ENTERPRESS

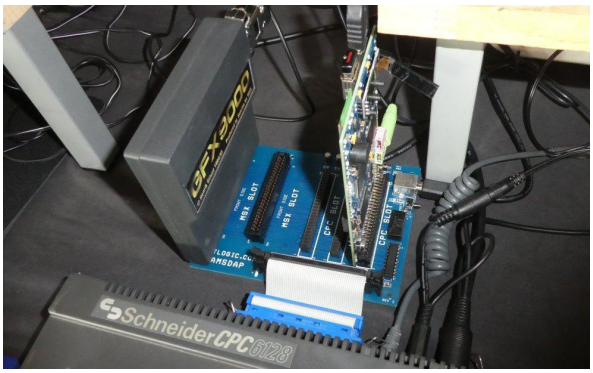
Magazin az ENTERPRISE felhasználóknak

2019/3–4. május–augusztus

A VL1772-ről újra

Kérem a következőt!

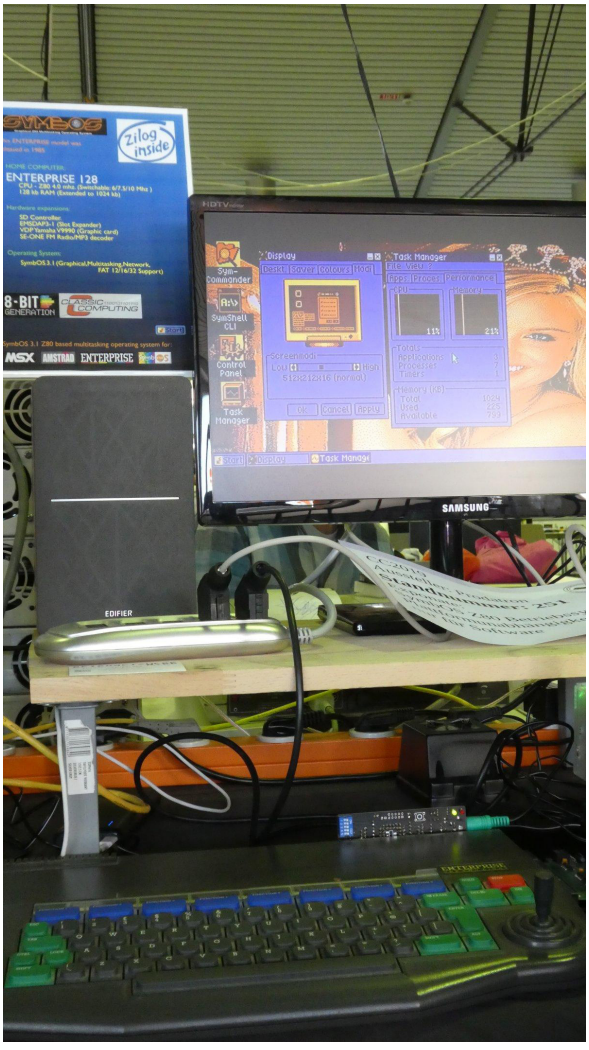
avagy NextBASIC újdonságok

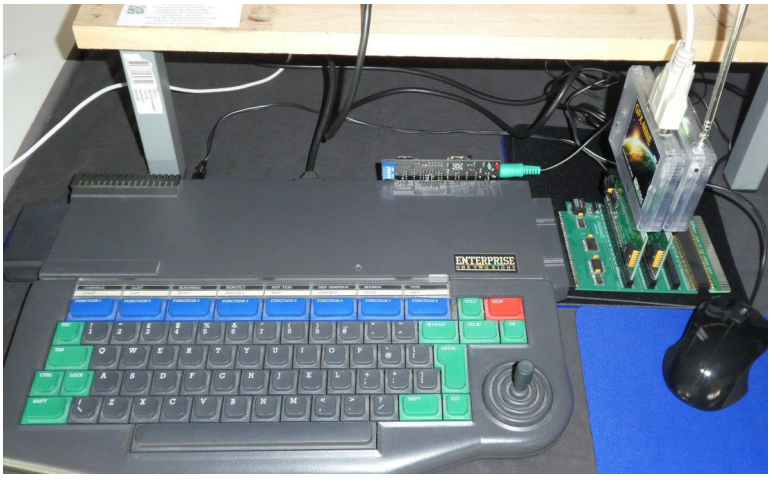


Podatron és a SymbOS

a Classic Computing 2019 kiállításon

2019. 09. 20-22.
BörnekenHalle
Lehre, Germany

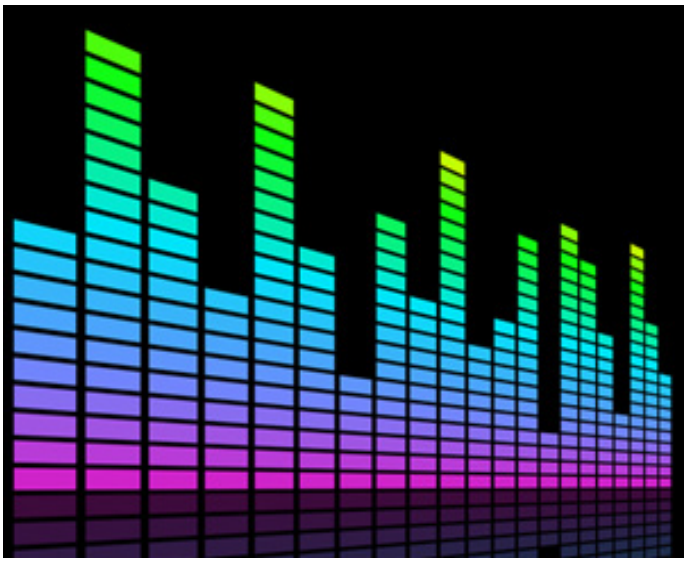




A SOUND utasítás rejtelmei IV.

A STYLE paraméterről bővebben, 2. felvonás

Írta: Bodnár Tamás
(Szipucsu)



Az előző számban végignéztük a STYLE paraméter egycsatornás effektjeit (a háromféle torzítást, és a zajcsatorna hangzásának megváltoztatását a polinomszámláló értékének beállításával), és a kétcsatornás effektek egy részét (a zajcsatorna magasságának változtatását egy másik csatorna által és a gyűrűmodulációt). Már csak kettő kétcsatornás effekt maradt hátra, a felül- és az aluláteresztő szűrő.

A kétcsatornás effektek:

5. Felüláteresztő szűrő:

A felüláteresztő szűrő a használatát tekintve nagyon hasonló a gyűrűmodulációhoz: Valamelyik csatornán ki kell adnunk a SOUND paramétereként a STYLE 64-et, és a nála eggyel nagyobb számú csatorna frekvenciája módosítani fogja az előbbi csatorna hangzását. A gyűrűmoduláció lényege is az, hogy a csatorna hangzását egy másik csatorna frekvenciája módosítja, csak a gyűrűmodulációnál ez a hatás erőteljesebb, a felüláteresztő szűrőnél pedig lágyabb. Fizikailag a felüláteresztő szűrő azt jelenti, hogy egy adott frekvencia alatt nem engedi át a hanghullámokat, tehát „felül ereszt át".
Tehát egy SOUND utasításban a STYLE 64 paraméterrel érhetjük el, hogy a nála eggyel nagyobb számú csatorna felüláteresztő szűrőként funkcionáljon.

1. Tehát, ha pl. a SOURCE 0 csatornán adjuk ki a STYLE 64 paramétert, akkor a SOURCE 1 csatornán szóló hang fogja a szűrő frekvenciáját adni, például:

```
SOUND STYLE 64,SYNC 1,PITCH 41
SOUND SOURCE 1,SYNC 1,PITCH 48,LEFT 0,RIGHT 0
```

(Mivel alapértelmezésként a SOURCE 0 csatornát használja a SOUND, ezért ez itt az első SOUND utasításból ki is maradhat. Az órajelcsatorna hangerejét akár 0-ra is állíthatjuk a LEFT 0,RIGHT 0 paraméterekkel, így is kifejti hatását a másik csatornára.)

2. Ha a SOURCE 1 csatornán adjuk ki a STYLE 64 paramétert, akkor az a SOURCE 2 csatorna frekvenciáját használja szűrőnek:

```
SOUND SOURCE 1,STYLE 64,SYNC 1,PITCH 41
SOUND SOURCE 2,SYNC 1,PITCH 48,LEFT 0,RIGHT 0
```

3. Ha a SOURCE 2 csatornán adjuk ki a STYLE 64 paramétert, akkor a 3-as, a zajcsatorna lesz az órajelcsatorna. Ilyenkor kicsit érdekesebb a helyzet, mert a zajcsatorna frekvenciája alapból 31.25 KHz, így az eredeti hang eléggé sajátos lesz:

```
SOUND SOURCE 2,STYLE 64,SYNC 1,PITCH 41
SOUND SOURCE 3,SYNC 1,LEFT 0,RIGHT 0
```

De természetesen ilyenkor is megadhatunk a zajcsatornának más frekvenciát is, ha a STYLE 1 / 2 / 3 paraméterrel egy négyszögjelcsatornát használjuk fel. Így más lesz a hatása a szűrésnek. Például, ha a 2-es csatorna adja a frekvenciát is a zajcsatornának:

```
SOUND SOURCE 2,STYLE 64,SYNC 1,PITCH 41
SOUND SOURCE 3,SYNC 1,LEFT 0,RIGHT 0,STYLE 2
```

4. Ha a zajcsatornán (SOURCE 3) adjuk ki a STYLE 64 paramétert, akkor az a 0-ás csatornát fogja használni szűrőnek. Azonban, mivel a zajcsatorna frekvenciája alapból jóval magasabb, mint négyszögjelcsatornáké, így ennek a szűrésnek nem igazán lesz hallható eredménye:

```
SOUND SOURCE 3,STYLE 64,SYNC 1
SOUND SYNC 1,PITCH 48,LEFT 0,RIGHT 0
```
(Mivel a SOURCE 0 az alapértelmezett, itt is kihagyható.)

A zajcsatornának természetesen megadhatunk egy másik frekvenciát is, ha a frekvenciát valamelyik négyszögjelcsatorna állítja elő számára. Ez a négyszögjelcsatorna lehetne az is, amelyet a szűrőzéshez is használunk, de ez nem eredményez hangot! (Ha tehát a fenti példában

STYLE 64 helyett STYLE 65-öt adunk meg - 64+1, ahol a 64 a szűrőre vonatkozik, az 1 pedig a 0-ás csatornára, amely a frekvenciát adja -, az egyáltalán nem fog szólni.) De természetesen a zajcsatorna frekvenciáját egy másik négyszögcsatorna is szolgáltathatja, így már van hang. Azonban így már csak egyetlen csatorna marad szabadon további hangoknak.

A szűrt csatornán lehet torzítás is, és/vagy a szűrőként használt csatornán is beállíthatunk torzítást, ez is kihatással van a végeredményre.

A felüláteresztő szűrő néhány felhasználási területe

Dallamhoz jól használható a felüláteresztő szűrő. Egyszólamú zene hangzását érdekesebbé tehetjük, ha a szűrőt bekapcsoljuk, és az órajelcsatornán egy kvinttel (7 PITCH értékkel) vagy egy oktávval (12 PITCH értékkel), vagy egy duodecimával (oktáv+kvint, azaz 19 PITCH értékkel) nagyobb hangmagasságot adunk meg. Ilyenkor érdemes az órajelcsatorna hangerejét is nullára állítani. Gyűrűmodulációnál általában akkor érünk el jobb hangzást, ha az órajelcsatorna hangereje is viszonylag nagy, felüláteresztő szűrőnél azonban a kisebb vagy a 0 hangerő eredményez általában jobb hangzást. Próbáljuk ki a következő programot! A 140-es sorban a G+7 helyett megadhatunk G+12-t vagy G+19-et, más lesz a hangzás:

```
100 ENVELOPE NUMBER 1;0,63,63,1;0,-20,-10,3;0,
      -43,-53,50
110 FOR I=1 TO 37
120 READ G,H
130 SOUND PITCH G,DURATION H,STYLE 64,
      ENVELOPE 1,SYNC 1
140 SOUND PITCH G+7,DURATION H,LEFT 0,
      RIGHT 0,SYNC 1,SOURCE 1
150 NEXT
160 DATA 44,20,37,10,39,10,41,10,42,10,44,20,37,20,37,20
170 DATA 46,20,42,10,44,10,46,10,48,10,49,20,37,20,37,20
180 DATA 42,20,44,10,42,10,41,10,39,10,41,20
190 DATA 42,10,41,10,39,10,37,10,39,20
200 DATA 41,10,39,10,37,10,36,10,37,20
210 DATA 37,10,39,10,41,10,37,10,39,40
```

Kétszólamú zenénél is használhatunk felüláteresztő szűrőt, de inkább csak akkor, ha a két szólam között egy tercnyi a különbség (zenei szakkifejezéssel ezt tercpárhuzamnak nevezzük). Váltakozva kistercek és nagytercek is lehetnek. Ha a magasabb hangot adó szólamot még egy oktávval feljebb helyezzük, hangerejét nullára csökkentjük, és az alsóbb szólamon felüláteresztő szűrőt állítunk be hozzá, sajátos, kellemes hangzást kapunk!

```
100 ENVELOPE NUMBER 1;0,63,63,1;0,-20,-10,3;0,
      -43,-53,50
110 FOR I=1 TO 30
120 READ G,H,J
130 LET G=G-10:LET H=H-10
140 SOUND PITCH G,DURATION J,STYLE 64,
      ENVELOPE 1,SYNC 1
150 SOUND PITCH H+12,DURATION J,LEFT 0,
      RIGHT 0,SYNC 1,SOURCE 1
160 NEXT
170 DATA 41,44,20,41,44,20,42,46,20,44,48,20,44,48,20,
      42,46,20,41,44,20,39,42,20,37,41,20,37,41,20,39,42,
      20,41,44,20,41,44,30,39,42,10,39,42,40
180 DATA 37,41,20,37,41,20,39,42,20,41,44,20,41,44,20,
      39,42,20,37,41,20,36,39,20,34,37,20,34,37,20,36,39,
      20,37,41,20,36,39,30,34,37,10,34,37,40
```

(A program csak illusztráció!)

A gyűrűmodulációnál már előző számunkban bemutattuk, hogy ha alacsony torzítással és egy kvint hangkülönbséggel összekapcsolunk két csatornát, akkor a torzított gitárhanghoz nagyon hasonló hangot kapunk. A felüláteresztő szűrőt is használhatjuk hasonlóra, tehát alacsony torzítással és kvint hangkülönbséggel két csatorna között bekapcsoljuk a szűrőt. Gyűrűmodulációnál itt mindkét csatorna hangerejét egyformának érdemes beállítani, szűrőnél pedig az órajelcsatornát érdemes halkabbra venni vagy teljesen el is némítani. Így a gitár hangja kicsit lágyabb, inkább Dire Straits-esebb lesz, míg a gyűrűmodulációs gitárhang keményebb, inkább Joan Jettesebb. A következő program ízelítőt ad a felüláteresztő szűrős gitárhangból. A 170-es sorban a STYLE 80 (64 + 16) azt jelenti, hogy a felüláteresztő szűrőt és az alacsony torzítást egyszerre használjuk:

```
100 ENVELOPE NUMBER 1;0,63,53,1;0,-10,10,3;0,
      -10,-10,80;0,-10,-30,30;0,-33,-23,2
110 CLEAR SOUND
120 RESTORE
130 DO
140 READ IF MISSING EXIT DO:G,H
150 LET G=G+24
155 LET H=H/1.3
160 SOUND PITCH G+7,DURATION H,SYNC 1,
      STYLE 16,SOURCE 1,LEFT 0,RIGHT 0
170 SOUND PITCH G,DURATION H,SYNC 1,
      SOURCE 0,ENVELOPE 1,STYLE 80
180 LOOP
190 DATA 37,160,39,13,34,7,39,13,34,7,39,10,37,40
195 DATA 39,13,34,7,39,13,34,7,39,10,37,40,39,40,37,30,
      39,10,42,20,39,10,37,10,34,10
196 DATA 31.8,10,34,40,36.3,10,37,40,39,10,42,10,39,10,
      36.3,10,31.8,200
197 DATA 37,20,97,20,39,20,31.8,20,97,20,34,40
198 DATA 37,20,97,20,39,20,31.8,20,97,20,34,40
199 DATA 37,20,97,20,39,20,31.8,20,97,20,34,40
```

Az említett gyűrűmodulációs gitárhangot úgy érhetjük el ebben a programban, ha a 160. és 170. sorokat a következőkre cseréljük:

```
160 SOUND PITCH G,DURATION H,SYNC 1,
      ENVELOPE 1,STYLE 16
170 SOUND PITCH G+7,DURATION H,SYNC 1,
      SOURCE 2,ENVELOPE 1,STYLE 144
```

Ahogy a példából látható, a gyűrűmodulációs gitárhang-

hoz a 2-es csatornán kell STYLE 144-et megadni (128 a gyűrűmoduláció +16 az alacsony torzítás), ezen a csatornán 7-tel meg kell növelni a PITCH értékét. Míg a felüláteresztő szűrős gitárhanghoz az órajelcsatornán kell 7-tel nagyobb PITCH értéket megadni, melynek hangerejét célszerű 0-ra állítani. Az órajelcsatornán az ENVELOPE-ot nem is kell megadni, hiszen magát a csatornát nem hallhatjuk.

Ha a 0-ás csatornán felüláteresztő szűrőt állítunk be az 1-es csatornával, akkor a 2-es csatorna még szabad, itt beállíthatunk gyűrűmodulációt a 0-ás csatornával. Ennek a gyűrűmodulációnak a szűrt hanggal igen sajátos hangzása lesz!

6. Aluláteresztő szűrő

Az aluláteresztő szűrő azt jelenti, hogy adott hangmagasság felett nem engedi át a hanghullámokat, tehát „alul ereszt át". Ez az effekt kizárólag a zajcsatornán (SOURCE 3) alkalmazható.
Ha a zajcsatornán a STYLE 32 paramétert adjuk meg, akkor a zajcsatorna a 2-es csatorna frekvenciáját fogja használni aluláteresztő szűrőként. Önmagában ez csupán annyit eredményez, mintha egyszerűen a 2-es csatorna frekvenciáját használná a zajcsatorna a STYLE 3 paraméterrel. (Erről az előző, 2019. január-áprilisi számban olvashattunk részletesebben, A zajcsatorna magasságának változtatása egy másik csatorna alapján alcím alatt.) Hiszen a zajcsatorna hangmagassága igen magas, és mindkét paraméter a magasságát csökkenti egy adott hangmagasságra:

```
SOUND SOURCE 3,SYNC 1,STYLE
32    ! A STYLE 3 is ugyanazt eredményezi.
SOUND SOURCE 2,SYNC 1,PITCH 70,LEFT 0,RIGHT 0
```

Az aluláteresztő szűrőnek akkor lesz a fentitől eltérő eredménye (tehát nem csupán egyszerű frekvenciaátadás az órajelcsatornáról), ha módosítjuk a zajcsatorna polinomszámlálójának az értékét. Például a 9 vagy a 7 bites polinomszámlálót használva érezhető lesz a különbség az aluláteresztő szűrő és az egyszerű frekvenciaátadás között:

```
SOUND SOURCE 3,SYNC 1,
STYLE 32+12   ! vagy: STYLE 32+16.
SOUND SOURCE 2,SYNC 1,PITCH 90,LEFT 0,RIGHT 0
```

Még egy négyszögjelcsatornát felhasználhatunk, amely a 2-es csatornától eltérő hangmagasságot ad meg a zajcsatornának, és a zajcsatornán erre a csatornára is hivatkozunk az aluláteresztő szűrő mellett. Így a 32-höz hozzá kell adni 1-et vagy 2-őt, attól függően, hogy a 0-ás vagy az 1-es csatorna frekvenciáját használjuk-e fel:

```
SOUND SOURCE 3,SYNC 2,STYLE 33
SOUND SOURCE 2,SYNC 2,
PITCH 70,LEFT 0,RIGHT 0   ! A szűrő frekvenciája
SOUND SYNC 2,PITCH 60,LEFT 0,RIGHT 0
! A zajcsatorna frekvenciája
```

Természetesen megadhatunk különböző torzításokat is az órajelcsatornákon (akár a szűrő frekvenciájaként használt csatornán, akár a zajcsatorna frekvenciájához használt csatornán), ezek is kihatással vannak a hangzásra.

Hogyan állítsuk elő az Enterprise szürke színét? - vol. 2.

Előző számunkban írtam erről. Megmondom őszintén, kicsit elhamarkodott volt ez a cikk, mert nem gondoltam arra, hogy száradás után a szürke szín kivilágosodik. Igaz, hogy már így is elegáns, de sokkal világosabb szürke mint az Enterprise szürke színe...

Így hát újabb köröket kell ezzel kapcsolatban futni. Utánanéztem, és valószínű, hogy megvan a megfelelő szín (bár nem akarom ezt elkiabálni).

Ez szintén a Mr. Hobby terméke és a German Grey a szín kódja. Bízom benne, hogy ez a szín már stimmelni fog. Az eredményről a következő Enterpress Magazinban beszámolok!

Matusa István

A VL1772-ről újra

- avagy egy rejtélyes hiba nyomában...

Írta: Németh Zoltán (Zozosoft)

Az 1994 márciusi Enterpressben már írtunk a VL1172-es lemezvezérlő IC-ről, egy meglehetősen dühös hangvételű cikket, aminek az alapja a Főszerkesztő úr frissen vásárolt Faragó Gyula féle EXDOS kártyája volt.

Emlékeztetőül: azt a problémát tapasztaltuk ezekkel a VL1772-vel szerelt kártyákkal, hogy se az EXDOS FORMAT parancsával, se EPDOS-szal nem lehetett lemezt formázni. A formázás elindult, de rövid idő után hibával leállt. Meglepő módon a Zozosoft féle FAFO hibátlanul működött ezeken a kártyákon is.

Más hibajelenséget akkor nem vettünk észre, végül tudomásul vettük a dolgot, VL1772-es kártyán FAFO-t kell használni formázáshoz és kész. Biztos nem teljesen kompatíbilis a WD1772-vel...

Kb. 15 évvel később javításra került hozzám Tutus EXDOS kártyája, és amikor már sikerült életre kelteni, azt tapasztaltam, hogy a lemezműveletek igen gyakran Sector not found hibával leállnak, de Retry nyomására folytatódnak. Mivel az EXDOS kártyán, és a hozzá használt buszbővítőn is több hibás IC cserélve lett így azt feltételeztem, hogy valami elektromos sokk érte a kártyát (pl. menet közben kicsúszott a gépből) és ettől meghibásodott a lemezvezérlő IC is. Mivel akkoriban szereztünk be az USA-ból 50 db lemezvezérlő IC-t, nem okozott gondot a csere, a 94-es cikk alapját adó VL1772-es IC a rossz IC-s dobozba került.

2016-ban az EXDOS 1.4 fejlesztésekor eszembe jutott a VL-es történet, és arra gondoltam, hogy meg kéne nézni, hogy mi lehet az a inkompatibilitás, ami miatt az EXDOS FORMAT parancsa nem működik ezekkel az IC-kkel. A meglepetés ekkor jött: hiába teszteltem több VL1772 IC-t is, nem sikerült reprodukálni a problémát! Később az EXDOS 3 fejlesztése kapcsán újra próbálkoztam, de szintén nem sikerült előhozni a problémát. Arra gondoltam, hogy anno a Faragó féle kártyáknak volt valami egyedi hibája, ami időközben ki lett javítva. Pl. totál össze-vissza, a fiók aljáról összeszedett vegyes felvágott 74-es logikai IC-kkel épültek, ezeket az évek folyamán sorra kiforrasztgattam, és 74HCT-kre lett cserélve.
2019 májusában került elő megint a probléma. Pear említette, hogy a WD1772 ára már egekbe szökött (40-50 euró még a kínai oldalakon is!), amire javasoltam, hogy VL1772 még kapható elfogadható áron. Azonban amikor megkapta a megvásárolt IC-ket, kellemetlen helyzet állt elő: az összes IC hibás volt! Egy sima DIR még lefut, de a formázás egy idő után hibával leáll... nagyon ismerős jelenség, egyből előjöttek a 25 éves emlékek!

Újra leteszteltem a nálam lévő jó VL IC-ket, továbbra is semmi hiba... Pear letesztelte más gépeken is a hibásakat, ott sem működtek rendesen... Végül, amikor fotót rakott fel a hibás IC-ről, akkor ugrott be a dolog: ez nagyon ismerős! Előkerestem a rossz IC-s dobozból Tutus régi IC-jét (még szerencse, hogy nem dobtam ki anno!), és hopp 100%-ig egyforma volt a Pear-nél lévőkkel, még a gyártási idő is egyforma volt: 1987 2. hete. Elkezdtem tesztelni, rögtön jelentkezett is a hiba. Ekkor már biztosra vettük, hogy bizonyos gyártási szériák a hibásak. Pear kicseréltette az ebayes eladóval az IC-ket, 1988 13. heti példányokra, amilyenből nálam is volt működő darab. Ezek a kicserélt IC-k már mind hibátlannak bizonyultak. Ezzel a rejtély egy része megoldva, de továbbra is ott a kérdés: mi is ez a titokzatos hiba? Lehet-e tenni ellene valamit szoftveres oldalról, hiszen a FAFO rejtélyes módon működik a hibás IC-kkel is...

Doboz nélküli, szabadon álló meghajtóval tesztelve egyből feltűnt a rendellenes fejmozgás: a fej véletlenszerűen a 0. sávra ugrik. Ez FORMAT esetén végzetes hiba, hiszen elkezdi a korábban felülírt sávokat felülírni, így az ellenőrzésnél egyből kiakad majd hibával. Normál lemezműveleteknél bizonyos szintig észrevétlen marad a probléma, mivel a FISH párszor újrapróbálkozik hiba esetén, így csak akkor jut el a felhasználóhoz a hibajelzés, ha egymás után többször is bekövetkezik a hiba.

Első ötlet az volt, hogy lehet, hogy azért működik a FAFO, mert az visszafele irányban, azaz bentről kifele formáz?

Azaz lehet, hogy a fej befelé léptetése hibás? Készült az EXDOS 3-ból egy módosított változat, ami szintén bentről formáz... ...és továbbra is előjött a hiba!

Közben a fejlesztés alatt álló DRVTEST (EXDOS kártya és meghajtó tesztelő program, részletesebben majd egy külön cikkben írunk róla) is bevetésre került, de érthetetlen módon nem sikerült észrevehető hibát produkálni, sem a fej kézi ide-oda léptetésekor, sem a teljes meghajtó teszt ciklus alatt. Még akkor sem, amikor a Verify opció engedélyezve van, amikor a lemezvezérlő beolvas egy szektorazonosítót a sáv pozicionálás ellenőrzéséhez. Rejtély... Végül az tűnt fel, hogy a teszt ciklus során néha szabálytalan ütemet hallani. A fejmozgásról készült videó felvételt alaposan megnézve sikerült is észrevenni, hogy igen, néha a fej akkor is a 0. sávra ugrik amikor nem kéne!

A DRVTEST-be már az elején bekerült egy olyan ellenőrzés, hogy hibát jelez, ha a fejnek a 0. sávon kéne állni, de nem érkezik 0. sáv jelzés a meghajtó felöl. Most azt kértem Bruce-tól, hogy rakjon bele olyan vizsgálatot is, hogy akkor is legyen hibajelzés, ha a fej a 0. sávra érkezik, miközben nem oda kéne. Innentől kezdve gyönyörű szépen jöttek a hibajelzések, egy teljes teszt ciklusban több száz darab :) Hamarosan egyértelművé vált, hogy a SEEK

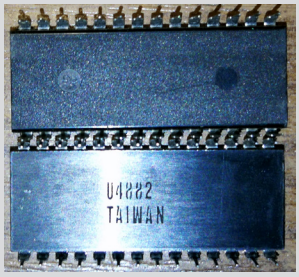
parancs hibás az érintett IC-kben, néha RESTORE (azaz 0. sávra pozicionálás) hajtódik végre helyette. A két parancs kódja között csak egyetlen egy bit a különbség! Valószínűleg valami gyártási hiba történt, ami miatt az adott döntési rész nem működik megbízhatóan. És úgy tűnik a hiba az évek múlásával súlyosbodik, annak idején csak a FORMAT esetén tapasztaltuk a hibát, de kb 15 évvel később már a normál lemezműveleteknél is felbukkant. Még az is lehet, hogy új korukban még hibátlanok voltak, hozzánk 7 éves korukban jutottak el. Lehetséges, hogy valami szennyeződés került a szilícium lapkára, ami szép lassan korrodálja a kérdéses részt.

Fent maradt még a FAFO vs FORMAT kérdés... némi idő után sikerült rájönni a rejtély kulcsára: a FAFO csak a – hibával nem érintett – STEP parancsokat használja, míg az EXDOS általános DISKIO-ja nem törődik azzal, hogy egy vagy több sávot kell pozícionálni, minden esetben a SEEK parancsot. Ezek után Bruce készített egy StepSeek nevű üzemmódot, ami a SEEK parancsot ciklusban kiadott STEP parancsokkal helyettesíti. Ebben az üzemmódban hibátlanul üzemelnek a hibás IC-k is! Tervek szerint ez az üzemmód is bekerül az EXDOS 3-ba, így az már a hibás IC-kkel is tud majd hibátlanul együtt működni. Mivel fogynak az értelmes áron beszerezhető lemezvezérlő IC-k, így nem árt, ha a selejtet is fel tudjuk használni :-D Pear időközben jelezte, hogy a Timex FDD3000 is hibátlanul üzemel ezekkel, ROM programjának vizsgálatakor kiderült, hogy szintén csak STEP parancsokat használ. Még nem próbáltam, de várhatóan a Spectrumos SpeccyDOS illesztő is működik 4s0 ROM használata esetén, ez szintén ciklusba tett STEP parancsokat használ, eredetileg azért, hogy WD1772 esetén is megvalósítsa a WD1770 lassabb fejléptetési tempóját, régi lassú meghajtók számára.

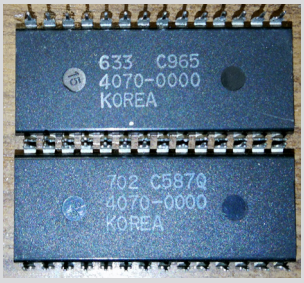
Eddig tesztelt JÓ VL1772-esek gyártási ideje: 1988 13. hét, 1989 21. hét, 1992 30. hét. ROSSZak: 1986 33. hét, 1987 02. hét. Akinek van tapasztalata további jó/rossz VL1772-kel, az írja meg a szerkesztőségnek, vagy a fórumra/facebook csoportba.

VL1772

Jó VL1772



Rossz VL1772



SPECCYALISTA VILÁG
A SPECCYALTA BARÁTI KÖR LAPJA
SINCLAIR.HU

ARCKÉPCSARNOK:

Matthew Smith

Csillag születik

Matthew Smith neve valószínűleg egy Spectrumosnak sem ismeretlen. Ő írta – többek közt – a Manic Minert és a Jet Set Willyt. A Manic Minert, ami az egyik legikonikusabb Spectrumos játék és aminek számos remake-je és nem hivatalos kiegészítője látott napvilágot az elmúlt 30+ évben. Na de kezdjük az elején...

Smith 1966-ban született Londonban. Első számítógépét karácsonyra kapta 1979-ben, ami egy Tandy TRS-80 volt. Erre irt is egy Delta Tower One című Galaxian klónt, amiből Smith szerint kb. 50 példányt adott el a kiadó, ami jónak számított akkor (1980-ban vagyunk!). Matthew csak este tudott dolgozni a gépen, mert a TRS-80 mindig elszállt, ha valaki bekapcsolta a teafőzőt.

Második kiadott játéka a Monster Muncher volt, ami VIC-20-ra készült BASIC-ben.

A Miner Willy franchise

1983-ban a Bug-Byte cég megbízta Matthew-t 3 Spectrumos játék írásával. A cég adott neki egy ZX Spectrumot, amin fejleszthetett. Az első játék a Styx volt, 16K-s Spectrumon futott és 5.95 GBP-be került. A játék célja: eljutni a kijárathoz, miközben szörnyek próbálnak elkapni minket, akiket le lehet lőni. A játék nagyon sikeres volt, Smithnek 30.000 GBP-t hozott, a Bug-Byte pedig nekiadta a Spectrumot.

Ezután szabadságra ment Olaszországba, ahova magával vitt egy piros jegyzetfüzetet. Ebbe a füzetbe rajzolta a Manic Miner pályáit és terveit. Hazatérése után 8 héttel elkészült a játék, aminek grafikajat és zenéjét is ő csinálta. Ez volt az első Spectrumos játék, aminek folyamatos zenéje volt, nem csak effektek. Matthew egyik kedvenc játéka a Donkey Kong volt, amit játéktermekben játszott. Viszonylag drága szórakozás volt, ezért a Manic Minert kicsit a Donkey Kong mintájára csinálta meg, illetve egy 1982-es Atari játék, a Miner 2049'er is inspirálta. A fejlesztés Tandy-n történt, ami szintén Z80 alapú, viszont rendelkezett 5 MB HDD-vel, ami felgyorsította a programozást. Valószínűnek tartom, hogy a gumi billentyűzet hiánya is inkább előny volt. :-) Egy érdekesség: a Manic Minert később Matthew cége, a Software Projects is kiadta. Ezt a Bug-Byte-os szerződés egy apró hibája miatt tehette meg, a játékot 'átvette' a cégtől. Elég sok platformra portolták, többek közt Amigára, Commodore 64-re ill. később megjelent Gameboy Advance-ra is. Egy interjúban a szerző elmondta, hogy a keresett pénzt ész nélkül elszórta. Nem hibáztatom: 17 évesen kevés ember hoz jó pénzügyi döntéseket. Néhány éve azt nyilatkozta, hogy a mai eszével házat venne annyi pénzből.

Míg a Manic Miner fejlesztését nagyon élvezte Matthew, a Jet Set Willy – az ő szavait idézve – egy pokol volt, menedzser lihegett a nyakába és nem volt elég idő a tesztelésre. A bolti verziót nem is lehet végigjátszani több programhiba miatt, ezekre később a Bug-Byte POKE-okat adott ki. A fejlesztési idő kb. háromszorosa volt a Manic Minerének, pedig a. Egy 20 évvel későbbi interjúban Matthew elmondta, hogy még soha nem játszotta végig a Jet Set Willyt. A Manic Minerhez hasonlóan, ebben a játékban is folyamatos zene hallható: a menüben Beethoven Holdfény szonátája, a játékban pedig a Hegedűs a háztetőn darab Ha én gazdag lennek című muzsikája szólt.

A játék 1984-ben jelent meg és a Manic Minernél is nagyobb sikert aratott. Több hónapig uralta a Spectrumos eladásokat.

Matthew eredeti terve az volt, hogy a végén Mariát ágyba kell vinni, aztán Maria megöli Willy-t. A szerző saját bevallása szerint „alkohol és egyéb ártalmas hatóanyagok" befolyása alatt írta a játékot. A fejlesztés befejeztével 2 hónap szabadságra ment, mire visszaért Liverpoolba, már többen a játék folytatásáról beszéltek. Ez 1985-ben el is készült a Software Projects kiadásában – de nem Smith írta meg–, neve Jet Set Willy II: The Final Frontier lett. Ez nem teljesen új játék lett, inkább a Jet Set Willy kiegészítője - ma valószínűleg DLC-nek hívnánk. :-)

Rögös utakon

A Jet Set Willy után egy The Mega Tree nevű játékon kezdett el dolgozni, amit Commodore 64-re írt, ekkor már saját cégénél. Ez nem vonzott elegendő tömeget és Matthew üzleti partnerei türelmetlenek voltak, ezért 3 hónappal a fejlesztés kezdete után leállt a projekt.

1987-ben több játékmagazinban is hirdetések jelentek meg egy új játékról, az
Attack of the Mutant Zombie Flesh Eating Chickens From Mars-ról, amit szintén Matthew írt, de soha nem adta ki. Állítólag azért, mert nem volt elégedett a programmal, miután elkészült.

1988-ban bezárta Software Projects cégét, aztán eltűnt a színről, részben anyagi okok miatt. 1995-ig Wallasey-ben élt, ahol felnőtt, egy ideig kórházban is volt, pszichiátrián kezelték. Ennek több oka is volt, saját bevallása szerint kiégett. Nem volt nagy titok az sem, hogy nem vetette meg a drogokat. A Manic Miner fejlesztése alatt is használt kábítószereket, némelyik szörny ennek az eredménye. :-)

Az Internet terjedésével páran feltették a kérdést, hogy mi történt Matthew Smith-szel. Sok pletykát lehetett olvasni, hogy éppen hol van és mit csinál. Páran biztosan emlékeznek a Where is Matthew Smith oldalra.

1995-ben Hollandiába utazott, hogy helyretegye az életét. A kompig stoppal jutott el. Egy kommunában lakott és többféle munkát is elvállalt, de egyik sem volt játékfejlesztés. Dolgozott élelmiszergyárban ill. supermarketekben virágokat is pakolt.

1997-ben deportáltak az országból, mert nem újította meg a tartozkodási engedélyét. Miután visszatért Angliába és felfedezte az Internetet, meglepődve tapasztalta, hogy sokakat érdekel, hogy mit lett vele.
Matthew Smith újratöltve

1999-ben egy Runecraft nevű cégnél kapott állást. A cég egyetlen játéka a Scrabble volt Game Boy Color-ra mielőtt csődbe mentek.

2010-ben az Elite Systems újra kiadta – több Spectrumos játékkal együtt – a Manic Minert és a Jet Set Willyt iOS-re, Androidra és a Kindle tabletekre is. Később megjelent Windows Phonera is.

2013-ban az Elite Systems bejelentette, hogy Matthew Smith-szel együtt egy új játékon dolgoznak. Sajnos ennél többet nem árultak el és azóta sincs több info.

A mostánaban készült interjúkból sajnos latszik, hogy az élet megviselte Matthew-t. Nem lett happy end a korai sikerekből. Reméljük még hallunk felőle és visszatér egy felejthetetlen alkotással.

"Five years ago I was a washout. Ten years ago, I was history... Now I'm a legend." -- Matthew Smith

"Öt éve egy katasztrófa voltam. Tíz éve történelem voltam... Most viszont legenda vagyok." -- Matthew Smith

Barabás Péter (z0d)

Megjelent a Speccyalista Világ 2016/2. számában

NEXTBUILD - Integrált fejlesztőrendszer az

Helyzetjelentés

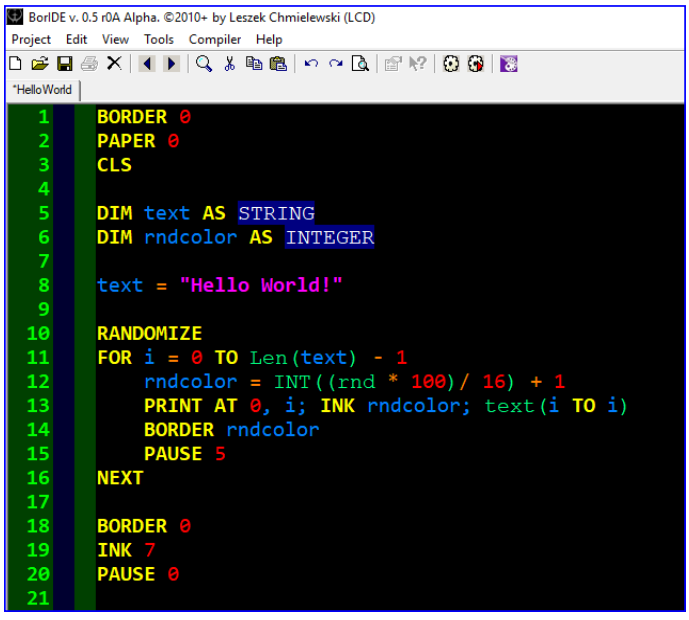
Az előző cikkemben úgy zártam a soraimat, hogy az új Spectrum személyi számítógép, a ZX Spectrum NEXT már szinte itt van a sarkon. Bő félév múlva azt kell belátnom, hogy nagyot tévedtem. A NEXT project, bár hatalmas lépéseket tett meg a végkifejlet felé, a mai napig sem fejeződött be.
Érdekes módon a problémát nem maga a hardware, hanem a gép „kasztnija" okozza (amibe beleérthetjük magát a dobozt, a billentyűzetmembránt és a billentyűzet gombjait is). Ráadásul megint eltűnt jelzés nélkül egy beszállító cég, ami ezúttal is nagyot tudott lassítani a folyamatokon. A legfontosabb cél most a teljes, dobozos NEXT prototípusának legyártása és tesztelése. Ha ez megvan és sikeres, csak akkor indulhat a sorozatgyártás és az elkészült termék kiküldése a Kickstarter tagoknak. Ennek az időpontját találgatni teljesen értelmetlen, csak remélni lehet, hogy legalább karácsonyra minden lezárul. Ami biztos, hogy az alkotók nagyon keményen dolgoznak, és szívüket-lelküket beleteszik a projektbe. Aggódni ezért szerintem felesleges, a gép meglesz, csak ki kell várni valahogy.
A hardware mellett a gép firmware-je, illetve operációs rendszere és felhasználói kézikönyve intenzív fejlesztés alatt van. Több kisebb-nagyobb játék és egyéb szoftver is be lett jelentve, sőt néhány már el is készült és megvásárolható az interneten keresztül letöltés formájában vagy korlátozott példányszámban akár dobos termékként is.

NEXTBUILD, egy eszköz mind felett?

A számos, fejlesztés alatt álló projekt közül szeretném kiemelni a NEXTBUILD integrált fejlesztőrendszert, ami nem más, mint több fejlesztőeszköz, két emulátor és egy nagyszerű IDE (integrált fejlesztői környezet) együttese.
Miért izgalmas ez? Egy számítógépes platform szíve-lelke a hatékony és lehetőség szerint kényelmes fejlesztői eszközök megléte. A fejlesztők nem tudnak új programokat készíteni, ha ezek nem állnak rendelkezésre és programok nélkül a hardware maga halálra van ítélve. Minél jobbak ezek az eszközök, annál többen mernek belevágni új projektekbe (akár hobbistaként vagy amatőrként is), annál jobban pezseg a platform körül az élet. Az eszközök mellett persze fontos a jól dokumentált és elérhető fejlesztői könyvtárak/API-k megléte is (amin keresztül a programozó el tudja érni és vezérelni tudja a gép egyes részeit: a grafikát, hangot, háttértárat, stb.) Ezek még azért erősen képlékeny állapotban vannak jelenleg, hiszen maga a gép is fejlesztés alatt áll, de egyre több információ áll rendelkezésre a NEXT weboldalán és a különböző fórumokon, Illetve szerencsére maga a NEXT közösség is nagyon segítőkész.
A NEXTBUILD lényegében egy keretrendszer, ami lehetővé teszi azt, hogy magát a programozást, a kód lefordítását, tesztelését illetve kipróbálását egy programon belül el lehessen végezni. Integrált eszköz nélkül ezekre a fejlesztés során külön fordítót, szövegszerkesztőt és emulátort kell használni, de ezeknek a megtanulása időigényes lehet és lassíthatja is a fejlesztést. A rendszert kezdő és profi fejlesztőknek egyaránt ajánlják, és akár normál ZX Spectrum programokat is lehet majd benne fejleszteni, bár az ajánlott célplatform a NEXT.
Lássuk akkor, milyen komponensekből is áll az új rendszer:
BorIDE: Integrált fejlesztői környezet. A programkód kényelmes szerkesztését teszi lehetővé, „Syntax highlighting" támogatással.
ZXB: Basic fordító (a rendszer „lelke") amivel assembly sorokat is tartalmazó programokat is fordíthatunk. A fordító a felhasználó által definiált könyvtárak használatát is megengedi.
CSpect és FUSE: két nagyszerű emulátor. A CSpect ZX Spectrum Next-et, a FUSE normál Spectrumokat képes emulálni.
NextLibs – ZX Spectum NEXT könyvtárak, amelyek segítségével a ZXB fordító el tudja érni a Next hardware-t és így látványos (a rendszer képességeit teljesen kihasználó) programokat tudunk készíteni. A könyvtár persze még szintén fejlesztés alatt áll, eddig az SD kártya filekezelő és a NEXT továbbfejlesztett grafikai képességeit kihasználó rutinokból készült el néhány (Layer2, sprite-ok).
Maga a ZXB fordító is tartalmaz saját könyvtárakat (matematikai, grafikus, szövegkezelő rutinokat.)

Hello World!

Hogy lássuk, milyen egyszerű folyamat a programkészítés a NEXTBUILD rendszerben, készítsük el a már-már kötelező „Hello World" programot egy kicsit felturbósítva:
A Nextbuild rendszert elindítva nyomjuk meg a „New" gombot vagy válasszuk a Project->New menüpontot.
A megjelenő ablakon gépeljük be a következő rövid programot:



új ZX Spectrum NEXT személyi számítógéphez

Az F11 gomb megnyomásával a Nextbuild lefordítja programunkat, elindítja a beállított emulátort (az alapbeállítás szerint a CSpect), majd betölti az emulátorba a programunkat, ami el is indul automatikusan.

A sima Spectrum programok esetében - amik nem használják ki a Next különleges képességeit - beállíthatjuk, hogy a CSPect helyett a FUSE emulátor fusson. Ehhez csak a „Launch Fuse" checkboxot kell kiválasztani a jobb oldalon megjelenő opciók közül.
Fontos megemlíteni, hogy jelenleg a CSpect emulátor csak .sna image fájlokat képes fogadni, tehát ilyen esetben a fordító mindig sna image-t fog készíteni. A többi választható output-típus (pl. tap, tzx, asm, vagy bin) beállítása ilyenkor nincs hatással a működésre.

A program futtatásának eredményeként, a „Hello World" feliratnak karakterenként és minden karakter esetében különböző színnel kell megjelennie, valahogy így:

Rövid magyarázat a programhoz:

Az első három soron beállítjuk a képernyő és a keret színét feketére, majd töröljük a képernyőt.
Az 5-6. soron két szám típusú változót deklarálunk. Az első magát a szöveget, a másik az aktuális betűszín számkódját tartalmazza majd a futás során.
A 8. soron beállítjuk a megjelenő szöveget a deklarált, „text" nevű és string típusú változónkba.
Utána a RANDOMIZE utasítással a Spectrum Basic véletlenszám-generátorát inicializáljuk, hogy minden egyes futtatásnál más és más színben jelenjenek meg a szöveg egyes betűi.
A 11. és 16. soron egy ciklust definiálunk. A ciklus magját (12-15. sorok) annyiszor fogja a program végrehajtani, ahány betű van a szövegünkben.
A ciklusmagon belül a véletlenszám-generátor segítségével meghatározzuk az aktuális betűszínt, majd a 11. soron ki is íratjuk azt. Az AT és az INK parancs segít, hogy az „i" ciklusváltozó által meghatározott sorba és oszlopba kerüljön a betű a megfelelő színnel.
A 14. soron beállítjuk a BORDER (keret) színét az aktuális színre (hogy legyen egy kis „effekt" a program futása során), a betű kiírtatása után várunk egy picit, majd jöhet a következő betű.
Ha teljes szöveg megvan, akkor a keretet feketére, a betű színét fehérre állítjuk, majd a PAUSE 0 paranccsal megállítjuk a programot egy gombnyomásra várva. Erre a CSpect emulátor miatt van szükség, mivel a PAUSE parancs nélkül a program futása után az emulátor azonnal reseteli a gépet.
A programablak alatt egy üzenetablak kapott helyett, amiben a fordító üzenetei (például az esetleges fordítási hibák szövege) jelennek meg. A hibák szövegét a program egy jegyzettömb ablakban is megmutatja nekünk. Ezt az ablakot zárjuk le még a következő indítás előtt, különben a fordítás hibára fog futni!
Amint látható, a programozás és a tesztelés nagyon egyszerű, a program gombnyomással indítható és az eredmény azonnal látható.
Fontos megjegyezni, hogy a NextBuild program még nem készült el teljesen, pár funkció még nincs elkészítve vagy éppen befejezve, és hibás működésre is számítanunk kell a használat során. Mindezek ellenére a Nextbuild már most is egy nagyon kényelmes és hasznos fejlesztőeszköz, amit nyugodt szívvel lehet ajánlani mindenkinek, aki a Spectrum Next programozására kedvet érez.
Több igen fejlett példaprogram is található a csomagban, érdemes közülük szemezgetni. Ezekből kettőt ajánlanék: az egyik a BigScroller, ami a demó programokban néha látható hatalmas szöveget „tekeri végig" a képernyőnkön nagyon simán és nagy sebességgel. A másik a Chicken, ami akár egy rendes mászkálós/platformer játék egyik helyszíne is lehetne. Magához a ZXB fordítóhoz is sok példa van mellékelve, ezek a ZXBC\examples könyvtárban találhatóak.
Remélem sikerült kedvet csinálnom a NEXTBuild-hez. Én magam nagy izgalommal várom az újabb verziók megjelenését. Az alap, amit az alkotók leraktak, nagyon erős, de szerintem még hosszú út van hátra addig, amíg a NEXTBuild-ből igazi, kiforrott fejlesztőrendszer lesz. Sok múlik azon, hogy a NEXT-es közösség hogy fogadja a NEXTBUILD-et: lesznek-e elegendően, akik használják, illetve a fejlesztők kitartása is töretlen marad-e. Véleményem szerint erre minden esély megvan.

Befejezésül szeretném a teljesség igénye nélkül felsorolni, hogy kik vesznek részt a NEXTBUILD, illetve az egyes komponensek fejlesztésében:
BorIDE integrált fejlesztői környezet: Leszek Chmielewski (LCD)
XZB Spectrum Basic fordító: Boriel (http://www.boriel.com/wiki/en/index.php/ZXBasic)
CSpect emulátor: Mike Dailly (https://dailly.blogspot.com)
Fuse emulátor: Philip Kendal és még sokan mások (http://fuse-emulator.sourceforge.net)
NextLibs: David Saphier

Vándorffy Tamás (Vándor)

Megjelent a Speccyalista Világ 2018/3. számában

sinclair
ZX Spectrum Next

Kérem a következőt!

avagy NextBASIC újdonságok

A sztárfellépő késik, de azért belesünk a függöny mögé és felvázoljuk, hogy mi újat hoz majd a Next BASIC.

Számos meghatározó hardveres újdonsággal áll elő a ZX Spectrum Next, és a legtöbbhöz szerencsére BASIC nyelvi támogatás is társul majd, hogy ne csak azok aknázhassák ki a vas lehetőségeit, akik a kontinentális talapzat mélységéig ismerik a gép felépítését és a gépi kódú programozást. Nagy a csábítás, hogy a grafikus képességek vagy a hang irányából kezdjük a lehetőségek tárgyalását, de indításként maradjunk inkább a legalapvetőbb erőforrás, a memória kihasználásánál.

A ZX Spectrum Next 1 vagy 2 MB-os RAM kapacitása a 80-as években bármilyen háttértárnak is becsületére vált volna, memóriaként pedig kifejezetten elképzelhetetlen volt, legalábbis az otthoni számítástechnikában. A Z80 azóta is csak 16-bites címbusszal dolgozik, ami a tudomány mai állása szerint is csak 65536 memóriarekeszt enged közvetlenül címezni, így ha el szeretnénk érni az extra memóriát, ahhoz trükköznünk kell. Szerencsére ezt elintézi helyettünk a NextZXOS és a Next BASIC. A BANK utasítás segítségével a tár bármelyik 16K-s blokkját (bankját) használhatjuk memória- és fájlműveletekhez.
Persze ez sem úgy van, ahogy mi mórickásan elképzelnénk, nem cím szerint szekvenciálisan osztották ki a bankok sorrendjét, hanem a korábbi 128-as Spectrum modellekkel való kompatibilitás érdekében ragaszkodtak a hagyományokhoz, így a NextZXOS használja az első 9 bankot:

Sorszám	Foglalás
0	Standard ZX Spectrum memória (címtartomány: 49152-65535)
1	RAMdisk
2	Standard ZX Spectrum memória (címtartomány: 32768-49151)
3	RAMdisk
4	RAMdisk
5	Standard ZX Spectrum memória (címtartomány: 16384-32767)
6	RAMdisk
7	A NextZXOS számára fenntartott munkaterület
8	A NextZXOS számára extra képernyőmódokhoz (lo-res, Timex hi-res és Timex hi-colour módok) és egyéb adatokhoz fenntartott munkaterület

A bankok indexelése 0-tól indul, bank 9-től kezdve a memória a programozó rendelkezésére áll, 1 MB-os RAM méret esetén 47, 2 MB-os RAM esetén 111 a legnagyobb használható bank index. (Ezt adja vissza a MAXBNK rendszerváltozó értéke is.) Lehet garázdálkodni a 48K-s Spectrum standard területét képező 5, 2 és 0 sorszámú blokkokban is, de óvatosabban: csak az 5-ös bank 0-6911-ig terjedő képernyőtára biztonságos és a RAMTOP rendszerváltozó feletti részek, egyébiránt könnyen belerondíthatunk a BASIC munkaterületbe és a standard rendszerváltozókba, melyek a képernyőtár felett sorakoznak.

Ahogy a táblázatban látható, az 1, 3, 4 és 6-os számú szeleteket a jó öreg RAMdisk használja, a 128-as Spectrum tulajdonosok BASIC-ből leginkább ennek használatával, azaz fájlműveletekkel tudták kihasználni az extra memóriát. Ha valakinek nem lenne elég az, ami a 9-es banktól felfelé rendelkezésére áll, akkor a RAMdisk területeket felszabadíthatja a BANK 1346 USR paranccsal. (Ekkor persze RAMdisk balra el.) A7-8-as bankokat a BASIC BANK parancsok segítségével nem lehet elérni.

A következő parancsok használhatók a BANK-kal kombinálva:

POKE, PEEK, USR, COPY TO, ERASE, FORMAT, CLEAR, LAYER.

Az alábbi példák segíthetnek megérteni a BASIC adta memóriakezelési startégiáját:

BANK 9 USR 49152
Elindítja a bank 9-ben elhelyezett gépi kódú programot az adott címtől. Előtte belapozza a megadott indexű bankot a 48K-s memória legfelső 16K-s szegmensébe, így a

címnek mindig 49152 és 65535 között kell lennie. Ügyelj a RAMTOP-ra és az UDG-re!

BANK 10 POKE 0,255
Elhelyezi a 255-ös decimális értéket a 10-es bank legelső (nulladik) byte-jában. Egy-egy érték beszúrására célszerű utasítás, de aki egész blokkokat másolna FOR ciklusba ágyazott BANK ... POKE utasítással, az legyen szíves ne tegye, mert alább jön még ugyanerre a célra sokkal alkalmasabb eszköz.

BANK 9 PEEK 15000 TO numvar
Elhelyezi a 9-es bank 15001-edik memóriarekeszének tartalmát a numvar nevű numerikus változóban. Mivel a bank 16K hosszú, ezért a PEEK paramétereként legfeljebb 16383 adható meg. Hát igen, nem szép megoldás a 21. században, hogy nem egy függvény visszatérési értékével adják át a művelet eredményét, de egyrészt nyelvtől függ, hogy mikor történik valóban érték vagy cím szerinti átadás, másrészt 8 biten azzal gazdálkodunk, amit találunk.

BANK 11 COPY TO 9
Az egész 16K területet átmásolja a BANK 11-ből a 9-be. Gyorsabb, mint FOR és BANK ... POKE utasításokkal, hidd el.

BANK 11 COPY 4096,256 TO 9,8192
Átmásol 256 bájtot a 11-es bank 4097-edik (nullától számozunk, ugye) memóriarekeszétől a 9-es bankba, annak 8193. (azaz 8192-es számú) rekeszétől kezdődően.

BANK 12 ERASE 255
Teletölti a 12-es sorszámú (nem számít, de sorrendben a 13-ik) bankot 255-ös decimális értékekkel. Ha az értéket nem adod meg, akkor 0-t használ.

BANK 12 ERASE 9999,1024,255
Opcionálisan két paraméterrel bővíthető a BANK ... ERASE (eggyel nem!). Példánk feltölt a 12-es sorszámú bank 10000. byte-jától kezdve 1024 memóriarekeszt a 255-ös értékkel.

BANK 47 CLEAR
Eddig nem volt róla szó, de ha egy bank tartalmát bármilyen utasítással (BANK PEEK/POKE/COPY/ERASE/LOAD) megváltoztatod, akkor a rendszer foglaltként jelöli azt. Ha nincs rá szükséged, akkor érdemes felszabadítani, hogy a rendszer saját céljaira használhassa.

BANK NEW newbankvar
Lefoglalja a következő szabad bankot és sorszámát eltárolja a paraméterként megadott newbankvar változóban. Ugyan a fentebb említett parancsok lefoglalják az érintett bankokat, de olykor a BANK NEW jól jöhet, ha fut egy rezidens program, melynek memóriaigényeit nem ismerjük.

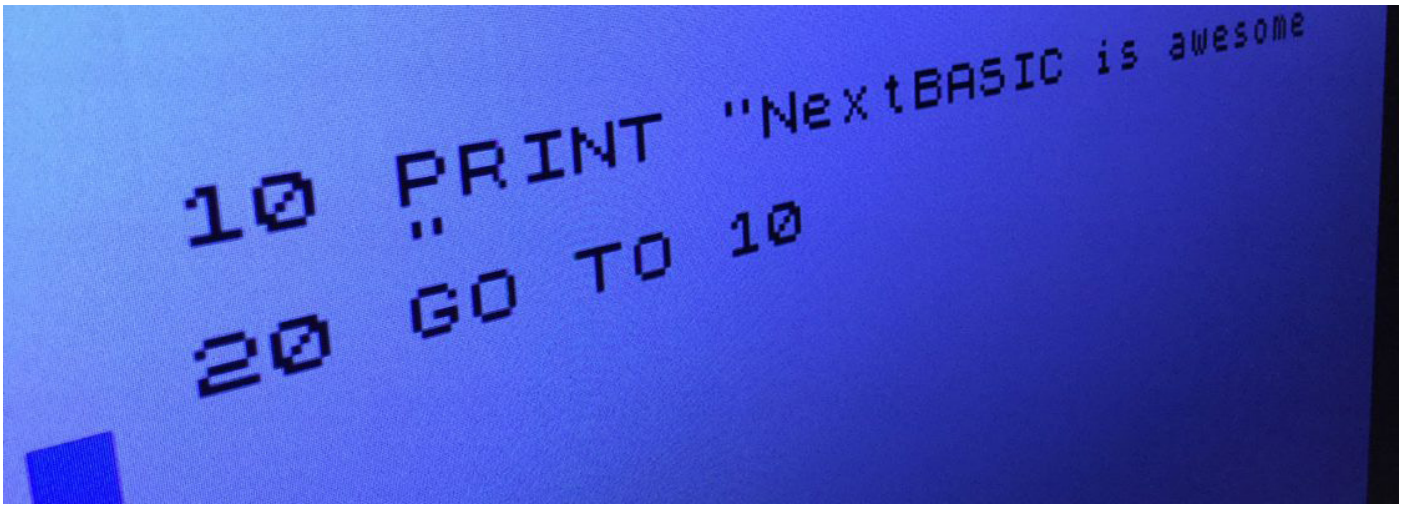
Összegzésként a következőket állapíthattam meg:

- A NextBASIC lehetővé teszi, hogy akár byte-onként, de legfeljebb 16K-s blokkonként elérd a memória egészét, eltekintve a rendszer által fixáltan használt részektől.
- A bankok sorrendjének semmi köze a 48K-s Spectrum címkiosztásához, attól független.
- A NextZXOS nem teszi lehetővé, hogy összevissza kioszd a bankokat az általad jónak tartott címtartományokba, mert ezt a rezidens programok és az OS nem tudnák hatékonyan követni, valamint a kompatibilitásnak is annyi lenne. A kiosztás fix és a legtöbb hozzáférés blokkmásolásból áll, de ne aggódj, nem lesz ez olyan lassú.
- Amit BASIC-ben lefoglalsz, arról a NextZXOS tudni fog, nem rondít bele és a rezidens programok is tiszteletben tudják tartani (ha megfelelően írják meg azokat)
- A képernyőtartalom és a sprite-ok minden módban egyszerűen menthetőek (részletesen lásd majd a cikk folytatásában)
- Az ESX DOS fájl műveletek a megadott terület tartalmát (pl. screen vagy tömb) is tiszteletben tartva képesek dolgozni, igazi 8-bites Kánaánt nyitva a 48K-ból kiszabadult programozók számára.

A következő részekben még tovább vesézgetjük a NEXT BASIC újdonságait, a fájlkezelő parancsokat és egyéb grafikai lehetőségeket.

Egri Imre (Zimi)
Folytatjuk!

Megjelent a Speccyalista Világ 2018/3. számában



Entersnake

Írta: Bodnár Tamás
(Szipucsu)

A játék a Hetedhéten túl könyvben bemutatott Hamika nevű basic játékprogram alaposan továbbfejlesztett és Zzzippel lefordított változata. A Zzzip bevetése következtében fergeteges sebességgel képes kukacunk száguldozni, de csak akkor, ha a játéktéren felbukkanó varázsszerekből felvesszük, amelyik erre képessé tesz.
A Hamikához képest nem csak változatos hangeffektekkel és zenékkel bővült ki a játék, hanem a képernyő is sokkal színesebb lett. Egyes pályákon kimehetünk a képernyő oldalán és a másik oldalán megjelenünk. Játék közben többféle tárgy bukkan fel a képernyőn, melyeket felvéve sebességünket gyorsíthatjuk vagy lassíthatjuk, a hátralevő időnket megduplázhatjuk vagy rosszabb esetben megfelezhetjük, különleges képességekre tehetünk szert vagy éppen rossz képességeket szerezhetünk, plusz életet kapunk vagy éppen elvesztünk egy életet, és egyéb furcsaságok történhetnek.
A játék célja a megadott mennyiségű villogó gombócot összegyűjteni, majd a bal oldalt lent nyíló kapun át elhagyni a játékteret. A pályák indulásakor pont annyi gombócot kapunk a képernyőre, mint amennyit össze kell szednünk, azonban ha kifutunk az időből, vagy ha bizonyos tárgyakat veszünk fel, az összeszedendő gombócok száma megnövekszik, vagy éppen felére csökken. A játék tehát időre megy, viszont nincs életvesztés, ha letelik az idő, csak újabb gombócokat kapunk a képernyőre.

A játék húsz pályát tartalmaz, melyek fokozatosan nehezednek. Életet akkor vesztünk, ha falnak vagy akadálynak ütközik a kukacunk. Ilyenkor a pályát teljesen elölről kell kezdenünk. Pontszámot csak a megevett gombócok után kapunk (a felvett tárgyak után nem), illetve a pályák végén a kukac elért hosszát és a hátralevő időt is hozzászámolja a gép a pontszámunkhoz. A felvehető tárgyak segítségével elérhetjük, hogy több gombóc legyen a képernyőn, mint amennyit a pálya teljesítéséhez össze kell szednünk, így a kukacot rendkívül nagyra is meghízlalhatjuk, mielőtt elhagyjuk a játékteret, így nagyobb pontszámra tehetünk szert. Tehát nem akkor érjük el a legnagyobb pontszámot, ha mindegyik pályát a lehető leggyorsabban teljesítjük, hanem akkor, ha az egyes pályákon minél többet bolyongunk, minél több gombócot felszedve.
A menüben három nehézségi fokozat közül választhatunk az 1, 2, 3 gombok segítségével. Az alapbeállítás a medium, azaz közepes fokozat. Ebben a fokozatban az idő múlásával párhuzamosan a sebesség is egyre növekszik, míg az idő el nem éri a nullát, ekkor visszaáll a lassabb sebesség. Könnyű (easy) fokozatban a sebesség nem gyorsul az idő múlásával párhuzamosan. Nehéz (hard) fokozatban pedig még gyorsabban gyorsulunk, ahogy telik az idő. Könnyű fokozatban 2, közepes fokozatban 5, nehéz fokozatban 6 pontot kapunk egy gombóc elfogyasztásáért.

Játék közben a következő kijelzőket láthatjuk:

Fent:
LIVES - életek száma. Kezdetben 3, maximum 8 lehet.
EASY / MEDIUM / HARD - a nehézségi fokozat.
LEVEL - aktuális szint

Lent:
SCORE - pontszám
TIME - hátralevő időnk
SPEED - sebesség: 1 a leggyorsabb, 9 a leglassabb
LENGTH - a kígyó hossza gerezdekben (karakterekben) mérve. Jelentősége azért van, mert minimum 16 hossz esetén lesz két tárgy felvételének pozitív hatása. Valamint, a pálya teljesítésekor az elért hosszt hozzászámolja a pontszámhoz. A kígyó hossza minden gombóc megevése után nő. Ha 8 életünk van és még egy életadó tárgyat fel akarunk venni, akkor is növekszik a kígyó hossza. Kellőképpen elszántak, akár 200-ra is növelhetik a kígyó hosszát, de már a 120 körüli érték is komoly problémákat okozhat még az egyszerűbb pályákon is.
LEFT - Az összeszedendő gombócok számát jelzi, ami a kapu kinyitásához szükséges. Ha eléri a nullát, és még veszünk fel gombócokat, azokat „lenyeli", nem kerülnek beszámításra, és az idő leteltével mindig pluszban újabb gombócokat kell összeszednünk.

A játékképernyőn a következő tárgyakkal találkozhatunk:

kígyógomba - plusz élet. Kezdetben 3 életünk van. Összesen 8 életet gyűjthetünk össze. Ha telhetetlenek vagyunk és még egy életet felveszünk, az életek száma eggyel csökken, a kígyó hossza viszont négy gerezddel nő!
számok 1-9-ig - sebességünket változtatják meg. Minél kisebb a szám, annál gyorsabb lesz a kígyó.
megrepedt személyi igazolvány - véletlenszerű értékre változtatja a sebességet. Ha még 16 gerezdnél rövidebb a kukac, akkor az a kellemetlen hatása is van ennek a tárgynak, hogy az egész képernyőt 180-kal megfordítja, ilyenkor a fel és le irányok megcserélődnek.
A sebesség egészen addig nem változik meg, amíg újabb sebességváltoztató szert fel nem veszünk! Még elhalálozás vagy a pálya teljesítése esetén sem áll vissza az eredeti (7es) sebesség!
mutáns növény - Ha 16 karakternél rövidebb a kukac, akkor további gombócokat szór el a képernyőn, megnövelve az összeszedendő gombócok számát is, viszont az időt maximumra állítja vissza. Azonban, ha már legalább 16 a hosszunk (ezt a keret színének a megváltozása is jelzi), akkor a tárgy felvétele megfelezi az összeszedendő gombócok számát, viszont az időt harmadára csökkenti! Ez főleg akkor jó, ha elég nagy mennyiségű gombócot (például akár 30-40-et) kellene összeszednünk. Ha már összeszedtünk minden gombócot, nyitva a kapu, és akkor vesszük fel ezt a tárgyat, akkor a kaput bezárja, és újabb gombócokat kell gyűjtenünk.
villanyoszlop - T alakja a Time szóra utal. A hátralevő időnket véletlenszerűen megduplázza vagy megfelezi. Ha az idő nagyobb, mint 100, akkor csak megfelezheti, hiszen túl nagy kiváltság lenne olyan sok idővel szaladgálni (araszolni) a képernyőn.
narancs - Felvételekor mindig újabb gombócokat szór el a képernyőn, megnövelve az összeszedendő gombócok számát is. Ha nyitva a kapu és akkor vesszük fel, az a legrosszabb, mert bezárja a kaput, újabb gombócokat szór el a játéktéren, és még a képernyőt is elfordítja 180 fokkal.
menet nélküli kiscsavar - Nagyon sokat segíthet, de nagyon sokat árthat is a felvétele. Ha már minimum 16 egységnyi hosszú a kígyó, akkor nagyon jó a hatása, mert a kígyó gombócokat fog maga után húzni, de ezzel párhuzamosan nem növekszik az összeszedendő gombócok száma, tehát a pálya teljesítését nagyon leegyszerűsíthetjük vele. Ha még nincs meg a 16 egységnyi hossz, akkor viszont felvételével azt érjük el, hogy a kígyó falat fog húzni maga után, mellyel a találkozás halálos! Hatása akkor múlik el, ha letelik az idő, vagy ha sebességmódosító tárgyat veszünk fel, illetve ha elvesztjük egy életünket vagy teljesítjük a pályát.

Ha a játéktér a feje tetejére állt, az úgy is marad, amíg fel nem veszünk egy sebességmódosító pirulát (valamilyen számot). Elhalálozással vagy a pálya teljesítésével is visszaáll az eredeti rend.

A játék célja tehát nem feltétlen az, hogy az összes, képernyőn lévő gombócot összeszedjük. Hiszen ha a mutáns növénnyel megfeleződik az összeszedendő gombócok száma, vagy ha a menet nélküli csavar felvétele után gombócokat húzunk magunk után, jóval több gombóc is lehet a képernyőn, mint ami a pálya teljesítéséhez szükséges. Általában az a legcélszerűbb, amikor még 16 alatt van a kígyó hossza, hogy különböző varázsszerekkel további gombócokat szóratunk a képernyőre, melyeket összeszedve könnyel elérhetjük a 16 egységnyi hosszt. Ezután már a menet nélküli csavarral és a mutáns növénnyel trükközve könnyen kinyithatjuk a kaput. Közben persze nem baj, ha nem megyünk a falnak és nem is állítjuk feje tetejére a képernyőt.
Kreatív és merész játékosok az utolsó pályát egyszerűbben is teljesíthetik.
Játék végén beírhatjuk a nevünket. A pontszámtáblát a program el is menti.

A grafikus-karakteres (gra-cha) üzemmódokról, 1. rész

Írta: Bodnár Tamás (Szipucsu)

Talán minden Enterprise tulajdonos kísérletezgetett a 80-as, 90-es években a SET VIDEO MODE és a SET VIDEO COLOR utasításokkal. (A brit angolban colour, az amerikai angolban color ez a szó. Bruce Tannerék, a basic készítői mindkét nyelvváltozat beszélőire gondoltak, így gépünk elfogadja a SET VIDEO COLOR és a SET VIDEO COLOUR utasítást is. Hasonlóan különbözik a brit és amerikai angolban pl. az odour – odor (illat), honour – honor (becsület), parlour – parlor (szalon), humour - humor szó is). Bizonyára mindenki botlott olyan esetbe, amikor olyan videólap nyílt meg, amelyekre ha karaktereket írt, a karakterek olvashatatlanok voltak. Valaki talán azt gondolta, valamit itt elszúrtak a gép készítői. Pedig itt rendkívüli lehetőségek nyílnak meg előttünk, amit a Felhasználói kézikönyv még csak érintőlegesen sem említ!
Nézzük csak! A SET VIDEO MODE 0 a jól ismert 40 karakteres videólapot jelenti, amely a gép indulásakor is farkasszemet néz velünk. A SET VIDEO MODE 2 a TEXT 80-nal elérhető, 80 karakteres videólapot jelenti. A SET VIDEO MODE 1 a nagyfelbontású grafikus lap, amilyet az F6 funkcióbillentyűvel is megnyithatunk. A SET VIDEO MODE 5 az általában nem annyira hasznos, kisfelbontású grafikus lap, míg a SET VIDEO MODE 15 a kis különc, egyedi attribútum mód. Összefoglalva:

Szöveges videomódok:
SET VIDEO MODE 0 – 40 karakteres szöveges lap
SET VIDEO MODE 2 – 80 karakteres szöveges lap

Grafikus videomódok:
SET VIDEO MODE 1 – nagyfelbontású (hires) grafikus lap
SET VIDEO MODE 5 – kisfelbontású (lores) grafikus lap

Attribútum videomód:
SET VIDEO MODE 15

Egy valótlan állítás következik: A SET VIDEO COLOR csak a grafikus módokhoz állítja be a színek számát – ezt sugallja a Felhasználói kézikönyv, legalábbis azok számára, akik azt gondolják, színek a grafikához járnak, a szöveghez nem. Így megadhatjuk például a SET VIDEO MODE 1 (nagyfelbontású grafikus lap) és 5 (kisfelbontású grafikus lap) mellé a SET VIDEO COLOR 0-3 akármelyikét, így 2, 4, 16 és 256 színnel telepingálható grafikus lapokat kapunk:

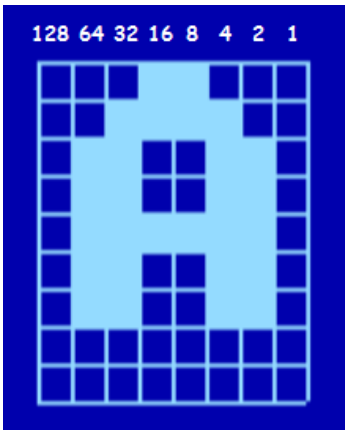
SET VIDEO COLOR 0 – 2 szín
SET VIDEO COLOR 1 – 4 szín
SET VIDEO COLOR 2 – 16 szín
SET VIDEO COLOR 3 – 256 szín

És itt jön valami érdekes! A SET VIDEO MODE 0 (40 karakteres szöveges lap) és 2 (80 karakteres szöveges lap) mellé, tehát a szöveges videolapok mellé is megadhatunk SET VIDEO MODE-nak 0-tól eltérő értéket is! Mi történik ilyenkor? Olyan szöveges lapokat nyithatunk így meg, melyeken egy karakteren belül 4, 16 ill. 256 színt is használhatunk (a háttér színét is beleértve). Ilyenkor alaphelyzetben nem lesz olvasható ezeken a videolapokon a szöveg, mert a karakterek helyett azokra csak nyomokban emlékeztető krikszkrakszok fognak megjelenni. Tehát akár áttehetjük ilyen módba a Hamikát, a Baltazár az űrszemetest vagy bármelyik, karakteres képernyőn futó játékot, és a karakterek helyén krikszkrakszok jelennek meg, viszont azok legalább szép színesek lesznek. Ilyen videolapot nyit meg a következő program:

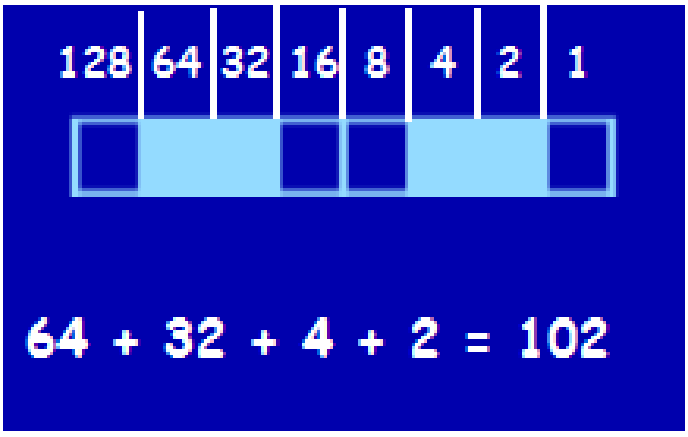
```
100 SET VIDEO MODE 0    ! 40 karakteres szöveges lap
110 SET VIDEO COLOR 1   ! 4 szín
120 SET VIDEO X 20
130 SET VIDEO Y 20
140 OPEN #1:"video:"
150 DISPLAY #1:AT 1 FROM 1 TO 20
```

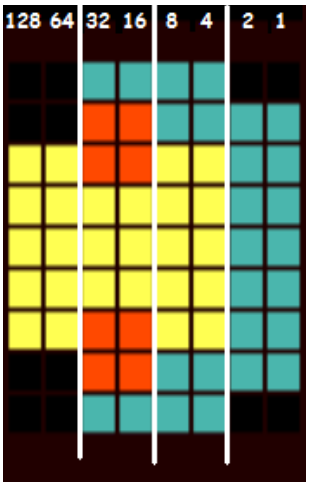
Ha a PRINT #1 utasítással karaktereket írunk erre a lapra, akkor az alap karakterkészlet olvashatatlanul fog megjelenni.

A helyzet az, hogy ezekben a grafikus üzemmódokban is karakterek jelennek meg, csak azok máshogy definiálhatók, mint a jól megszokott karakterek. A jól megszokott karakterek (melyek például a basic indulásakor tárulnak elénk) mindössze egyetlen színt használnak egyetlen karakteren belül. Binárisan itt a 0 jelenti a háttér színét, az 1 a karakter színét, és ilyen karaktereket a demokazettáról is ismert (bár azóta valami furcsa figura által továbbfejlesztett) karaktertervező programmal is lehet tervezni. Ezek a karakterek a kétszínű (SET VIDEO MODE 0, ha 40 karakter van egy képernyőn és SET VIDEO MODE 2, ha 80 karakter) üzemmód használatára valók:

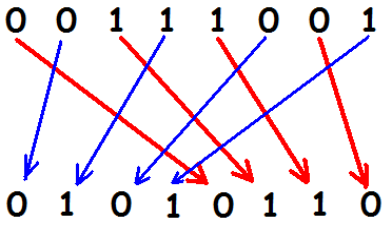


Négyszínű karakteres üzemmódban egy karakteren belül négy színt is lehet használni, egy háttérszínt, és ezen kívül 3 másik színt. Azonban, míg a jól megszokott, kétszínű karakterek 8 pixel szélesek, addig a négyszínű karakterek csak 4 pixel szélesek, tehát a színpompát a felbontás rovására vehetjük igénybe. Az ilyen karakterek definiálásához tehát nem 8, hanem 4 pixelhez kell megadni az adatokat, viszont pixelenként a színt is meg kell adnunk. Mivel egy biten csak két színt lehet tárolni (0 a háttérszín és 1 a tinta színe), a 4 színhez két bit kell. Így egy pixel színét két bit határozza meg, 00 a háttér színe, 01 az első szín, 10 a második szín és 11 a harmadik szín. Ezek a színek bármilyen színek lehetnek, természetesen, ezt a SET PALETTE utasítással tudjuk beállítani. A normál karaktereknél 8 pixelt kell megadni soronként, a négyszínű karaktereknél csak 4 pixelt kell megadni, viszont a színeket két bit tárolja, ezért mindkét karaktertípusnál ugyanúgy 8 biten tárolható egy karakter egy sora:





A bemutatott példán a karaktersor első pixele hattérszínű (bitekben megadva 00), a következő pixel a 3. szín (ez bitekben megadva 11), az azt következő szín a 2. szín (bitekben megadva 10), az utolsó szín pedig az 1. szín (bitekben 01). Ezt a bitsorozatot összeolvasva a következőt kapjuk:



00111001

Azonban, a kijelzésre kerülő bitek sorrendje nem így generálódik, hanem az egyes képpontok színét meghatározó bitek kevert sorrendben vannak, akárcsak a grafikus módoknál. Minden bitpár kiválaszt egy színt a 0-ás, 1-es, 2-es, 3-as paletta színek közül:

eredeti bitsorrend: b7,b6 b5,b4 b3,b2 b1,b0
módosított bitsorrend: b3,b7 b2,b6 b1,b5 b0,b4

Ez azt jelenti, hogy például a fenti bitsorozat (00111001) bitjeinek a sorrendjét meg kell változtatni a következőképpen: Gyakorlatilag jobbról, a második számjegytől indulva minden második számjegyet át kell helyezni az új bitsorozat jobb oldalára. Ha elértünk bal oldalra a végére, akkor az eredeti bitsorozat elejéről, jobbról indulva újra minden kimardt második számjegyet hozzá kell raknunk az új bitsorozathoz, jobbról balra haladva Így kapjuk az eredet 00111001 bitsorozatból a 01010110 bitsorozatot. Ez utóbbit át kell váltani kettes számrendszerből tízes számrendszerbe:

1x0 + 2x1 + 4x1 + 8x0 + 16x1 + 32x0 + 64x1 + 128 x0 = 86
vagy: BIN(01010110) = 86

Így, ha a fenti ábrán látható módon akarjuk beállítani a négy pixel színét egy karaktersorban, akkor a SET CHARACTER utasításba 86-ot kell megadni az adott sorhoz.

A négyszínű karakteres grafikus mód igen jól használható játékprogramokhoz. Ezt a videomódot használja az utóbbi években elkészült Bricky Prise, a Treasure Cave, vagy a Hamika folytatása, az Entersnake is. Négyszínű karaktereket jelenleg Endi Gracha Editorával szerkeszthetünk, a mindenki által jól ismert, hagyományos, négyzetrácsos papíron kívül. Mivel négy pixel szélességben nem igazán lehet sajnos érdemben rajzolni, leginkább a 2x2 karakter szélességű elemek mutatnak jól. Karakteres animációt is használhatunk ebben az üzemmódban, így eléggé látványos dolgokat lehetne kihozni belőle még basic-ben írt, Zzzippel fordított programmal is. Sok lehetőség rejlik ebben az üzemmódban, amiből még nem sokat használtunk ki eddig.

Ha nem az adott karakternek megfelelő üzemmódban printeljük a karaktert a képernyőre, krixkrax fog megjelenni, ami esetleg nyomokban emlékeztet az eredeti karakterre. Hiszen ilyenkor a karakterek pixeleit ill. a színeket tároló bitek más értelmezést kapnak. Ezért találkozott a gyanútlan felhasználó is krixkraxokkal a rendszerváltás környékén, amikor a SET VIDEO MODE és SET VIDEO COLOR utasításokkal kísérletezett.

A 16 színű és a 256 színű karakteres módokról következő számunkban lesz szó.

Feasibility Experiment

1983 - Digital Fantasia
kaland, szöveges

Írta: Kiss László (Lacika)

A Mysterious Adventures sorozat 5 részében Brian Howarth visszanyúlt a gyökerekhez: az Adventureland-et utánozza (melyből később ő maga készítette a grafikus verziót). Egy meglehetősen furcsa világban, ahol keverednek a műfajok és az időjárási viszonyok, egyszerűnek gondolható feladatunk van: 10 értékes tárgyat kell összegyűjtenünk a királyság elrejtett zugaiból, egy, a játék elején még ismeretlen helyszínre. Újdonság, hogy a feladatot korlátozott időn - 698 kör (vagyis kiadott parancs) - belül kell teljesítenünk. A program bizonyos időnként kijelzi a hátralévő körök számát, és sürgető üzeneteket küld. A birodalom meglehetősen sok helyszínt ölel fel, viszont egyszerre csak 5 tárgyat bírunk el. Így az amúgy nem túl összetett játékmenetet a tárgyak ide-oda hurcolása teszi hosszabbá. A minket akadályozó ellenfeleket ezúttal nem kicselezni kell, hanem legyőzni. Erre viszont csak a megfelelő fegyverrel vagyunk képesek, pl. egy sárkány előtt ne kezdjünk el egy rövid karddal hadonászni, mert csak nevetség tárgyát fogjuk képezni...

A 10 megszerzendő értékes tárgy (melynek egy része fegyverként is használatos):

BOX OF EMERALDS,
CRYSTAL SCIMITAR,
EBONY SPEAR,
GOLD AMULET,
IVORY STATUETTE,
IXION SHIELD,
LARGE RUBY,
ORNATE DAGGER,
RED DIAMOND,
SILVER CHALICE

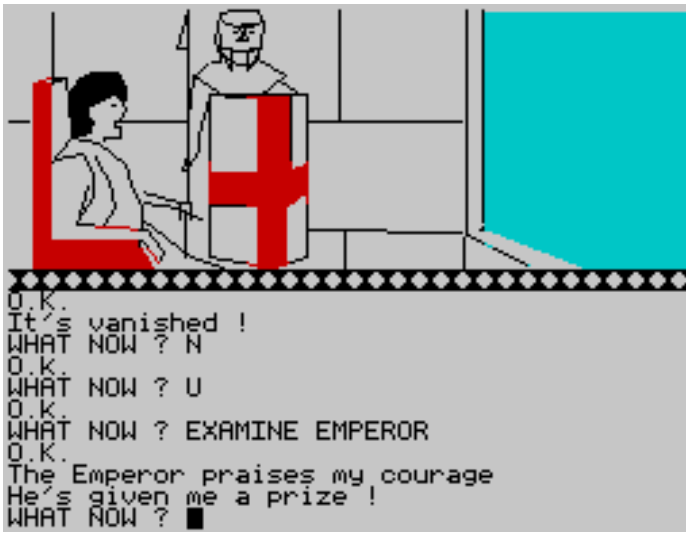
A program „legérdekesebb" vonása, hogy igencsak bugos - ez az újrakiadásra is igaz - , egészen pontosan sok helyszín képe helyett csak villogó krikszkrakszot látunk. Sőt olyan helyszín is van, melynek kirajzolása után kiakad a program. Kikapcsolt grafikával viszont tökéletesen játszható (a képek amúgy is alulmúlják a sorozat színvonalát).
A program használata - a grafikai mizériát leszámítva - teljes mértékben megegyezik a sorozatban megszokottakkal. A játék jellegéből adódóan működik a SCORE parancs: az elérhető maximális pontszám száz, vagyis minden, a helyére vitt kincsért 10 pontot kapunk. A LOOK és az EXAMINE parancsot viszont megkülönbözteti a program! Az EXAMINE parancsot érdemes (kell) használni, azzal érhetünk el biztos sikert.

A program szókészlete:

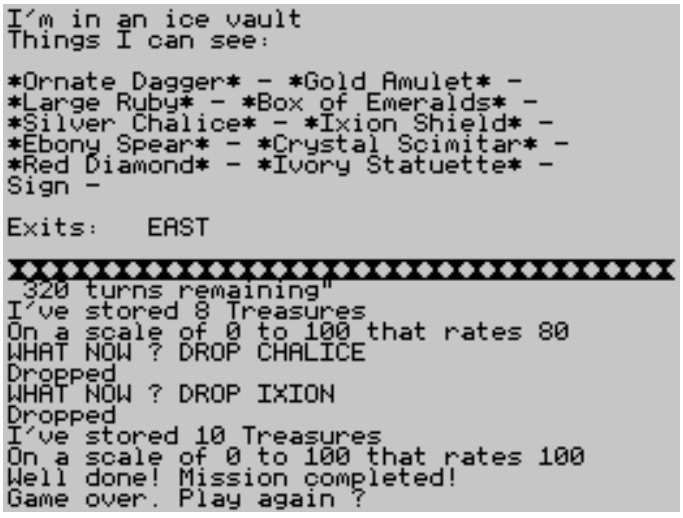
1
2
3
4
5
ALTAR
AMULET
ATTACK
BLACK
BLOCK
BOX
BRAZIER
CABINET
CAN
CAVE
CHALICE
CLIMB
CLOTH
CREVICE
DAGGER
DIAMOND
DIG
DRAGON
DROP
EBONY
EMERALDS
EMPEROR
EXAMINE
FILL
FIX
FLINT
GAUNTLETS
GET
GLADIATOR
GRILLE
GUARD
HUTS
ICE
INSCRIPTION
IRON
IVORY
IXION
KEY
KILL
LAMP
LEATHER
LION
LOOK
MOVE
OPEN
PATH
PLAIN
PLAITED
PRAY
PRESS
PUSH
RED
READ
ROCKS
ROPE
RUBY
SCIMITAR
SCORE
SEARCH
SHAPES
SHIELD
SHOVEL
SIGN
SKELETON
SLAB
SPEAR
STATUETTE
STEPS
STONE
SWORD
TAKE
TIE
TO
TREE
TUNNEL
UNLIGHT
UNLOCK
USE
WAIT
WINDOW
WITH

A megoldás:

(Halovány foltok között találjuk magunkat, mivel mindenki látja mi van az 1. képen, nem is részletezem...) W (egy kastély udvarán állunk), N, EXAMINE ROCKS, GET DAGGER, N, W, EXAMINE CREVICE (a hasadékban egy kovakövet találunk, amely lámpagyújtáshoz ideális szerszám), GET FLINT, E, N, N, N (Egy hasadék szélén egy megégett fát látunk), EXAMINE TREE, GET BLOCK, S, S, S, S, S (visszaértünk a kastély udvarára), E, E, D, N, GET RUBY, E, E, (a gladiátorok szobájában lévő pajzs kiváló védelmet nyújt az oroszlánokkal szemben) GET SHIELD, W, N, E, E, S, E, E, (egy kőkorszaki faluba érkeztünk...) N (most meg egy gleccseren állunk, egy jégbarlang előtt...), DROP RUBY, DROP DAGGER, S, E, GET ROPE, W, S, E, E (egy bánya előterében állunk), S, (a bánya raktárhelyiségében egy olajlámpát találunk) GET LAMP, N, E, N, DROP FLINT, DROP BLOCK, DROP LAMP, (A bánya 5 emelete között lifttel közlekedhetünk.



A megfelelő számozott billentyű megnyomásával juthatunk a kívánt szintre.) PRESS 2, E, N, GET SHOVEL, S, W, PRESS 1, S, W, W, W, N, W, W, N, W, W, S, S, S, U, W, W, N, N, N, N, N, TIE ROPE, TO TREE, CLIMB ROPE (a fához rögzített kötélen lemászunk a hasadékba), N, EXAMINE SKELETON, GET AMULET, W, CLIMB ROPE, S, S, S, DIG, USE SHOVEL, USE SHOVEL (kétszer ásunk), GET KEY, S, S, OPEN DOOR (az imént kiásott kulccsal kinyitjuk az ajtót), DROP KEY, GO DOOR, GET CAN, EXAMINE CAN (a kannában olaj van), E, E, E, GO WINDOW, GET STATUETTE, S, E, N, E, E, E, S, S, E, E, N, DROP STATUETTE, S, S, S, DIG, USE SHOVEL, MOVE SLAB (elmozdítjuk a deszkaborítást, alatta lépcső vezet lefelé), GO STEPS, D, W, GET GAUNTLETS, EXAMINE ALTAR, READ INSCRIPTION, DROP AMULET, E, U, U, N, N, W, W, N, N, W, W, W, S, S, S, U, W, W, WEAR GAUNTLETS, GET BRAZIER (a páncélkesztyű nélkül a forró szénserpenyő érintése halálos), DROP SHOVEL, N, N, GET BOX, S, S, E, E, D, N, E, N, E, E, S, E, E, N, GO CAVE, DROP BRAZIER, WAIT (a forró szénserpenyő alagutat olvaszt a jégben), GET BRAZIER, GO TUNNEL, DROP BRAZIER, WAIT, GET CLOTH, DROP GAUNTLETS, W, READ SIGN (a jelzés szerint ez a kincsek gyűjtőhelye), DROP BOX, E, S, S, GET RUBY, GET STATUETTE, GO CAVE, GO TUNNEL, W, DROP RUBY, DROP STATUETTE, E, S, S, GET DAGGER, S, S, E, E, E, N, GET LAMP, EXAMINE LAMP (hiányzik a kanóc a lámpából), FIX LAMP (a ruhaanyag csíkkal „megjavítjuk" a lámpát), FILL LAMP (feltöltjük a lámpát olajjal), DROP CAN, GET FLINT, GET BLOCK, PRESS 4, LIGHT LAMP (a kovakővel meggyújtjuk a lámpát), DROP FLINT, DROP BLOCK, E, E (egy kiállítóterembe jutottunk, az őr nem enged a vitrin közelébe), KILL GUARD (a törrel), EXAMINE CABINET, GET DIAMOND, W, W, UNLIGHT LAMP, PRESS 1, DROP LAMP, S, W, W, W, S, GO STEPS, D, W, GET AMULET, PRAY, GET SCIMITAR, E, U, U, N, N, N, GO CAVE, GO TUNNEL, W, DROP AMULET, DROP DIAMOND, DROP DAGGER, E, S, S, S, W, W, N, N, W, W, W, S, KILL LION (a kristály handzsárral), N, U, EXAMINE EMPEROR (az uralkodó megjutalmaz), DROP SHIELD, GET IXION, D, S, S, S, GET SWORD, U, W, W, GET SHOVEL, E, E, D, N, N, N, E, E, KILL GLADIATOR (a karddal), DIG, USE SHOVEL, DROP SHOVEL (egy fekete kulcsot ásunk ki), GET BLACK, E, S, S, E, E, N, GO CAVE, GO TUNNEL, DROP SWORD, W, DROP SCIMITAR, E, S, S, S, S, E, E, E, N, GET LAMP, PRESS 4, GET FLINT, GET BLOCK, LIGHT LAMP, PRESS 5, E, OPEN DOOR (a fekete kulccsal), GO DOOR, DROP BLOCK, GET SPEAR, W, W, PRESS 1, S, W, W, W, N, W, W, N, N, W, W, W, S, S, S, U, W, W, N, N, N, N, N, CLIMB ROPE, N, N, E, KILL DRAGON (az ébenfa dárda és az Ixion pajzs kell hozzá), DROP LAMP, GET CHALICE, S, S, W, CLIMB ROPE, S, S, S, S, S, E, E, D, N, N, N, E, E, E, S, S, E, E, N, GO CAVE, GO TUNNEL, W, DROP SPEAR, DROP CHALICE, DROP IXION



dBase II. 2.43 (IS-DOS) – IV. rész

5. Adatbázis-állományok használata

A hosszú előkészítés után ebben a fejezetben megismerkedünk azokkal a legfontosabb parancsokkal, melyek lehetővé teszik egy adatbázis használatát. Bemutatjuk az adatbázis-állomány készítési módját, a rekordok feltöltését, módosítását. Megismerkedünk néhány adatmegjelenítési lehetőséggel.

5.1. Adatbázis-állomány létrehozása

Adatbázis-állomány felépítéséhez többféle parancs közül választhatunk, ám ezek között csupán egyetlen olyan van, amellyel az indításkor mintegy a „semmiből" készíthetünk új állományt. Ez pedig a

CREATE <állománynév>

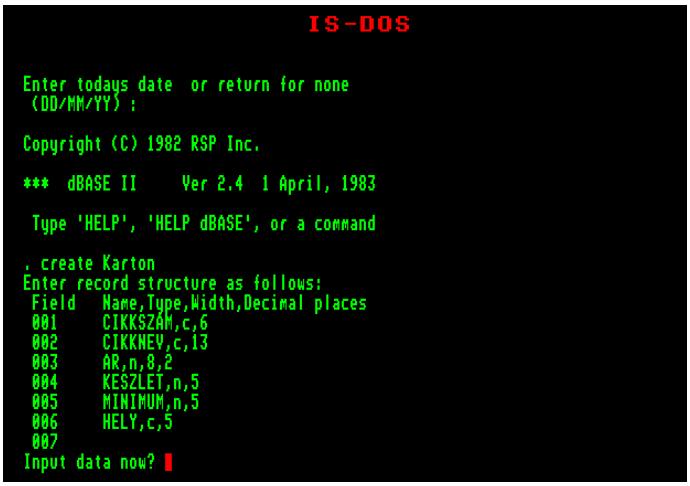
parancs. Példaként adjuk ki a

CREATE karton

parancsot. (A „karton" a lemezes állomány neve lesz.) Körültekintően járjunk el a parancs kiadásakor, mert ha olyan állomány nevet választunk ami már létezik a lemezen, azt kérdezés nélkül felülírja a dBase!
Ha az állománynévben nem adunk meg kiterjesztést, adatbázisunk automatikusan „.DBF" típusjelet kap. Ehelyett bármilyen hárombetűs jelzést adhatunk neki, de akkor azt mindig le kell írni, ha erre a lemezes állományra szeretnénk hivatkozni.
A CREATE parancs a teljesképernyő-szerkesztő parancsokhoz hasonlóan működik. Egymás után kell megadnunk a leendő adatbázisunk mezőinek pontos definícióját. Egy mező különböző tulajdonságai egy sorba kerülnek. Az adatbázisok szerkezetének leírásakor ismertetet szabályok szerint kell megadnunk:

- a mező nevét
- a mező típusát (egyetlen karaktert, a típus kezdőbetűjét kell leütni)
- a mező szélességét - csak karakteres és numerikus típusnál van értelme
- a tizedesjegyek számát - ha numerikus mezőt definiáltunk

Bármilyen hibás (nem megengedett) karaktert vagy értéket írunk be definícióba, azonnal figyelmeztetést kapunk, és amíg ki nem javítottuk nem mehetünk tovább. Az egyes elemeket (a mintának megfelelően vesszővel kell elválasztani.)
A CREATE parancs végrehajtását és ezzel az új adatbázis-állomány létrehozását befejezhetjük egy üres mezőnév bevitelével (‚ENTER' megnyomása), ekkor valóban lemezre kerül az új állomány.



Már most jegyezzük meg, hogy a dBASE használata során bármikor, ha úgy érezzük, nem akarjuk a kiadott parancsot befejeztetni, esetleg az éppen kiadás előtt álló parancsot szeretnénk „elfelejteni", semmissé tenni, nyomjuk meg az ‚ESC' feliratú billentyűt („escape" = szökés, menekülés). Néhány esetben a módosítások elmentése még az ‚ESC' segítségével sem úszható meg. Ezekre az adott helyen mindig külön felhívjuk a figyelmet, ‚de nem árt már most megjegyezni, hogy általában azok az információk, `amelyek egyszer már lekerültek a képernyőről - hogy továbbiak jelenhessenek meg - rögtön tárolódtak is.
Az adatbázis-szerkezet lemezre mentése után még egy kérdésre kell válaszolnunk: „Input data now? (Y / N)" (= Most töltünk be rekordokat?). Ha az ‚Y' leütésével jelezzük adatbeviteli szándékunkat, akkor a dBASE egymás után, rekordonként nevükkel együtt kiírja az üres ‚mezőket, melyeket teljesképernyő-szerkesztő módon kitölthetünk. A rekordok felvitelének befejezése egy üres rekord bevitelével vagy a ‚CTRL'+'Q' billentyűk megnyomásával történik.
Később, ha már kezelni tudjuk létrehozott adatbázis-állományainkat, visszatérünk az adatbázis-készítés további lehetőségeire (ld. az „Adatbázis készítése meglévő állományok felhasználásával" című fejezetet). Ne feledjük el, hogy a CREATE parancs nem alkalmas régi állomány módosítására!

5.2. Adatbázis használatba vétele

A létrehozott, lemezen tárolt állományainkat használat előtt meg kell nyitni, mielőtt tartalmát lekérdeznénk vagy módosítanánk. Adatbázis megnyitására egyetlen parancs használható:

USE <adatbázisállomány-név>

Ez a parancs „.DBF'" típusjelet feltételez, ha nem adunk meg mást, és csak adatbázis megnyitására alkalmas. Egy munkaterületen csak egy adatbázis fér el, így egy másik állomány azaz egy új USE parancs - lezárja az előzőleg megnyitottat. Több állomány egyidejű nyitva tartásához

minden új ESE előtt a SELECT parancs segítségével át kell költöznünk a másik munkaterületre.
A különböző területen megnyitott állományaink tartalmához bárhonnan hozzáférünk, de a rekordmutatót csak a kiválasztott munkaterületen tudjuk mozgatni. Ezért ezt megkülönböztetésül „aktív" adatbázis-állománynak is nevezzük.
Az adatbázis-állományokban vannak a legfontosabb információk, a nyilvántartandó adatok, ezért ezek gondos lezárására mindig nagyon ügyeljünk. A dBASE igyekszik kevés lemezműveletet végezni, így a nyitva felejtett állományok a legközelebbi megnyitáskor nem biztos, hogy tartalmaznak minden bevitt adatot. (Az ilyen balesetek úgy kerülhetők el a legkönnyebben, hogy betartjuk a szabályt, és mindig QUIIT paranccsal lépünk ki az adatbázis-kezelőből. Ez ugyanis mind szükséges információt felvezet a lemezekre is, mielőtt visszalépne operációs rendszerbe.)
Egy adatbázis-állomány lezárása azt is jelenti, hogy azok a segédállományok, amelyek szorosan hozzá tartoznak, azaz nélküle nem létezhetnek, szintén lezáródnak. (Ilyen az index- és a formátumállomány valamint ide tartozik a memória-állomány is.)
Az aktuális, kiválasztott munkaterületen megnyitott adatbázist a

USE

parancs zárja le.

Az adatbázisok megfelelő használata érdekében nem engedhetjük meg magunknak, hogy elfelejtsük, milyen szerkezettel hoztuk létre őket. Ha a körülmények szerencsétlen összejátszása miatt ez mégis megtörténik, könnyen kaphatunk hibaüzenetet. Ilyenkor nagyon hasznos, ha kiíratjuk magunknak újra a pontos szerkezetleírást. Ezt valósítja meg a

DISPLAY STRUCTURE

parancs, amely az aktív adatbázisról nyújt bőséges információt.
A parancs kiírta az adatbázis nevét, rekordjainak számát, az utolsó -aktualizálás dátumát, az egyes mezők nevét, típusát, hosszát és egy rekord teljes hosszát. Ne ijedjünk meg, a számítógép nem felejtett el összeadni. Egy rekord teljes hossza egy byte-tal mindig nagyobb, mint a mezők hosszának összege. A dBASE ezt az egy karakternyi helyet is tárolásra használja. Hogy pontosan mit ír bele, azt az adatrekordok törléséről szóló fejezetben áruljuk el.

5.3. A rekordmutató egyszerű mozgatása az adatbázisban

Egy új adatbázis készítése és azonnali feltöltése után a rekordmutató az adatbázis végén van, egy lemezen található állomány megnyitása után pedig a legelső rekordon. Az adatbázis rekordjaira vonatkozó parancsaink következményeként a rekordmutató legtöbbször elmozdul előző helyéről, és az utolsó „kézbevett" rekordra áll. A rekordmutatót azonban mi magunk is beállíthatjuk egy adott helyre a következő parancsok valamelyikével:

GOTO <n>
GO <n>
<n>

forma közvetlenül az „n"-edik sorszámú rekordra állítja a rekordmutatót
A rekordok sorszámukat a rögzítés során, folyamatosan emelkedő rendben kapják, ezt a rögzítési sorrendet nevezzük fizikai sorrendnek. Az adatbázisok rendezése után egy ettől általában eltérő logikai rendet is felvehetnek a rekordok. A logikai rendben másként is lehet mozgatni a rekordmutatót, erre a rendezési lehetőségeknél visszatérünk.

5.4. Adatok megjelenítése az adatbázisból

Rekordok megjelenítésére alapvetően két parancs áll rendelkezésünkre. Ezek egymástól az alapparancsban különböznek, de megadható paraméterek ugyanazok. Az egyik parancs folyamatos listázza a megadott rekordokat (LIST kulcsszóval), a másik pedig képernyő oldalanként megáll és megvárja, hogy elolvassuk a monitorra kiírt információkat (DISPLAY kulcsszóval). Az interaktív munka során utóbbit használjuk gyakrabban, éppen működési módja miatt.
A két parancs sok lehetséges paramétere közül egyesével nézzök végig a fontosabbakat. (Ezt a módszert később is követni fogjuk, bonyolultabb parancsok ismertetésénél.)
A legfontosabb paraméter az érvényességi kör, mely a parancs formájából láthatóan elhagyható. Hatására az érvényességi körbe eső rekordokat a képernyőre írja a rendszer.

DISPLAY [<érvényességi kör>]

Nézzünk végig a lehetséges érvényességi kört:
DISPLAY az aktuális rekordot (amin a rekordmutató áll) jeleníti meg
DISPLAY RECORD x a x. rekordot jeleníti meg (a rekordmutató az x. rekordra áll)
DISPLAY NEXT x
az aktuális rekordtól x számú rekordott jelenít meg (a rekordmutató az utolsó megjelenített rekordra áll)
DISPLAY ALL az adatbázis összes rekordját megjeleníti (a rekordmutató az utolsó rekordra áll)

Gyakran előfordul, hogy a DISPLAY utasítással megjelenített rekordok nem férnek ki egy képernyőre. Ilyenkor a „képernyő végén" megjelenő felirat: „Waiting", amely - mint már láttuk - arra szólít fel bennünket, hogy a listázás folytatásához nyomjunk meg egy tetszőleges billentyűt.
Bármely példában megadhatjuk a LIST kulcsszót a DISPLAY helyett. A LIST érvényességi körének alapértelmezése „ALL", azaz minden rekordot kiír. További különbség, hogy folyamatosan, megállás nélkül listáz.
Mindkét parancs táblázatosan jeleníti meg a rekordok tartalmát. Az első oszlopban olvasható rekordsorszámok a rekordok fizikai (rögzítési) sorrendjét tükrözik. Ez sokszor nagyon lényeges információ, ezért ennek kikapcsolásáról minden egyes listázási parancsban külön gondoskodnunk a DISPLAY és LIST parancsok „OFF" paraméter szerepeltetésével:

LIST OFF
DISPLAY OFF

Egy olyan lista készítése után, ahol az adatbázis végéig kell kiírni rekordokat („ALL" érvényességi körrel), a rekordmutató az adatbázis végére mutat. Ha megadjuk, hogy egy konkrét rekordot írjon ki (DISPLAY RECORD <n>), akkor a rekordmutató visszakerül az adatbázis belsejébe. Ugyanezt elérhetjük a GOTO paranccsal is.

A DISPLAY és LIST parancsban paraméterként megadható kifejezésekből álló lista is a következő módon:

LIST [<kifejezéslista>]

Ez nyújt lehetőséget annak a gyakori igénynek a megoldására, hogy az aktív adatbázis mezői közül csak néhányat jelenítsünk meg (a mezőnév önmagában kifejezésnek számít, hiszen változó).
Ha egy állományunk például túl „széles", egy-egy rekord nem fér el a képernyőn, ezért két sorban jelenik meg. Ez a forma meglehetősen olvashatatlan. Ilyenkor elég, ha azokat a mezőket íratjuk ki, melyekre éppen kíváncsiak vagyunk.

LIST mezőnév1, mezőnév2

Használhatjuk a DISPLAY parancsot is „ALL" érvényességi körrel. Ha ez utóbbit választjuk, akkor ez már két paraméter egyidejű használatát lenti, ne felejtsük el, hogy ezek sorrendje nem lényeges. Az alábbi két parancs mindegyike helyes:

DISPLAY ALL mezőnév1, mezőnév2
DISPLAY mezőnév1, mezőnév2 ALL

Az aktuális rekord mezőire hivatkozhatunk a „?" parancsban is. Például a

? mezőnév1, mezőnév2

az aktuális rekord mezőnév1, és mezőnév2 mezőit jeleníti meg (ez egy szép mondat...)

5.4. Numerikus mezők összege

Az adatbázisból nemcsak rekordonként tudunk adatokat megjeleníteni, lehetőségünk van numerikus mezők összegének vagy átlagának kiíratására is. Egy vagy több, számokat tartalmazó oszlopot összegez a

SUM [<kifejezéslista>] [TO <memóriaváltozó-lista>][<érvényességi kör>] [FOR <feltétel>][WHILE <feltétel>]

parancs. Ha memóriaváltozóba akarjuk helyezni az eredményt, pontosan annyi nevet kell felsorolnunk, ahány numerikus kifejezést megadtunk; ha nincs kifejezéslista, akkor az aktív adatbázis minden numerikus mezőjét összegzi. A FOR és WHILE paraméterek segítségével megadott feltétel az összegzésben részt vevő rekordok körét tudjuk befolyásolni.

5.5. Válogatás az adatbázisból adott szempontok szerint

Adatbázis használatakor általában a rekordoknak csupán egy részén kívánunk elvégezni egy-egy tevékenységet. Ezek a rekordok többnyire nem férnek bele a lehetséges érvényességi körökbe, mert nem egymást követően lettek rögzítve, és így az adatbázis különböző pontjairól kell őket összeválogatni. Szerencsére a WHILE és FOR paraméterek parancsvégrehajtás közben tudják elvégezni ezt a válogatást, így egyetlen utasítás kiadásával minden megfelelő rekordon elvégeztethető a művelet.
A parancsszintaktika ismertetésénél már leírtuk, hogy e két paraméter között milyen különbségek vannak, most példákkal is illusztráljuk ezeket.

```
                    IS-DOS

Enter todays date  or return for none
 (DD/MM/YY) :03/06/03

Copyright (C) 1982 RSP Inc.

***  dBASE II     Ver 2.4  1 April, 1983

 Type 'HELP', 'HELP dBASE', or a command

. use karton
. list
00001  001101 golyostoll       23.50   125     5 12.34
00002  000213 kek tinta        34.60    50     5 12.33
00003  012677 cimke             0.90  1340   100 11.23
00004  001321 piros tinta      43.50    76     5 11.23
00005  045677 toltotoll       231.00    34     1 11.22
00006  026267 tuzogep         409.00    12     1 12.33
00007  000333 tuzokapocs        1.50   500    50 11.24
00008  002631 miltonkapocs      1.50   500    50 11.24
00009  000012 ragaszto         12.70    77    10 13.45
00010  000001 geppapir          0.50  2550   100 12.45
.
```

A „KARTON" nevű állományban ugyanarról a helyről indulva nézzük meg működésüket. Először adjunk ki egy DISPLAY parancsot FOR paraméterrel:

```
. goto 7
. display cikkszam, cikknev, ar for ar<10
00003  012677 cimke              0.90
00007  000333 tuzokapocs        1.50
00008  002631 miltonkapocs      1.50
00010  000001 geppapir          0.50
.
```

Minden olyan rekordból kiírta a megadott három mezőt, amely kielégíti a feltételt, azaz „ar" nevű mezőjében egy tíznél kisebb szám van (anélkül, hogy külön kiírtuk volna, az érvényességi kör nem csupán az aktuális rekord, ez is a FOR paraméter műve). A rekordmutató az állomány végére került.
Most ugyanolyan feltételekkel, de WHILE paramétert szerepeltetve hajtassuk végre a parancsot:

```
. goto 7
. display cikkszam,cikknev,ar while ar<10
00007  000333 tuzokapocs        1.50
00008  002631 miltonkapocs      1.50
.
```

A rekordmutató ugyanott állt, feltételnek ugyanazt adtuk meg, az eredmény mégis más, nem vizsgálta meg az aktuális rekord előtti részt, és a kilences rekordon megállt (ez az első, amelyre nem teljesül a feltétel, a továbbiakban ez az aktuális rekord). Vegyük észre, hogy ezek paraméterek is csereberélhetők.

Válogatás ürügyén remek alkalmunk nyílik néhány különleges függvény, illetve művelet bemutatására, hiszen a WHILE és FOR paraméterekben megadott feltételek szerepelhetnek sok más parancs után is.
Tekintettel arra, hogy a leggyakrabban karakteres mezővel dolgozunk, ebben kell a legtöbbször válogatni is. Ezzel kapcsolatban legfontosabb, amit meg kell jegyeznünk, hogy a számítógép számára pl. a „b" és a „B" betűk annyira különböznek egymástól, mint mondjuk egy újszülött víziló a saját anyjától: több közös vonásuk van, de semmiképp sem tekinthetők azonosnak.

Ha rákeresnénk a „címke" cikknévre a következő utasítással:

LIST FOR CIKKNEV="CIMKE"

Akkor bizony a „címke" nem jelenik meg, mivel az adatbázisunkban kisbetűvel írtuk, és mi nagybetűvel kerestünk rá. A probléma megoldására használhatjuk a nagybetűssé konvertáló „!" függvényt:

LIST FOR !(CIKKNEV)="CIMKE"

(Az egyenlőség megvizsgálása előtt a „cikknev" mező tartalmát a dBASE „fejben" átalakítja nagybetűssé és így végzi el az összehasonlítást. A mező tényleges tartalma nem változik.)

Természetesen senki nem akadályoz meg benne, hogy akár összevont feltételeket is vizsgáljunk a logikai ÉS / VAGY fügvények segítségével:

LIST FOR CIKKNEV="címke" .or. CIKKNEV="tuzogep"
LIST FOR HELY="11.23" .AND. AR>10

Nagyon hasznos lehet szöveges adatok keresésénél a „$" relációs operátor, amellyel rész-karakterláncokat kereshetünk. Keressük meg például, milyen tintáink vannak készleten (melyik cikknévben szerepel a „tinta" karakterlánc):

LIST CIKKNEV,AR FOR „tinta" $ CIKKNEV

A „$" fügvény arra is alkalmas, hogy egy string tetszőleges részét kiolvassuk:

$ (KIF. vagy VÁLTOZÓ vagy KARAKTERLÁNC,KEZDET,-HOSSZ)

Példa: ? $(„Enterprise", 5,4)

Részkarakter-láncokat kereshetünk egy tetszőleges kifejezésben a „@" fügvénnyel:

@ (karakterlánc1,karakterlánc2)
@ (változó1,változó2)

Például a ? @ („ABLAK","KISABLAK") kifejezés értéke 4 lesz.

Figyelem!!! Attól függően, hogy milyen karakterkészletet használunk, lehet, hogy nem találjuk a „@" jelet, ez ne zavarjon minket, helyette a nagy ‚Á' betűt kell használni, amit PC billentyűzetről az ‚SHIFT+0' megnyomásával csalhatunk elő. A @ karakter használatára később is szükségünk lesz, tehát ezt jól jegyezzük meg.

5.6. Az adatbázis karbantartása

Az adatbázisok használatakor szinte naponta kell meglévő rekordjainkat módosítani, esetleg egyet-egyet megszűntetni, vagy új rekordokat az adatbázisba illeszteni, azaz a rekordokat naprakész állapotban kell tartani. Ezt a tevékenységet nevezhetjük az adatbázis karbantartásának.

5.6.1. Rekordok hozzáfűzése, módosítása
Az adatbázis végéhez az

APPEND

parancs segítségével új rekordokat fűzhetünk (a parancs végrehajtása kísértetiesen hasonló a CREATE parancsban végrehajtott rekordfeltöltéshez).
Az adatbázis belsejébe új rekordot szúrhatunk be a

INSERT [BEFORE] [BLANK]

panccsal. Opció nélkül az aktuális rekord mögé helyezi el az új rekordot. BEFORE opcióval (before = előtt) az aktuális rekord elé kerül az rekord (a rekordmutató nem mozdul el, a mögötte elhelyezkedő rekordok eggyel hátrébb lépnek, eggyel megnő a sorszámuk). A beszúrás után rögtön feltölthetjük adatokkal a rekordot (hasonlóan az APPEND parancsnál látottakhoz). Amennyiben nem kívánjuk az új rekordot adatokkal feltölteni, csak egy üres rekordot kívánunk beszúrni az adatbázisunka, használjuk a BLANK opciót.

Létező rekordjainkat az EDIT vagy CHANGE parancs segítségével módosíthatjuk, a két parancs azonos paraméterekkel rendelkezhet, működésük némileg eltérő. Az EDIT utasítással teljes képernyős módban szerkeszthetjük a rekordot:

EDIT - a kívánt rekordtól kezdjük a szerkesztést és folytathatjuk addig, amíg ki nem lépünk belőle (CTRL+Q) (Nem támogatja a dBase II összes változata!)
EDIT x - az x rekordtól kezdjük a szerkesztést.

A CHANGE parancs használata némileg összetettebb. Itt nem, teljes képernyős módban szerkeszthetjük a feltételeinknek megfelelő rekordjaink kívánt mezőjét:

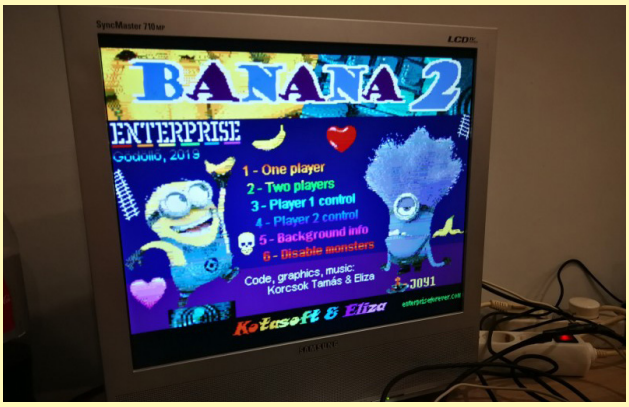
CHANGE [<érvényességi kör>] FIELD <mezőlista>

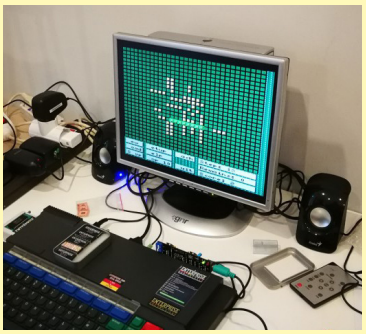
A parancsban használható a WHILE és FOR paraméter. Vegyük észre, hogy a FIELD szó nem hagyható el! Ezzel az utasítással a „KARTON" állományunkban könnyen módosíthatjuk a 100Ft-nál drágább termékek árát:

CHANGE FOR AR>100 FIELD AR

Folytatjuk!

Enterprise Klub 2019. március 2.

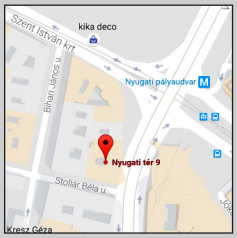




ENTERPRISE KLUB

Egy évben 8 alkalommal

Helyszín:
Nyugati Oktatási Központ, Skála terem
Budapest (V. ker.)
Nyugati tér 9.
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 - 2019/3-4. május-augusztus
Főszerkesztő: Matusa István
Szerkesztőségi főmunkatárs: Németh Zoltán (Zozosoft)
A csapat: geco, Povi, Kiss László, SzörG, szipucsu, lgb, Bakó Róbert, Tamási Istvánné, Kőszegi Ádám, Virág Attila
Design, nyomdai előkészítés: Matusa István
Weboldal: http://enterpress.news.hu
E-mail: info@enterpress.news.hu
A lap időszakosan - korlátozott példányszámban - nyomtatott formátumban és elektronikus formában is megjelenik.
ENTERPRESS e-magazinok:
http://enterpress.news.hu/index.php/magazin