónál hosszabban nyomva tartott 1-es billentyűre is elindul.

Miután összeválogattuk a kívánt tartalmú és méretű ROM-okat, azokat binárisként fűzzük össze (Windows alatt parancssorból - CMD): copy /b ELSO.ROM + MASODIK.ROM + ... + UTOLSO.ROM FUZOTT.BIN, majd ezt égessük be a megfelelő méretű (max. 256/512kB) (E)EPROM-ba.



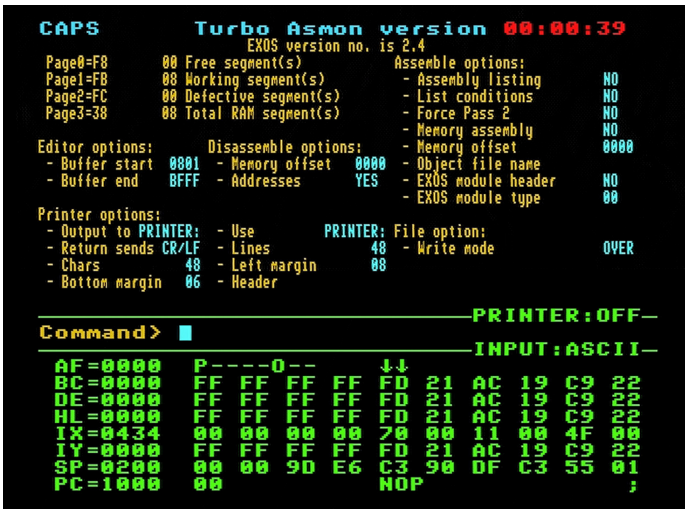
Én W27E040-et használtam erre a célra, abba sok fér, és elektronikusan törölhető, nem kell UV-lámpával vacakolni.

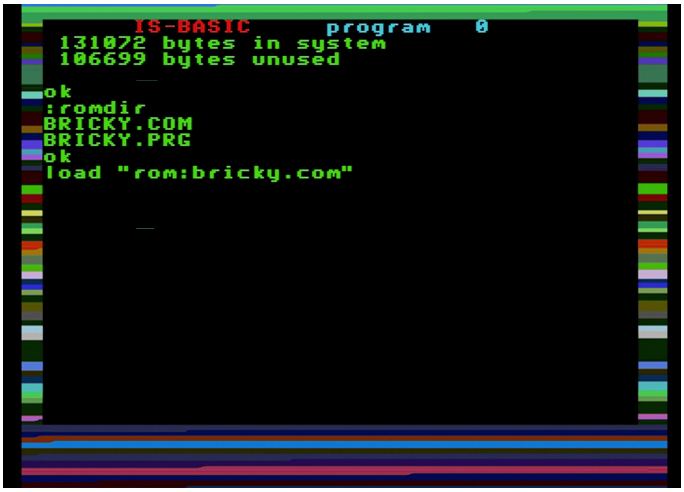


Fontos, hogy a ROM-foglalat melletti színes jumpereket is pakoljuk át az alkalmazott memória IC-nek megfelelően (lásd fenti Wiki link).

Miután ezzel megvoltam, elkezdett érdekelni a floppyillesztő RAM-bővítőként való alkalmazása.
Rendeltem hát Kínából két AS6C4008 SRAM IC-t, ezeket próbáltam a fenti kártyával, sikertelenül. Természetesen a jumpereket átpakoltam a Wikin jelzetteknek megfelelően, az IC pedig helyes orientációval került a ROM helyére. Már akkor felmerült bennem, hogy valószínűleg mást is kellene mókolni a kártyán, arra tippeltem, hogy a GAL tartalmát is át kellene írni, de erről semmi konkrét infót nem találtam. Pedig a fentebb linkelt honlap implicit módon azt sugallja, hogy működnie kellene. Próbáltam SD-kártya illesztővel és anélkül, utóbbin a legfrissebb, 0.6-os verziójú FW-vel. Persze azt sem zártam ki, hogy a RAM-ok simán rosszak, vagy hamisítványok, de gondoltam akkor látnia kellene a memóriatesztnek, de csak az alap 128kB-ot jelezte ki.
Az IC-gyűjteményemből előkerestem egy hazai beszerzésű (Kontel) HM62256ALP-t, a kártyát megfelelően

bejumpereltem, beraktam, és azzal sem ment. Ugyanúgy nem adott hibajelzést, de nem is látta a rendszer a plusz 32kB-ot. A ROM-ot visszarakva és visszajumperelve továbbra is szépen működött a kártya, csak RAM-bővítőként nem akart menni. Multiméterrel megméregettem az élcsatlakozó és az IC-foglalat közti sávokat, azokat rendben lévőnek találtam. Kínomban kivettem a 240B jelzésű IC-t is (ez opcionális), és visszajumpereltem AT-ra a kártyát (eddig XT-n volt), valamint kipróbáltam az ‚A'dvanced test-et is, de az eredmény ugyanaz maradt (vagyis semmi). Mivel itt teljesen elakadtam, az enterprise-forever-en kérdezősködve Zozosoft, alias Németh Zoltán sietett a segítségemre, aki írt két programot a kártya tesztelésére, egyet olvasáshoz (ROM)? egyet pedig íráshoz (RAM). Szkóp vagy logikai analizátor hiányában multiméterrel mértem a CE jelet (foglalat #22-es lába), mely alapból 4,17V-ot mutatott, ez olvasáskor 3,88V-ra esett, íráskor lényegében nem változott (4,15V). Pedig írásnál ugyanúgy kellett volna változnia, de RAM esetén a CE mindig 1-ben volt. Úgy tűnt, a GAL-ban lévő program csak olvasásra engedélyezte az SRAM IC-t, valószínűleg egy korábbi program volt benne, ami csak ROM-ot kezelt (a kártyámon 2018 májusi dátum szerepel). Zozo saját privát levelezését átnézve kiderült, hogy 2020 zűrzavaros tavaszán (a COVID első hulláma idején) egy másik fórumtaggal már egyszer megfutották ezeket a köröket, csak az eredmény nem lett publikálva, ezért feledésbe merült. Viszont így legalább megkerült a RAM-ot támogató .jed fájl mellett az 512kB-os memóriákat címző változat is, ehhez viszont minimális





hardveres módosításra is szükség van: a teljes, 512kB-os bővítőkapacitás kihasználásához annyit kell tenni, hogy az 512kB verziójú .jed fájlt égetjük a GAL-ba (én ehhez egy TL866 II Plus programozót használtam, köszönet érte Óré Andrásnak), és a rendszerbusz csatlakozó B27-es lábát (ami a billentyűzet felől nézve, a gép előtt ülve a felső sorban hátulról a hetedik), amin az A18-as címvonal érhető el, összekötjük a RAM IC 1-es lábával (szintén A18). A címvezeték IC felőli végét precíziós foglalat lábával illesztettem az integrált áramkör kihajtott lábára, zsugorcsővel izoláltan, plusz találtam egy megfelelő vastagságú blankolt szigetelődarabot, amit rá lehet húzni pluszban (főleg az 1-es láb esetén jön jól), vagy ha nem akarom rádugni az A18-at semmire, akkor megakadályozza, hogy véletlenül rövidre zárjon valamit.
Ez működik (E)EPROM-mal is: a W27E040-esbe is lehet 512kB-ot írni, de arra figyeljünk, hogy ott a #31-es láb az A18. Nyilván az lenne a legjobb, ha a J2-es tüskesor 12-es pinjére vezethetnénk az A18 jelet, ahogy a v2.1 esetén alapból van, de az fixen földre lett bekötve, és a vezetősáv sajnos nem a forrasztási oldalon halad, így elnyírni kb. esélytelen, az alkatrész oldal zsúfoltsága miatt.

Kérdés, hogy van-e gyakorlati jelentősége a RAM-bővítésnek? Aki SymbOS-t akar komolyabban használni, annak amúgy is erősen ajánlott a SymbiFace3 kártya beszerzése, de legalább létrehozhatunk egy jó nagy RAMDISK-et, és létezik néhány játék is (főként Noel Persa átiratok), melyek kihasználják a bővített memóriát:

Castlevania
Legalább 128 kB RAM kell a futtatáshoz.
128KB esetén a rendszerszegmens felülíródik, nincs töltési/mentési opció és végződés (end sequence), és 4 zene eltávolításra került (LPT van a helyén)
144KB-nál hiányzik a végződés és 4 zene eltávolítva (LPT miatt)
160KB nincs végződés
176KB-nál és fölötte minden opció elérhető

The Next War
Legalább 128KB memóriát igényel.
A játék zenéje csak legalább 256KB memóriával rendelkező gép esetén töltődik be, ebben az esetben 11 másodperc ismétlődik, a teljes zene betöltéséhez 2,5 MB szükséges.

Where Time Stood Still
128KB RAM esetén a rendszerszegmens és a 0-ás lap is felülíródik
144KB RAM esetén a rendszerszegmens felülíródik
160KB RAM és felette, a rendszerszegmens, és 0-ás lap érintetlen marad.

Microprose Soccer Reloaded
128K+-on van csak játékállás mentés és töltés.

The Hobbit
640KB+-nál betölti az egész játékot a memóriába, és nincs utántöltögetés.

Wolfenstein 2004
Játék közben zene csak 128KB+-on van.

Norht & South
Legalább 256Kb RAM-mal rendelkező EP esetén az egész program a memóriába töltődik.

Bobby Bearing

1986 - The Edge

Írta: Kiss László
(Lacika)

Az Edge Games (eredetileg Softek néven futott) a nyolcvanas években összehozott pár nem túl emlékezetes, de üzletileg sikeres játékot a kor 8 bites gépeire. Ezen platformok kihalása után a cég még megpróbálta régi sikereit új platformokon is elsütni, már mérsékeltebb sikerrel. A jogtulajdonos - Tim Langdell - ezért új, jövedelmezőnek tűnő bevételi forrás után nézett: Mindenkit beperelt vagy perrel fenyegetett aki és ami az „Edge" szót használja a nevében. A jó Tim karrierje „csúcspontján az Electroninc Arts-nál is bepróbálkozott, a Mirror's Edge miatt. Mint kiderült, egy mamutvállalat már túl nagy falat egy ilyen pitiáner alaknak, ezért a pofára esés után kénytelen volt újra elkezdeni dolgozni. Ennek terméke volt a iOS-platformokat megcélzó EDGEBobby2 című játékuk. A játék hirdetéseiben „a roppant nagy sikernek örvendő Bobby Bearing folytatása" jellemzést biggyesztették hozzá. Ebből arra lehet következtetni, hogy Tim Langdell - tévesen - azt hiszi, ez cége egyik legjobb játéka... Egyébként a játékot nem ő írta, pikáns történeti szál, hogy 1990-ben Tim cégének két volt programozója pert nyert ellene - azért mentek perre, mert Langdell egyszerűen nem fizette ki a munkájukat.

A Bobby Bearing sem valami eredeti ötlet: egy izometrikus nézetű akció-kalandjáték. Ilyen és hasonló Knight Lore-klónok tucatszám jelentek meg a ,80-as évek közepén, számtalan játékot találhatunk (elég, ha csak a Batman vagy Head over Heels játékokra gondolunk), amik szórakoztatóbbak az Edge játékánál. Anno csak annyit sikerült kideríteni a játékról, hogy valami idegesítő, golyóval gurulós őrületről van szó.

A játék főhőse teljesen egyedi módon egy csapágygolyó(!), aki Bobby névre hallgat. Szüleivel és testvéreivel Technofear-ben élnek, sok más acél „élőlénnyel"(!?) együtt. Unokatestvérük és Bobby négy testvére egy nap nem hallgattak az intő szóra, és elkóboroltak. A gonosz csapágyak elcsalták az ifjú golyókat. Bobby feladata, hogy 4 testvérét, és unokatestvérét megtalálja, és hazakísérje őket otthonukba! Testvéreinket, majd unokatestvérünket, sorban egymás után kell hazakísérni, megadott időn belül, (sorban: Barnaby, Bert, Bungo, Boogle, Osborne.). A jobb alsó sarokban látszik, kit kell éppen hazavinni (a többi szereplőt addig kár is keresni). A bal alsó sarokban látszik, mennyi időnk van még, hogy a soron következő rokont hazakísérjük a start-képernyőn látható lyukba. A „hazakísérés" alatt azt kell érteni, hogy egy másik golyót (ugyanúgy néznek ki, mint mi), az akadályok között ellökdössük(!) egészen hazáig, majd végül nekünk is be kell gurulni a lyukba. Szerencsére a képernyő széléről nem tudunk leesni a mélybe, egy képernyőről, csak a szélén, nyilakkal jelzet helyeken tudunk átjutni.

Főhősünk - golyó jellégéből adódóan - csak gurulni tud, önerőből még pattanni sem képes. Emelkedőn tudunk felfelé gurulni, de a legkisebb „lépcsőn" sem tudunk feljutni (ezek egyirányú átjárók). Dolgunk megnehezítéséről le-fel mozgó paltformok (valamelyik liftként is használható), kapcsolók / rejtett kapcsolók, gonosz fekete golyók hivatottak gondoskodni. Bár főhősünk acélból van, a lefelé mozgó platók kilapíthatják (rokonaink keményebb fémből vannak, ők nem lapulnak ki). A golyók ugyan nem tudnak bántani, viszont bosszantó és időtrabló módon össze-vissza lökdösnek, egyetlen szerencsénk, hogy mi valamivel gyorsabbak vagyunk.

Technofear alapterülete meglehetősen nagy, ugyanakkor változatosnak kevésbé nevezhető: a képernyők többsége egyetlen „csatorna", amik szinte ugyanúgy néznek ki, csak a színükben különböznek, megkönnyítve az eltévedést. A kevéske szobában, ahol valamit „kombinálni kell", a kapcsolók jelentik az egyetlen lehetőséget az interakcióra.

Ha letelik az idő, az adott rokont már nem tudjuk megmenteni, az idő újraindulása után a sorban következő megmentendő szereplőt kell keresnünk.

Irányítás:

Billentyűzet (W-P - jobbra fel, A-L - balra le, X, V, N - jobbra le, C, B, M - balra fel).

Q - időveszteség árán abba a pozícióba rak, ahonnan bejöttünk a képernyőre,

Az ENTERPRISE verzióban KEMPSTON-t választva EXT1, SINCLAIR-t választva a beépített botkormánnyal játszhatunk. Működik a billentyűzet is, de pause (SPACE billentyű) nincs.

A menüben a 6-7 billentyűkkel tudjuk az irányítást kiválasztani, a 0 megnyomásával indíthatjuk a játékot.

The Sword of Ianna

Írta: Kiss László
(Lacika)

Réges-régen a világot a Káosz istene uralta, amiből persze semmi jó nem sült ki. Ianna - egy másik istenség - azonban olyan kardot készíttetett, ami képes legyőzni a gonoszság teremtményeit. A kardot egy nagy harcos, Tukaram érdemelte ki, aki hosszú harc után végre békét hozott a világra. De a gonosz nem nyugszik, néhány évszázaddal később (no igen, halhatatlanoknak könnyű türelmesnek lenni...) újra támadnak! Tukaram persze rég halott, de a kardja fellelhető! Jarkum-nak - a „szerencsés" kiválasztott harcosnak - tehát rendet kell teremtenie a gonosz istenség csúf teremtményei között! Ilyen előzményekkel indul a RetroWorks parádésan kivitelezett játéka. A program - játékmenetben (és színvonalban) - alig tagadható módon hasonlít a Prince of Persia-ra. Hősünk ugyan ezúttal nem egy ifjú herceg, hanem egy nagydarab barbár harcos, de faladata hasonló. A játék folyamán előttünk kibontakozó (minimalista) történet közepette kell eljutnunk - a szükséges tárgyak megszerzésével - a nehézségi szint kijáratáig, ami a következő szintre visz. Főhősünk, rengeteg - gondosan animált - mozdulatra képes: Termetét meghazudtoló ügyességgel kapaszkodik, mászik, ugrál. De szükség esetén egyetlen gombnyomással előkapja farzsebéből jól megtermett kardját, hogy miszlikje vágja a rá támadó lényeket. Ebből talán már nem sejthető, hogy főhősünknek a kardján, és akrobatikus képességein kívül az eszét is próbára kell tenni, hogy elhárítsa az elé tornyosuló akadályokat:

Az akadályok között a szörnyek a legnyilvánvalóbbak (mondhatni „szemetszúróak"...). Megadott pozícióban járőröznek, ha egy vonalba kerülünk velük és felénk fordulnak, észrevesznek. Ilyenkor ők is kardot rántanak, és ránk támadnak. Az egyszerű(?) csontvázak nem sok vizet zavarnak, de van pár kellemetlen figura, aki - ha rosszkor pislogunk - lekaszabolnak, legalábbis, amíg kevesebb a tapasztalatunk. Merthogy a játékban a harcok folyamán tapasztalatot szerzünk és megfelelő mennyiségű győzedelmes harc után szintet lépünk! Összesen 8 szint van, nagyobb szinteken nagyobb sebzést osztunk ki és jobban bírjuk a gyűrődést. Persze szörnyekből is vannak alacsonyabb, magasabb szintűek... Figyeljünk erre, mert ugyan egy egyes szintű csontváz nem sok gondot okoz, de egy kettes szintű semmi perc alatt miszlikbe aprít! Nincs mese, itt ki kell ismernünk a szörnyek harcmodorát! Az egyszer legyőzött ellenfelektől szerencsére végleg megszabadulunk, legközelebb, amikor arra járunk, már nem jelennek meg. Itt érdemes pár szót mondani a harcrendszerről: a minket ért sérülések (ide értve a harcban szerzett sérüléseket, vagy amit azért szerünk, mert nagyobb magasságból esünk le) energiánkat csökkentik, a „műszerfalon", a portrénk melletti piros sáv ezt jelzi. Mellette a világoskék sáv a megszerzett tapasztalati pontot jelzi. Ha ez eléri a maximumot, szintet lépünk, amit az L (level) jelzés számlál. A mi paramétereink mellett, jobbra a képernyőn rajtunk kívül tartózkodó szörnyek (maximum 2) energiája és szintje látható. Ha elfogy az energiánk (meghalunk), az utolsó „ellenőrző ponttól" kezdjük újra a kalandot. A program nem számlál életet.

Terepakadályból a legkézenfekvőbb a játéktér oldalán lévő kapu (az első szinten sárga), melyet a hozzájuk tartozó kapcsoló (egy nagy kar) elmozdításával, vagy az ajtót őrző szörny legyőzésével lehet kinyitni. A faajtókkal könnyebb dolgunk van: ezeket egyetlen kardcsapással összetörhetjük. Tipp: ha egy ellenségre ráeresztünk egy kapcsolóval működtethető ajtót, kilapul!

A szemközti falon látható ajtók ugyanúgy működnek, mit a képernyő oldalán lévők. Ha kinyitottuk, a ‚fel' iránnyal tudunk rajta keresztülmenni. Némelyik ajtó mellett viszont különböző színű zárat látunk, ezek nem kapcsolókkal működnek, meg kell találni a megfelelő színű kulcsot hozzá. Ha nálunk van a megfelelő kulcs, közeledtünkre az ajtó magától kinyílik. (Sokféle alakú ajtó van, sajnos nem minden ajtóról látszik egyértelműen, hogy tényleg az.)

A földön heverő ládák, edények - vagy éppen golyók - akadályt képeznek ugyan előttünk, de egy kardsuhintással ezeket is széttörhetjük. Általában valamilyen tárgyat is rejtenek, ritkábban kapcsolóként is szolgálnak!

A karókra, tűzre különösen figyeljünk, ha ezekbe zuhanunk vagy lépünk, azonnal meghalunk!

Nyugodtan nevezhetjük magukat a platókat is akadálynak! Itt bizony lesz bőben ugrálás. Valahol helyben tudunk felugorni és felkapaszkodni, valahol csak nekifutásból tudunk átugorni (úgy nagyobbat ugrunk, mint helyből). Azon platók szélét, amiben meg tudunk kapaszkodni, a program más színnel

jelöli meg. Pl. az első szinten jól látható módon fehérrel. Ezen platók széléről - háttal állva - le is tudunk mászni, elkerülendő a nagyobb magasságból való leesést. Magyarán minden olyan képességünkre szükségünk lesz, mit Perzsia Hercegének... Tipp: ha egy „szakadék" túl széles és nem tudjuk átugorni, még lehet, hogy meg tudunk a plató szélébe kapaszkodni. Ugrás közben tehát mindig tartsuk nyomva a ‚fel' irányt.
Csapdák: felénk guruló sziklák, hólavina, stb. Akkor indulnak el, ha a pálya egy bizonyos pontjára értünk. Ha nem tudjuk időben kikerülni, az halálos. Józserivel csak akkor van esélyünk kikerülni, ha már ismerjük a csapdát, számítunk rá, így egy kis Rick Dangerous beütést is kap a játékmenet...
Említettük már a tárgyakat, nézzük meg, hogy milyen tárgyakat találhatunk:
Étel: különböző formában tálalva találhatjuk. Nem tudjuk felvenni, a helyszínen elfogyasztva azonnal pótolja elvesztett energiánkat.
Gyógyital: felvehetjük, majd a kívánt pillanatban elfogyasztva szintén energiánkat pótolja.
Kulcsok: a zárral ellátott ajtókat nyithatjuk ki vele.
Fegyver. Nem lesz sok, összesen háromféle fegyvert használhatunk.
Küldetésekhez kapcsolódó tárgyak:
Befőttesüveg: tökéletesen alkalmas arra, hogy a megfelelő kútnál szenteltvízzel töltsük meg.
Egyéb - áldozati oltárokra helyezendő - kincsek.
A „műszerfalon" a bal oldali tárgylistába kerülnek a kulcsok, gyógyitalok, kincsek. Középen látható az aktuális fegyver. A tárgyakkal a ‚H' megnyomása után megjelenő menüben tudunk bűvészkedni. (jobbra - balra, választás a tárgyak között, tűz - tárgy használata, le - fegyver választása). Ugyanebben a menüben az ‚X' megnyomásával szakíthatjuk meg a játékot, vagy a ‚H' ismételt megnyomásával léphetünk vissza a játékba. Megjegyzendő: a gyógyitalt nem érdemes tartogatni a következő szintre, mert nem marad nálunk!
Szépen animált főhősünk meglehetősen sokféle akrobatamutatványra képes. Az, hogy milyen mozdulatot tud, attól függ, kezében van-e a fegyver, vagy sincs. A ‚tűz 2' billentyűvel lehet a fegyvert előkapni, elrakni. Ha valahol leesünk, automatikusan elrakjuk a kezünkben lévő fegyvert. Fegyver nélküli „mutatványok":
jobbra - balra: séta,
tűz + jobbra - balra: futás (a tűz gombot nem kell nyomva tartani),
tűz: kapcsoló használata,
le: guggolás, tárgy felvétele,
fel: ugrás, kapaszkodás, belépés ajtón,
fel + jobbra - balra: ugrás (futásból, vagy állva).
Fegyverrel a kézben:
jobbra - balra: előre, hátra,
tűz fel: támadás felülről,
tűz + jobbra + balra: kétféle támadás csípőből,
tűz + le: támadás leguggolva,
fel: védés.
A játék összesen 8 hatalmas szinten játszódik. Már volt szó róla, hogy egyetlen életünk van. Az első szint is hatalmas, de ott legalább nem nagyon lehet eltévedni. A többi szint viszont kész labirintus, tele kapcsolókkal és zárt ajtókkal! Szerencsére a program annyira azért nem szívatja játékosokat, de a labirintusokat fel kell térképezni, a szörnyek viselkedését ki kell ismerni. Hogy a végcél egyetlen életünk birtokában elérhető legyen, a program egy ötletes megoldást tartogat: Minden szint elején (kivéve persze az elsőt) megkapjuk annak kódját! A megfelelő kódot a menüben (PASSWORD menüpontban) megadva az adott szinttől indíthatjuk a játékot - kímélendő a játékosok idegrendszerét és hajzatát...
Akik pedig a nyolcadik szint végén legyőzik a Káosz istenét is, tovább bolyonghatnak egy titkos pályán (ez lenne a 9. szint, aminek Nostalgium a neve). Ez régi spanyol játékok helyszíneit eleveníti meg (A RetroWorks spanyol csapat): Camelot Warriors, Sir Fred, Game Over, Livingstone Supongo, Army Moves, Fred, Abu Simbel Profanation, Freddy Hardest.
A kivitelezésről csak szuperlatívuszokban lehet beszélni. Az animáció, grafika gondosan megrajzolt és változatos. A labirintusok hatalmasak, könnyű bennük eltévedni (Aki Rastan-féle futkorászásra számít, csalódni fog...) A játékmenetet jól megkomponált zenék kísérik. Kell is a programnak a memória. A játék - méretére tekintettel - eredetileg lemezen jelent meg, ebből készült magnós (tap) verzió, bár ezt igazi gépen csak a türelmesebbek erőltessék.
Nehézségi szint kódok:
Level 2: D4545E558D
Level 3: 97574C54DE
Level 4: 96564354DD
Level 5: 91516654FC
Level 6: B050785719
Level 7: B35364571F
Level 8: A253645730
Level 9: 0CAFECAFE0
A játék először ZX Spectrum-ra és MSX2-re készült el 2017-ben. A CPC verzió 2020-ban jelent meg cartridge formájában, ezt konvertálta Geco. A program meglehetősen méretes, így a CPC verzió - a CPC-s lemezek kapacitása miatt - eleve cartridge-be lett tervezve, vagyis nincs belőle utántöltős változat (mint amilyen a Spectrum verzió). Enterprise-on a program egyben töltődik a memóriába, a RAM igénye 576KB. Sajnos utántöltős verziót nehéz lenne belőle csinálni, mert szint adatok vegyítve vannak más adatokkal, ráadásul egy-egy szinthez 128KB-nál is több memória szükséges. A CPC (és Enterprise) verzió 4 színű képernyőn fut, a Spectrum verzióval azonos felbontásban és képernyőméretben. Ezt, illetve a sztereó zenét leszámítva a játék azonos a Spectrum változattal. Változások az Enterprise-verzióban:
Extra színek kerültek a programba (ilyen pl. a játék végén a scroll-ozó szöveg egy színátmenetből sejlik elő), a legnagyobb módosítás a vízen lévő tükörkép, és a hullámzás: ez LPT módosítással lett megvalósítva, 2 verzió is bekerült, ami a program indulásakor véletlenszerűen kerül kiválasztásra, majd váltják egymást a menübe visszatéréskor.
A megfelelő botkormány adapterrel két gombos botkormányt is kezel a program.
1 tűzgombos botkormánynál ‚tűz+fel': kard kiránt, ‚le': kard eltesz.
Ha a ‚tűz 2', vagy a billentyűzet kard gombja (alapértelmezés szerint jobb SHIFT) lett megnyomva, akkor az egy gombos joystick kard elrakás funkció inaktiválódik, a menübe lépéskor aktiválódik újra.
Cheat beépítve: a C és H billentyű lenyomása után nem fogy az energiánk.
Irányítás:
EXT1, EXT2 botkormány vagy billentyűzet (O, P, Q, A, SPACE)
A billentyűzet a beépített botkormányra is definiálható.
tűz 2: jobb SHIFT
HOLD / PAUSE vagy H: szünet / kilépés

IS-FORTH - 7. rész

Intelligent Software - 1985
rendszerbővítő, FORTH programozási nyelv

Példa string változó inputjára: Az EXPECT és a SPAN használatával definiálhatunk egy olyan szót, ami bevesz egy maximum 255 hosszú szöveget, letárolja a PAD-ben, vagy bármilyen más, 255 hosszúra deklarált változóban. Példa:

```
: $? DUP 1+ 255 EXPECT SPAN C@ SWAP C! ;
```

Ez a szó beveszi a stringet, elteszi a hosszt, és mindezt tárolja azon a címen, amit előzőleg a verembe tettük. Példa:

```
PAD $?
```

A gép ezután várakozik az input szövegre. Miután megadtuk a stringet, gépeljük be a következőt:

```
PAD $.
```

Erre a beadott szöveg kiíródik.

String tárolás és string felvétel
A $@ operátor felvesz egy stringet és elhelyez a PAD-ben, aminek a címét a veremben találjuk. A $! operátor a PAD-ben lévő stringet teszi be egy megadott string változóba. Példa:

```
„ Fred" $CONSTANT NEV$
NEV$ $@ $.
```

A fentiekben deklaráltuk a NEV$-t és betettük a PAD-be. Ezután kiíratjuk a NEV$-t.

Szöveg input belsőleg definiált szóból

```
„ (idézőjel)
```

Ez az operátor nagyon hasznos string változók inicializálásához. Eredményképp az idézőjelben lévő szövegek bekerülnek a PAD-be. Gyakran használatos operátorral együtt és az EXOS-nak való paraméter átadáskor. Lényeges, hogy az első karakter és az első idézőjel közt egy szóköznek kell lennie. Például:

```
80 $VARIABLE JANI$
„ En vagyok a Jani!" JANI$ $!
```

Stringek összefűzése
Nagyon gyakran van szükség két string összekapcsolására. A CONCAT szó a veremben tárolt kezdőcímű karakterláncot hozzákapcsolja a PAD-ben lévő szöveg végéhez és az így létrejött szöveg kezdőcímét a verembe teszi. Például:

```
„ alacsony" $CONSTANT MA$
„ magas-"
MA$ CONCAT $.
```

String összehasonlítás
A $<> operátor megvizsgálja, hogy két string különbözik-e egymástól. Általában ez egy IF ... ELSE ... THEN struktúrában használatos. Billentyűzet input ellenőrizhető ezzel az összehasonlító operátorral.

Példaprogram a string-változók használatára

```
„ Szia! Mi a csaladi neved?" $CONSTANT H1$
„ Es a keresztneved?" $CONSTANT H2$
 „ Orulok, hogy megismerkedtunk „ $CONSTANT VALASZ$
„ „ $CONSTANT URES$
20 $VARIABLE CSALNEV$
20 $VARIABLE KERNEV$
: $? DUP 1+ 255 EXPECT SPAN C@ SWAP C! ;
: SZIA CR H1$ $. CR CSALNEV$ $? CR H2$ $. CR
  KERNEV$ $? CSALNEV$ $@ URES$ CONCAT
  KERNEV$ CONCAT CR VALASZ$ $. PAD $. CR ;
```

A programot a SZIA szóval hívhatjuk meg.

Byte-sorozatokat kezelő, kényelmes alapszavak:
TYPE (cím hossz - - - -)
Kiírja a kimeneti eszközre (alapértelmezés szerint a képernyőre) a cím-en lévő karaktersorozatot, a megadott hosszon.
CMOVE (honnan-cím hova-cím hossz - - - -)
Byte-sorozat mozgatása egyik címről a másikra. Az átvitel az alacsonyabb címeken kezdődik.
FILL (cím hossz karakter - - - -) A memóriát a cím-től kezdve hossz számú karakterrel tölti fel. Ha a hossz 0, nem történik semmi.
ERASE (cím hossz - - - -) A memóriát a cím-től kezdve hosz számú 0 byte-tal tölti fel. Ha a hossz 0, nem történik semmi.

A WORD

```
WORD ( c - - - - )
```

(ejtsd: vörd, jelentése: szó) tördeli a befolyamot a vermen megadott határolókarakter alapján. Azaz: a WORD beolvassa a befolyam karaktereit, egészen a határolóig, átállítja a megfelelő rendszerváltozókat (hogy a legközelebbi WORD ott kezdje az olvasást, ahol ez befejezte), a beolvasott szöveget pedig FORTH-füzér formájában a HERE címre teszi. A WORD a sor végét mindenképpen határolónak érzi.
Egy példa a WORD alkalmazására a LEVEL szó, amelyet így lehet majd használni:

```
LEVEL KATINAK
DRAGA KATI, HALALOMIG IMADLAK
PITYU
```

avagy

```
LEVEL ELEONORANAK
DRAGA ELEONORA, HALALOMIG IMADLAK
PITYU
```

Lássuk a megvalósulást:

```
: LEVEL
  32
  WORD


  CR CR
  ." DRAGA"
  HERE COUNT
  3 -
  TYPE
  ." HALALOMIG IMADLAK"
  CR 20 SPACES
  ." PITYU" CR
;

( a szóköz kódja)
( beolvassa a LEVEL után megadott nevet )
( az első szóközig, FORTH-fűzér formában )
( leteszi HERE címre )
( a fontos vallomásokat új sorban kezdjük )

( a vermen a név címe és hossza )
( „nak", „nek" nem kell )
```

A Kísérletező Olvasó esetleg kipróbálja a következőt:

```
32 WORD ABRAKADABRA HERE COUNT TYPE
```

és azt várja, hogy az ABRAKADABRA szót kapja válaszul, hiszen azt olvasta be a WORD. Ehelyett a TYPE szót fogja kapni, mivel az interpreter a parancssorok elemzésére maga is a WORD-öt használja, így, mire a TYPE végrehajtódik, az utolsónak végrehajtott WORD-őt rámolta a HERE címre.

Honnan ismerős? (WORD-del működő alapszavak)
A WORD-ben az a szokatlan, hogy nemcsak a veremről vesz adatot, hanem maga mögül, a befolyamból is kivesz egy szövegrészt és felhasználja. A jelenséggel már találkoztunk egyes FORTH alapszavaknál; ezek a szavak mind a WORD-öt hívják.
A legegyszerűbb ilyen alapszó a (, amelynek a forrásával is megismerkedünk:

```
: ( 41 WORD ;
```

(41 a záró zárójel kódja). A nyitó zárójel beolvassa, amit maga mögött talál, egészen a záró zárójelig (vagy a sor végéig). A WORD úgy állítja át a megfelelő rendszerváltozókat, hogy az interpreter legközelebb a záró zárójel utánról olvasson; így a befolyanmak a két zárójel közötti része egyszerűen nem rúg labdába.
A másik alapszó, egy csökkent értékű változata:

```
: ." 34 WORD HERE COUNT TYPE ;
```

(34 az idézőjel kódja). A WORD beolvassa a befolyamot az idézőjelig és FORTH-füzér formájában elhelyezi a HERE címen; innen egy COUNT TYPE sorozattal kiíratható.
Mennyivel tud ennél többet az igazi szó? A különbség akkor látszik, mikor a szót definíció belsejében akarjuk használni. Az igazi a definícióban megadott szöveget írja ki, az itteni változat a definiált szó végrehajtódásakor olvassa el a befolyamból a kiírandó szöveget.
WORD-del olvassa be az elfelejtendő szó nevét a FORGET is. A WORD munkál minden olyan szó belsejében, amellyel más szavakat definiálhatunk:

```
VARIABLE CONSTANT ;
```

Pontosabban, a CREATE szó hozza létre az új szótárelemet, a CREATE pedig a WORD-del olvassa be a létrehozandó szótárelem nevét.

3.7. Számkonverziók

Megfelelő típushoz megfelelő kiíratás
A . .R D. egyöntetűen egy-két számjegy-előállító alapszóra épül. Ugyanezekkel az alapszavakkal mi is bármikor írhatunk hasonló szavakat, amelyek valamilyen formában kiírják a vermen levő, adott típusú értéket.
A vermen levő bináris érték kiírható számjegyekké konvertálását a

```
<#
```

szó kezdi el és a

```
#>
```

fejezi be.
A konvertált jegyeket előállító szavakat (#, #S, HOLD, SIGN) a <# hívása után, a #> hívása előtt futtathatjuk.

A konverzió során a memóriában létrejön a konvertált jegyekből és kiegészítő jelekből álló füzér, amelynek címét és hosszát a #> a veremre teszi (így TYPE-pal kiírathatjuk). Aki már próbált egy számot az egyik számrendszerből a másikba átírni, az tudja, hogy ez csak visszafelé, az utolsó jegytől kezdve megy. A konvertáló szavak is „visszafelé

hordanak": a kiíratandó karakterekből álló füzér hátulról kezdve készül el. A füzér utolsó (elsőnek elkészülő) byteja a pad első byte-ja. (Ne csodálkozzunk tehát, ha a konvertáló szavak elrontják a pad elején tárolt adatainkat!) A

```
# ( ud1 - - - - ud2 )
```

szó beilleszti a füzérbe a (hátulról) következő jegyet. Működése: az ud1 előjel nélküli duplaszót elosztja a BASE tartalmával. Az ud2 hányadost a veremre teszi, hogy a konverzió folytatódhasson; kikalkulálja a maradékból a megfelelő számrendszerbeli számjegy kódját és beírja a füzérbe. A

```
#S
```

végigkonvertálja a vermen levő előjel nélküli duplaszót, annyi jegyet tesz a már előállítottak elé, amennyi szükséges (tehát értéktelen, vezető 0-kat nem). A vermen egy duplaszavas 0 marad. A #S forrása egyébként:

```
: #S ( ud - - - - dupla-0 )
  BEGIN
    # ( egy jegy )
   OVER OVER OR 0 = ( a vermen heverő duplaszó 0? )
  UNTIL ( ha nem 0 volt az utolsó duplaszó, folytatjuk )
;
```

A duplaszót, amelyben 3 konverzió részeredményeit tartottuk, a #> dobja el a veremről. A #, így a #S is a BASE-nek megfelelő számrendszerbeli jegyeket állít elő. A

```
HOLD ( c - - - - )
```

szó beírja a kapott c karaktert az eddig konvertált jegyek elé. Ha például duplaszavas, előjel nélküli számot akarunk úgy kiíratni, hogy az utolsó két jegy tizedesjegynek látszódjék:

```
: 2TJE
  <#
    # #
    46 HOLD
    #S
  #>
TYPE
;
          ( ud - - - - )
( konverzió eleje )
( előállt a két utolsó jegy )
( tizedespont )
( további jegyek )
( konverzió vége )
```

Vagy, ha magunk szeretnénk a vermen megadni, hogy hány tizedesjegy legyen:

```
: TJE
  <#
    0 DO # LOOP
    46 HOLD
    #S
  #>
  TYPE
;
          ( ud tizedesjegyek - - - - )
          ( konverzió eleje )
          ( tizedesjegyek )
          ( tizedespont )
```

Ha előjel nélküli, egyszavas értéket akarunk kiíratni, az sem gond. Könnyen csinálhatunk belőle kétszavasat, hiszen (pozitív számról lévén szó) a duplaszavas érték magasabb helyi értékű szava 0, egyszerűen ezt kell csak a veremre tenni. Erre szolgál az

```
U. (u - - - - )
```

Szó. Forrása:

```
: U. ( u - - - - )
0 ( a duplaszó magas helyiértékű szava )
<# #S #> ( a szükséges számjegyek előálltak )
TYPE SPACE
;
```

Többnyire azonban előjeles értékekkel dolgozunk. Ehhez meg kell ismerkednünk a

```
SIGN ( n d - - - - d )
```

szóval, amely szintén a <# és #> között használatos. A SIGN az a (azaz a verem harmadik eleme) előjelétől függően tesz a konvertált sorozatba mínuszjelet vagy sem. Példa a . szó lehetséges forrása:

```
: . ( n - - - - )
DUP ABS ( a felső példánynak nincs előjele )
0 ( könnyű belőle előjel nélküli duplaszót csinálni )
<# #S SIGN #>
TYPE SPACE
;
```

Végül egy szellemes példa Leo Brodie-tól, a STARTING FORTH című könyvből: Tegyük fel, hogy a vermen heverő duplaszó a másodpercek száma, és mi óra: perc :másodperc formában akarjuk kiíratni ezt az időtartamot. Lavíroznunk kell a 10-es és a 6-os számrendszer között.

```
: SEXTAL ( áttérés 6-os számrendszerre )
6 BASE ! ;

: OO
# ( egy decimális jel előállítása )
SEXTAL
# (egy „szextális jel előállítása )
DECIMAL
58 HOLD ( kettőspont )
;
```

A fenti programocska, ha jobban belegondolunk, a 60-nal való osztás maradékát írja ki, mégpedig tízes számrendszerben. (A hányadost pedig továbbadja a vermen.) Így a kívánt szót már könnyen megírhatjuk:

```
: SEC ( ud - - - - )
<# OO OO #S #> TYPE SPACE
;
```

4. A virtuális memória

Alapok

Mire ide eljut, az Olvasó bizonyára rengeteg programot írt már. Nagy részüket kernyőre szerkesztette, hogy javíthassa őket. A kernyők az ún. virtuális memóriában vannak. A virtuális memória a memória kiegészítése lemezzel, lemezes kernyőfile-lal vagy kazettával. (A legutóbbi kicsit nehézkes, mert a FORTH virtuális memória kezelési elvei közvetlen hozzáférést igényelnek).

A virtuális szó itt látszólagosat jelent. A virtuális memória elnevezés azt tükrözi, hogy a virtuális memória blokkjait - bár adathordozón vannak - programozáskor úgy látjuk, mintha a memóriában lennének. Mégpedig azért látjuk úgy, mert a

BLOCK (blokksorszám - - - - memóriacím)

szó gondoskodik róla, hogy a blokk, amelyre hivatkozunk, beolvasódjék a memóriába, ha eddig nem volt ott; a visszakapott címtől kezdve megtaláljuk a memóriában a blokk tartalmát. A blokk itt az olvasás-írás fizikai egységét jelenti, általában (lemezről lévén szó) egy szektort. A BLOCK szó:

beolvassa a memóriába a megadott sorszámú blokkot és visszaadja a beolvasott blokk címét;

vagy, ha a blokk egy előző beolvasás következtében már a memóriában van, akkor megkeresi és megadja a címét.

Az utóbbi esetben nem fordulunk a lemezhez; időt takarítunk meg anélkül, hogy erre programíráskor ügyelnünk kéne.

A virtuális memória blokkjait az interpreter a memória ún. blokkpuffereiben tartja. A BLOCK végignézi, hogy nincs-e a keresett blokk valamelyik blokkpufferben; ha megtalálta, akkor a blokkpuffer címét kapjuk, ha nem, akkor beolvas, de melyik pufferbe? A válasz egyszerű, ha van üres blokkpuffer. A blokkpufferek azonban kevesen vannak a virtuális memória blokkjaihoz képest. Többnyire valamelyik foglalt blokkpuffert kell felülírni. Mi történik ilyenkor a puffert foglaló blokkal? Elvész a tartalma, vagy visszaíródik az adathordozóra?

Ezt mi magunk döntjük el. Ha nem teszünk semmit, a puffer tartalma nem íródik vissza, tehát csak olvassuk a virtuális memóriát. Azt, hogy egy adott blokkot majd visszsza is akarunk íratni, a blokkra való hivatkozás (BLOCK) után az

UPDATE (- - - -)

szóval közöljük. Ettől az adott blokk „update-elt", vagyis módosított lesz. A szóhasználat egy kicsit félrevezető; a blokkpuffer tartalmát majd ezután fogjuk módosítani, az adathordozó blokkjait pedig még később, mikor a pufferre szükség van, vagy mikor minden módosított puffert visszaírunk.

Az UPDATE mindig arra a blokkra vonatkozik, amelyre a legutoljára hivatkoztunk a BLOCK szóval.

(Ez egyszerűen úgy történik, hogy a BLOCK leteszi a PREV rendszerváltozóba a blokk számát, az UPDATE pedig ott találja meg).

Az összes módosított blokkot a

FLUSH (- - - -)

szóval írathatjuk ki (például, mikor felállunk a gép mellől). Ha viszont vissza szeretnénk csinálni mindazt, amit a blokkpufferekben elkevertünk, az EMPTY-BUFFERS (----) után tiszta lappal indulhatunk; az interpreter üresnek tekinti az összes blokkpuffert, anélkül, hogy egyet is visszaírna (vagy a blokkpufferek tényleges tartalmát megváltoztatná. Az

EMPTY-BUFFERS

nem a puffereket törli, hanem a pufferek nyilvántartását írja át.)

Hogy a virtuális memóriában tárolt szövegeket ugyanúgy végrehajtathatjuk az interpreterrel, mint a billentyűzetről beírtakat, az (többek között) azon múlik, hogy az interpreternek mindig a WORD adja át a következő interpretálandó szót. A WORD a BLK rendszerváltozó tartalmától függően működik:

Ha a BLK 0, akkor a billentyűzetről beolvasott parancssor következő szavát veszi elő. A parancssor tárolására szolgáló memóriaterület, a parancspuffer címe a TIB rendszerváltozóban van.

Ha a BLK nem 0, akkor a virtuális memória egy blokkjának számát tartalmazza; ebből a blokkból veszi a következő szót a WORD.

itt érdemes megemlíteni, hogy a parancsuffer címét a

TIB (- - - - cím)

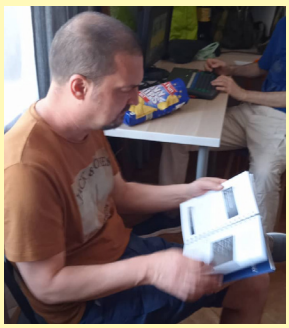
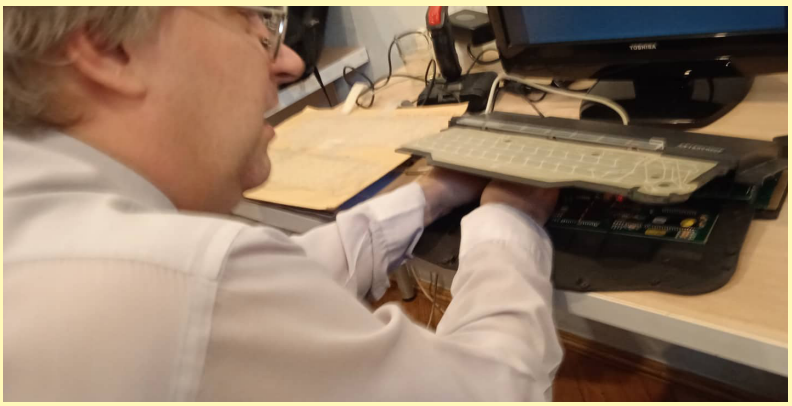
rendzserváltozó trtalmazza. A FORTH editorok gyakran az aktuális kernyő soraira vonatkozó műveletekből állnak. Az aktuális kernyő számát az SCR rendszerváltozó tartalmazza. Például a LIST is aktuálissá teszi a kilistázott kernyőt, azaz a számát elhelyezi az SCR változóban.

A BLOCK, az UPDATE, a FLUSH és az EMPTY-BUFFERS a virtuális memóriakezelés alapjai.

(Folytatjuk!)

ENTERPRISE Klub - 2022.

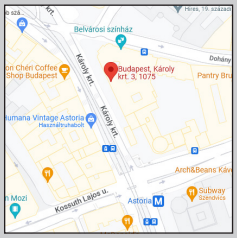



ENTERPRISE KLUB

Egy évben 8 alkalommal

ASTORIA
1074 Budapest,
Károly krt. 3/A.,
4. emelet, Genf terem
87-es kapucsengő
14 órától 19 óráig



További információ: www.enterpriseklub.hu

Ha te is szeretnél Az ENTERPRESS Magazin szerkesztője lenni, küldj cikket, játékleírást, játékismertetőt, vagy bármit amely az Enterprise számítógéppel kapcsolatos!

A cikkeket erre az e-mail címre küldheted:

info@enterpress.news.hu

ENTERPRISE FOREVER

https://enterpriseforever.com

ENTERPRESS Magazin - 2022/3-6. május-december
Főszerkesztő: Matusa István
Szerkesztőségi főmunkatárs: Németh Zoltán (Zozosoft)
A csapat: geco, Povi, Kiss László, SzörG, szipucsu, lgb, Bakó Róbert, Tamási Istvánné, Vaczkó Károly
Design, nyomdai előkészítés: Matusa István
Weboldal: https://enterpress.news.hu
E-mail: info@enterpress.news.hu
A lap időszakosan - korlátozott példányszámban - nyomtatott formátumban és elektronikus formában is megjelenik.
ENTERPRESS e-magazinok:
https://enterpress.news.hu/index.php/magazin