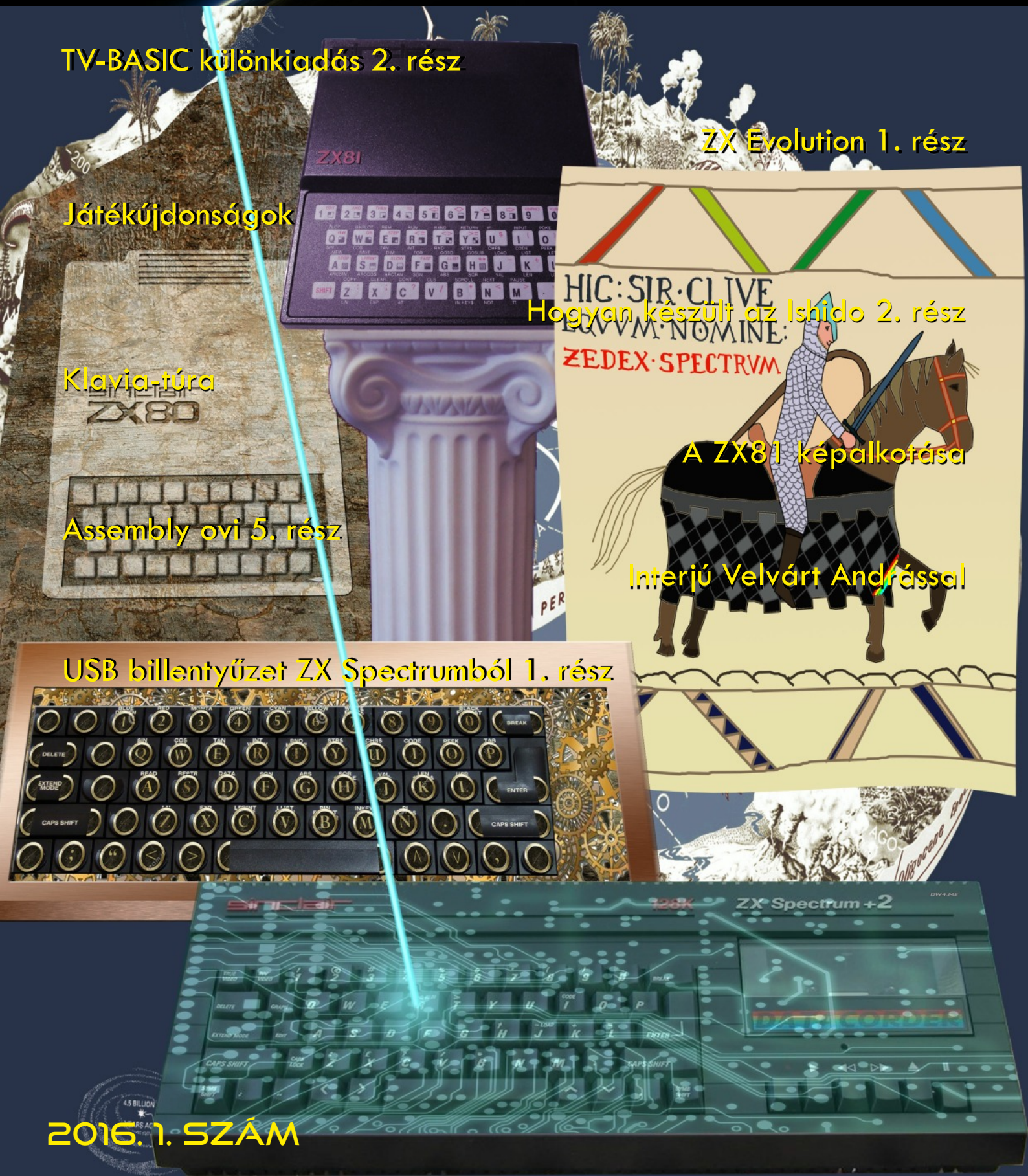
# SPECCYALISTA VILÁG

A SPECCYALISTA BARÁTI KÖR LAPJA

SINCLAIR.HU



TV-BASIC különkiadás 2. rész

ZX Evolution 1. rész

Játékújdonságok

Hogyan készült az Ishido 2. rész

Klavia-túra

A ZX81 képalkotása

Assembly ovi 5. rész

Interjú Velvárt Andrással

USB billentyűzet ZX Spectrumból 1. rész

2016.1. SZÁM

# Tartalom

# Beköszöntő

**Kedves olvasó!**

Végre itt a harmadik szám! Bár szerettünk volna korábban megjelenni, azonban végül kivártunk a mai nevezetes napig. Hisz a mai napon lett 34 éves jó öreg gumigombos dobozkánk, így mi ezzel a számmal kívánunk Boldog szülinapot! Bár külön szülinapos cikkel most nem készültünk, de így szerettünk volna legalább jelezni, hogy még 34 év távlatából is igyekszünk mindennapi életünkbe becsempészni őkelmét.

Most is igyekeztünk az egykori Spectrum Világ szellemiségét megőrizni és minden olvasónknak adni valami számára érdekes olvasmányt. Valójában ezért is ez a kis csúszás, megpróbáltuk összevárni, hogy minden témából szolgálhassunk valami érdekességgel.

Szellemiség alatt leginkább a tartalmi szálat gondoljuk, melyet igyekszünk egy letisztultabb, de mégis mai köntösbe öltöztetni. Ezt ugyan páran negatívumként rójják fel nekünk, de talán többen vannak azok, akik így szívesebben nyúlnak ezen témák után. Így aztán most egy kicsit igyekeztünk a választott Serif PagePlus kiadványszerkesztővel behatóbban ismerkedni.

A szerkesztőségünk címére ugyan kevés visszajelzés érkezett, ugyanakkor a fórumon és a különféle facebook csoportokban elcsípett visszhangok alapján azért még mindig úgy látjuk, hogy az alapkoncepciónk találkozott az olvasóink igényeivel, így ezt a vonalat igyekszünk továbbra is képviselni.

Nagy örömömmel köszönthettünk új szerzőket is, ráadásul így újabb állandó rovatoknak és cikksorozatoknak is örülhetünk. Ebben a számban elkezdjük feszegetni a klónozást és ismerkedni az egyes lehetőségekkel, megvalósult fejlesztésekkel.
A hardverészeknek továbbra is szeretnénk kedvezni, ezért a berkeinken belül folyó munkákba ismét engedünk némi betekintést, amihez most is Jocó nyújtott hathatós segítséget.

A Spectrumos témák mellett már ez előző számban elkezdtünk szemezgetni a ZX81-gyel foglalkozó írásokból, így van ez most is. Az elkövetkező számokban viszont talán sikerül bővítenünk a sort a Spectrum Világból megszokott Enterprise vonallal.
Reméljük a „lap” továbbra is elnyeri tetszéseteket és lesznek, akik szívesen bekapcsolódnak a további munkába akár csak egy-egy cikk erejéig. Ugyanakkor mindenképp várjuk véleményeteket, ötleteiteket melyekkel jobbá, érdekesebbé tehetjük a mi kis kiadványunkat.

Ezúton szeretném ismét megköszönni a szerkesztőségünk szerzőinek és munkatársainak azt a sok munkát, melyet befektettek ezen újabb szám elkészítésébe.

2016. április 23.

***Kardos Balázs (Balee)***

# Impresszum


**Főszerkesztő:** Kardos Balázs (Balee)

**Szerkesztés, tördelés: Kardos Balázs (Balee)**

**Grafika:** Molnár Péter (Mopi)

**Lektorálás:** Sári Gábor (Saga)

**Rovatvezetők:** Böszörményi Zoltán (Zboszor), Buzogány Csaba (Makranc), Mezei Róbert (M/ZX), Tanács Imre (Kapitány), Kardos Balázs (Balee), Lakatos Péter (Latyi.ca), Pgyuri, Taletovics Dávid (G.O.D)

**Szerzők:** Böszörményi Zoltán (Zboszor), Bicsák Lajos (Asimo), Gondos Csaba (Csaba), Mezei Róbert (m/zx), Pgyuri, Samu József (Sam. Joe)

**Szerkesztőség e-mail címe:** speccyalista.vilag.szerkesztoseg@sinclair.hu

**Kiadó:** Speccyalista Baráti Kör http://sinclair.hu

**Facebook:** https://www.facebook.com/Speccyalista-Világ-1860907330696250/

2016. április

# ARCKÉPCSARNOK

## Interjú Velvárt Andrással

A Sinclair.hu mindig is fontos feladatának érezte, hogy a felkutassa a hazai „Spectrumos éra” jeles alakjait és emléket állítson akkori tevékenységüknek. Különösen izgalmas, hogy ezúttal talán egy akkoriban népszerű Spectrum illesztő, a Micro-POKEer fejlesztésébe is betekintést nyerhetünk, melyről pont az előző számunkban olvashattatok méltatást Asimo billentyűiből.
Ráadásul a sors kezét is vélhetjük a mögött, ahogy Andrásra ráakadtam egy Facebook kommentjének köszönhetően, melyben épp megemlékezett a gyerekkori munkájáról.

**Sinclair.hu: Először is szeretném megköszönni, hogy ilyen lelkesen állsz ezen email interjúhoz. Rögtön belevágnék a közepébe és rákérdeznék a rejtélyre, amely a fejlesztőcsapat nevét övezte ezidáig. Ki is az a V&REW?**
**Velvárt András:** A V&REW az én „művésznevem” volt, a Velvárt Andrew (=András)-ból összerakva. Tudom, gyenge poén, de még csak 13 éves voltam.

**Sinclair.hu: Mióta Spectrumoztál már ekkor? Hogyan tanultad meg a gépi kódot?**
**Velvárt András:** Kicsit összemosódnak az évek, de ha jól rémlik, olyan 84-85 körül kezdtem a gépezést. Édesanyámék elváltak, sokat kellett dolgoznia - és egy másodállásába magával vitt, ahol egy rakat HT 1080Z számítógép volt, Junoszty monitorokkal. Nem nagyon tudtam velük mit kezdeni, de kaptam 3 BASIC útmutatót, és azokat kívülről megtanultam, hogy legközelebb ne unatkozzak. Aztán sikerült közeli ismeretségbe kerülnöm egy C64-el, és megírtam életem első programját – azt hiszem, amolyan „gondoltam egy számot” dolog volt. Akkor szembesültem azzal, hogy papíron programozni nem ugyanaz… de lenyűgözött a gép következetessége. A programban maradt azonban egy bug, amit csak a kinyomtatott forráskódból vettem észre – a cím körül az egyik oldalon több **** volt, mint a másikon. Édesapám ekkortájt hozott be „Nyugatról” egy Spectrumot, amivel rengeteget játszottam, amikor nála voltam. Végül az elmaradt gyerektartások fejében át is került hozzánk. Ez olyan 85-86 körül lehetett. Elkezdtem BASIC-ben programozni, klubokba járni, de féltem az assemblytől, fekete mágia volt számomra… végül egy kinyomtatott, magyarázatokkal tűzdelt Spectrum ROM könyv segített át a nehézségeken, illetve egy Csillag Péter vezette nyári tanfolyam.

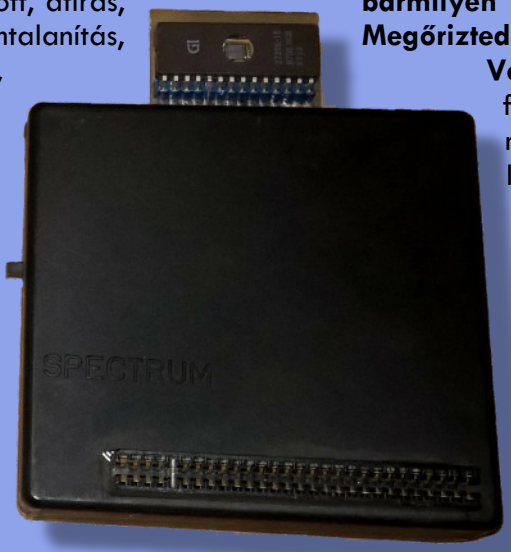
**Sinclair.hu: Hogyan kerültél kapcsolatba a Micro-Stúdióval? Miért, hogyan válasz-tottak?**
**Velvárt András:** Hát, pontosan nem emlékszem, de úgy rémlik, hogy valamelyik programozói verseny után jött oda Koszó István, hogy dolgoznék-e nekik. Soha nem dolgoztam még addig pénzért, de végül Anyu unszolására belementem.

**Sinclair.hu: A teljes ROM-ot egyedül készítetted?**
**Velvárt András:** Igen. Ahhoz képest egész jól sikerült, ha még így 30 évvel később is eszébe jut valakinek.

**Sinclair.hu: Egyedi ötlet volt vagy a Multiface adta az inspirációt?**
**Velvárt András:** Úgy rémlik, hogy a cél kifejezetten a Multiface funkcionalitásának koppintása volt. Az akkori, vasfüggönyös időkben elég nagy kereslet volt az ilyesmire.

**Sinclair.hu: Milyen eszközöket használtál a fejlesztéshez, teszteléshez?**
**Velvárt András:** Komolyabb assembert, Laser Genius volt a neve!? Percekig töltött csak az assembler, de úgy voltam vele, hogy megéri, mert sokat tudott. Volt egy sokkal gyorsabb assembler is, azt is használtam néha kis változtatásokra. Meg néha a disassemblert, ha csak valami picit kellett módosítani – akkortájt rengeteget használtam a gépi kód táblázatot.
A kész kódot aztán EPROM-ba égettem. Az EPROM égetőt a fejlesztéshez kaptam, valamint kaptam egy egyedi Micro-POKEer hardvert is, amiben könnyen kicserélhető volt az EPROM. Kb. 5-6 EPROM volt folyamatosan használva. Aki nem ismeri, az EPROM égetés akkortájt úgy működött, hogy az EPROM-ot először UV fénnyel ki kellett törölni (minden bit 1-be állt). Ezután jött az égetés, ami a megfelelő biteket 0-ba billentette. És ekkor tudtam csak kipróbálni, hogy mit csináltam. Mivel csak egy Spectrumom volt,

ez azt jelenti, hogy ha a forrásban egy betűt át kellett írni (pl JZ helyett JNZ), akkor először 2 perc alatt bejött az assembler, utána még egy perc, amíg a program forrása is megjött, átírás, fordítás, mentés, áramtalanítás, EPROM égető bedugás, kiírandó bináris betöltése kazettáról, égetés (közben előző EPROM-ot kivinni a fürdőszobába „napozni"), áramtalanítás, Micro-POKEer bedugása, EPROM behelyezés, bekapcsolás – és máris lehetett tesztelni. Ez így kb. 15 perces folyamat volt, nem számítva azt, amikor 10 esetből egyszer hibás lett az égetés. Ja, és debuggerről álmodni sem lehetett – ha valami nem ment, akkor a gép egyszerűen elszállt vagy lefagyott, lehetett találgatni, hogy mitől.




Később javult a helyzet – először lett egy 80K-s Spectrumom, amivel jól tudtam szimulálni a ROM váltást, és így az egész EPROM égetésesdi sokkal ritkább lett. Aztán beruháztam egy floppy-s egységbe, ami egészen elképesztő javulást hozott, a 2-3 perces programbetöltést 5 másoperc körülire vitte. Úgy éreztem, lobog a hajam a produktivitástól, hihetetlen volt.

**Sinclair.hu: Párhuzamosan folyt a szoftver és hardver fejlesztése vagy specifikáció alapján dolgoztál?**

**Velvárt András:** Én már kb. kész hardvert kaptam, de tudom, hogy kisebb-nagyobb változtatásokat folyamatosan csináltak rajta.


**Sinclair.hu: Talán ennyi idő távlatából nem indiszkrét a kérdés, hogy anyagilag is megérte neked ez a projekt?**

**Velvárt András:** Hetedikes – nyolcadikos voltam, egyáltalán nem az anyagiak voltak a fontosak. Az biztos, hogy ettől nem javult lényegesen az anyagi helyzetünk, de talán Anyunak nem kellett minden másodállásra igent mondania, és jutott az 1 forintos fagylaltgombócból is.

**Sinclair.hu: Voltak még más megbízások is?**

**Velvárt András:** Egyszer hoztak egy egeret Kempston joystickra kötve, és megkértek, hogy találjam ki, hogy hogyan lehetne vele rajzprogramokat irányítani (talán a Leonardo volt akkor a csúcs?) Életemben először akkor használtam egeret, amikor sikerült hozzá megírni a szoftvert. Már akkor is a jövőben éltem.

**Sinclair.hu: Maradt fent az utókornak a fejlesztésről bármilyen írásos emlék, pl. assembly lista? Megőrizted a kazettáidat?**

**Velvárt András:** Van egy pár kazettám és floppy-m, de nem tudom, hogy mik vannak rajta. Ha tudnál segíteni megnézni, akkor lehet, hogy kincsre bukkanunk.

**Sinclair.hu: A V&REW feliratban a V-re miért került OVER-rel egy függőleges vonal? Olyan, mint ha W&REW akarna lenni.**

**Velvárt András:** Látod, erre nem is emlékszem. Ha ott van, akkor azért, mert meg tudtam csinálni. Úgy lehettem vele, mint a hegymászók a heggyel.

**Sinclair.hu: Úgy tűnik, a 16K-s Spectrummal nem volt teljesen kompatibilis (pl. beégetett 48K adatmentés).**

**Velvárt András:** Wow, fogalmam sem volt. Remélem már lejárt a garancia, mert nem hiszem, hogy ki tudnám javítani.

**Sinclair.hu: A TURBO mentés /betöltés rutinja hogyan készült?**

**Velvárt András:** Valahol olvastam, hogy a turbó betöltés abban különbözik a rendestől, hogy pár érték más. Fogtam a gyári ROM-ból a betöltés és kimentés kódját, átmásoltam, aztán addig kísérleteztem a számokkal, amíg gyorsabb nem lett. Ha jól rémlik, simán csak elfeleztem az eredeti értékeket, de tény, hogy ehhez már jó minőségű magnó kellett. És így elértük a szédítő 2400 baudot (ami a gyári duplája)! Ez azt jelenti, hogy másodpercenként 300 bájtot is át tudott vinni, ami ugye percenként 18K, és így a 48K-s Speccy RAM-ját 3 perc alatt komplett betöltötte. Nagy szó volt ez a szokásos 5-6 perces betöltésekhez képest.

**Sinclair.hu: A TURBO snapshot mentés esetén nem kerül mentésre TURBO betöltő, így nem lehet visszatölteni, csak a MICRO-POKEerrel. Ez egy apró hiányosság, vagy szándékos? Esetleg későbbi verziókban javítottad?**

**Velvárt András:** Nem rémlik, hogy valamikor szempont lett volna… de hát az se rémlett, hogy ilyen gondok voltak. Rég volt, na. Azt is el tudom képzelni, hogy szándékos döntés volt, hogy a másolt programok betöltéséhez meg kelljen venni a hardvert. Mai szemmel nézve fura egy idők voltak azok.

**Sinclair.hu: A TURBO opció nem volt megbízhatóan használható az átlagos magnókkal. Ezt a problémát érzékelték, jelezték a vásárlók, esetleg javítottátok is egy későbbi verzióban?**
**Velvárt András:** Emlékeim szerint a turbó az első verziókban nem volt benne, de miután belekerült, nem nyúltam hozzá. De ugye lehetett választani turbó és sima mentés között, akinek nem ment a TURBO, az maradhatott a simánál.

**Sinclair.hu: Hány ROM verzió létezik? Asimo barátunk az 1.6-ot fejtette vissza. Mi volt a legelső és mi a legutolsó? Esetleg hozzáférhető még a legutolsó verzió?**
**Velvárt András:** Van egy EPROM-om, de nem tudom, van-e rajta bármi értékelhető. Sajnos a Speccy-met még nem sikerült életre bírni, de megnézhetjük közösen.

**Sinclair.hu: A Micro-Monitor felülvágta a képmemóriát. Esetleg RAM-mal bővítve megoldható lett volna, hogy pontos snapshot mentés készülhessen. Voltak ilyen tervek?**
**Velvárt András:** Ha voltak is, én nem tudok róla. Fontos volt az olcsó hardver, márpedig a RAM jelentősen megdrágította volna. Mivel a hardverben nem volt RAM, ezért csak a gép RAM-jába dolgozhattam – a szükséges változókat így a képernyő memóriába voltam kénytelen tenni. Ezért látszik némi „szemetelés", miután aktiválod a kütyüt.

**Sinclair.hu: Miért volt lehetőség a Micro-POKEer ROM szoftveres aktiválására (port 7Fh)? Elméletben elég lett volna csak a Spectrum ROM-ba visszalépéshez használni. Ezzel így csak lefagyott a Spectrum. Bug vagy feature? Esetleg voltak tervek a későbbi hasznosítására?**
**Velvárt András:** Miket nem találtatok meg! Szerintem így egyszerűbb volt a hardver – a kapott bitet beírta egy sima tárolóba. Lehet, hogy használtam arra, hogy az NMI kiváltása nélkül ellenőrizzem a beégetett ROM-ot, sajnos erre már nem emlékszem.

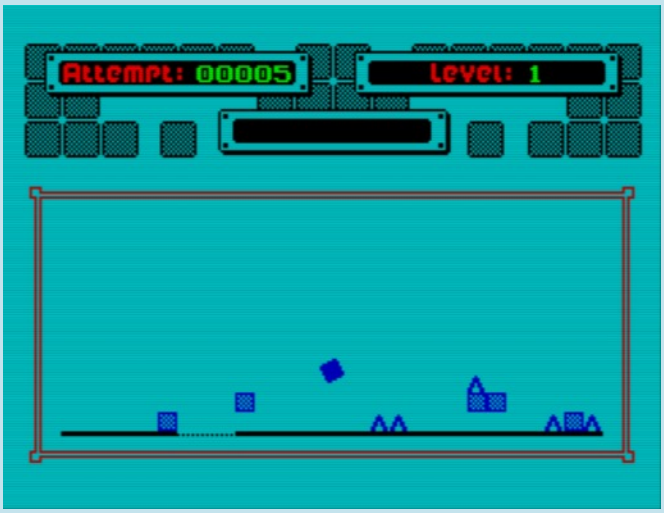
**Sinclair.hu: A szakmában maradtál? Mivel foglalkozol manapság?**
**Velvárt András:** Ma is olyan kocka vagyok, mint régen, csak közben kicsit kikerekedtem. Ha a kör négyszögesítése nem is megy, a kocka gömbölyítését sikerült megoldanom.
Microsoft technológiákkal dolgozom már vagy 17 éve. Tavaly és tavalyelőtt két társammal startupoltam egyet, mi csináltuk a 2 milliónál több letöltést megélt Guitar Hero szerű mobiljátékot, a SongArc-ot. Januárban publikálta a Pluralsight (az IT tudás Netflix-e) a Windows 10 programozásába bevezető online tanfolyamomat (http://bit.ly/win10xaml). Manapság elsősorban konzultálok kisebb-nagyobb projekteken – itthonra és külföldre is. 8. éve vagyok Microsoft MVP (Most Valuable Professional), ami egy évente újra és újra kiérdemelendő külsős szakértői díj. Most éppen Emerging Experiences kategóriában, ami olyan klassz dolgokkal foglalkozik, mint a beszéd alapú számítástechnika, HoloLens, inking, Kinect, gesztusok, stb. Szenvedélyem a jövő előrehozása a jelenbe, szóval úgy érzem, jó helyen vagyok.

**Sinclair.hu: Meg tudnád fogalmazni egy mondatban, hogy mit jelentettek neked a Spectrumos évek?**
**Velvárt András:** Egy mondatban nem. Annál sokkal többet jelent.
Az egész életemet meghatározta az az 5-6 év, teljesen más ember lennék nélküle. Szakmailag olyan alapot adott, amit egy mai fiatal már nem kap meg – egyszerűen jóval magasabb szintről indulnak. Manapság már nem olvasnak a takaró alatt OS-ek gépi kódját, nem ismerik egy gép teljes kapcsolási rajzát. A számítógép alakította ki a gondolkodásomat, és a Garádi „tans'úr" János vezette Óbudai Számítástechnikai Szakkör volt a hetem fénypontja, az a társaság, ahol igazán jól érezhettem magam (máig is vannak barátaim abból az időből). Nem beszélve a nyári táborokról Sóstón…

***Az interjút készítették:***
***Lajos Bicsák (Asimo) és Kardos Balázs (Balee)***

# JÁTÉKÚJDONSÁGOK

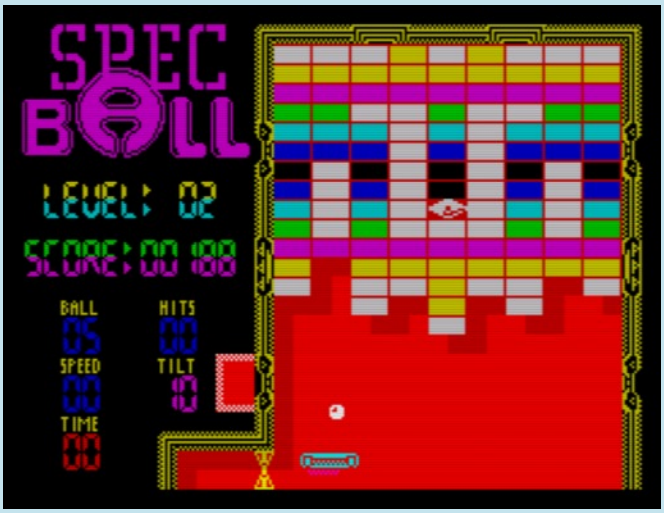
## Infeasible Game (Ian Munro)




Ian Munro játéka az egyik legkelendőbb Xbox indie játék, a The Impossible Game adaptációja. A játékmenet nagyon egyszerű, gyakorlatilag egyetlen gombot, az ugrást kell használni, plusz néha még egyet, a zászló lerakásához. A főszereplő, a kocka alatt állandó sebességgel mozog a talaj és különböző, egyszerű alakzatok kerülnek az útjába, melyek alatt át kell siklania, vagy felettük átugrania, vagy róluk elrugaszkodnia. A tüskéket és a vizet minden körülmények között kerüld el, a kockákra ráugorhatsz, de ne menj nekik. A zászló lerakása pedig arra jó, hogy ha meghalsz, márpedig meghalsz, akkor az utolsó zászlótól kell folytatnod a szintet és nem a legelejétől. A játékban AY zene fokozza a hangulatot.

## Specball (Zozosoft)

A Specball az 1988-ban Enterprise-ra megjelenő Enterball idén megjelent Spectrum változata. Az eredeti Enterprise sztori spectrumosított változata szerint 2213-at írunk. A hírhedt varázsló, a computer-mágus McKell Daughleny feldühödve azon, hogy a boltban az orra elől vitték el az utolsó Spectrumot, varázsjoystickja egyetlen suhintásával betölthetetlenné tette az összes Spectrum játékot. A rettegés szele söpört végig a világon. Mi lesz, ha az emberek nem játszhatnak soha többé szeretett programjaikkal? A varázsló végül megkönyörült a szenvedő emberiségen. Készített egy játékot, a Specballt, és elküldte a világ legügyesebb játékosainak. Ha egyiküknek sikerülne teljesíteni a játék összes pályáját, a programok megszabadulnának a varázslat alól, és az emberek újra játszhatnának kedvenc számítógépükön.



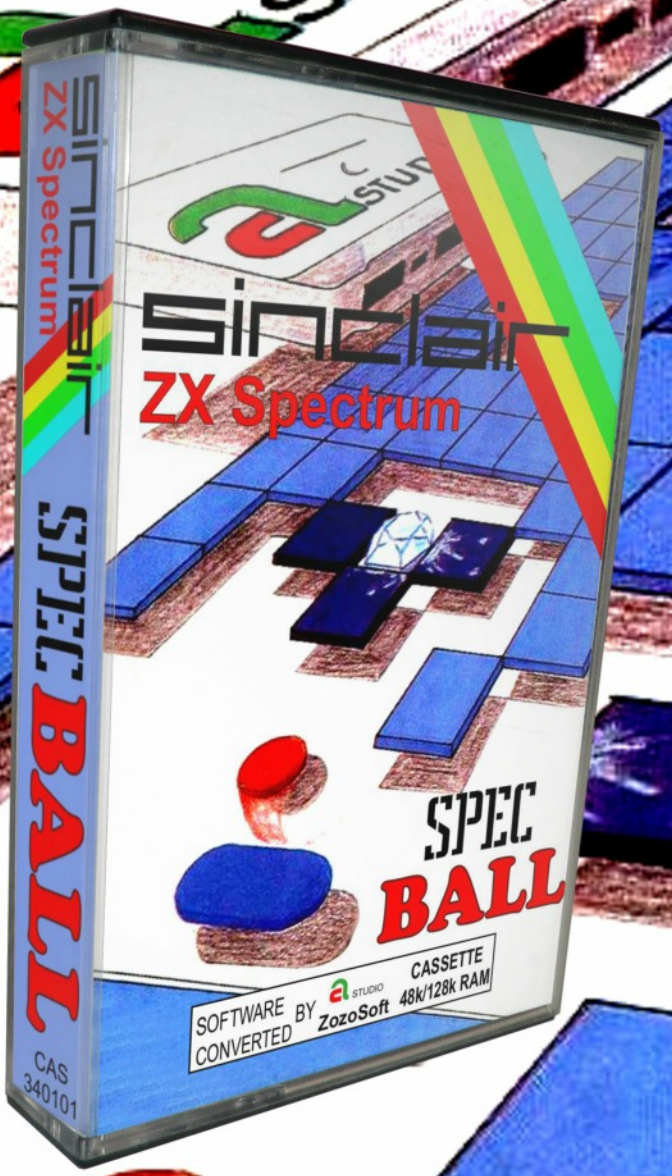

## Stars (Gumi) (kas29)


Kas29 hetedik (AGD-vel készült) játéka a Stars (Gumi). Gumi a neve az űrhajónak, amit vezérelni fogsz. Arra fogod használni, hogy minden szinten belökdösd a csillagokat a csillagcsapdába. Csillagok vannak mindenfelé a galaxisban, már nem működő űrállomásokon, a fekete lyukak közelében és még sok-sok helyen. Guminak 30 másodperce van egy-egy csillag csapdába juttatására, amiben megpróbálják akadályozni a 'Sharovert' nevű repülő képződmények, a lézerágyúk és a lézerek. De nem csak ellenségei vannak, hanem segítői is, a fagyasztó lefagyasztja az időt és a Sharoverteket, a kapcsoló lekapcsolja az ágyúkat és a lézert. Az élet egyértelműen egy plusz életet ad, a piramis pedig eljuttatja Gumit a következő szintre.

www.fanzix.hu

ÚJDONSÁG!

ÚJDONSÁG! TÖLTSD LE INGYEN! TÖLTSD LE INGYEN! TÖLTSD LE INGYEN!




STUDIO
ZozoSoft

# SPEC BALL

A hírhedt varázsló, McKell Daughleny varázsjoystickja egyetlen suhintásával betölthetetlenné tette az összes Spectrum programot. Azonban készített egy játékot, a SpecBallt. Ha valakinek sikerül teljesíteni a játék összes pályáját, a programok megszabadulnak a varázslat alól. Legyél TE a megmentőnk, törd meg a varázslatot, teljesítsd mind az 50 szintet!

# KLÓNOK HÁBORÚJA

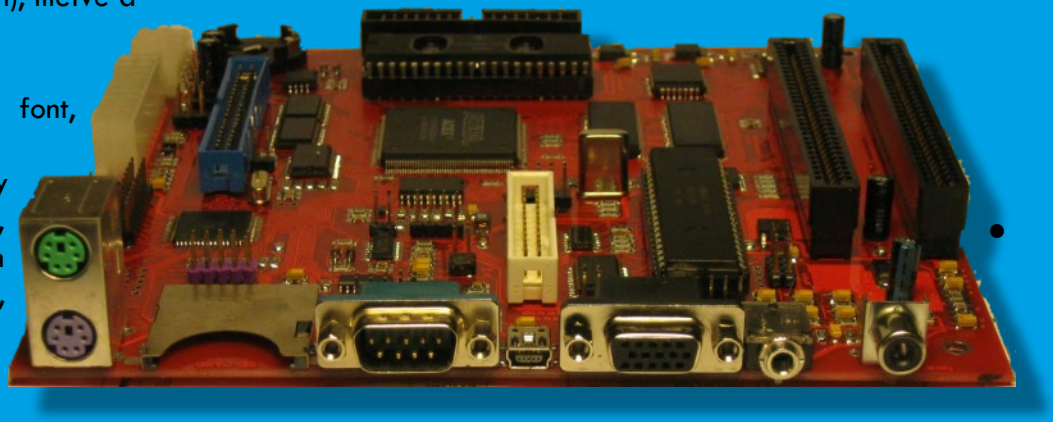
## AMIT SOSEM MERTÉL MEGKÉRDEZNI A ZX EVOLUTIONRÓL 1. RÉSZ

### Bevezetés

A ZX Evolution az orosz NedoPC csoport konstrukciója, egy modern hardver megoldásokból újraépített ZX Spectrum és Pentagon klón alaplap. A hardver fejlesztése 2009-ben kezdődött, eddig három revíziót ért meg és ma is tart. Az A revízióból csak 5 példány készült, ezek voltak az első mintadarabok. A B és C revíziók már nagyobb mennyiségben készültek, és megvásárolhatók voltak a tervezőktől, illetve megvásárolhatók jelenleg is (2016. január) közvetlenül http://tetroid.nedopc.com címről (a tetroid@inbox.ru emailcímen), illetve a http://www.sellmyretro.com weboldalról.

Az alaplap ára 95 angol font, illetve 6500 orosz rubel.

Létezik egy Unreal Speccy nevű emulátor program, amely orosz ZX Spectrum klónokat képes emulálni, ennek fejlesztésében részt vesz a NedoPC csoport és a TS-LABS csoport is. Ez a szoftver hajtja a ZX Evolution-t is, ATM3 néven. *A weboldalon fellelhető képernyőképek az emulátor TS-LABS által készült változatával készültek, illetve a legutolsó fejlesztői ROM-mal, melynek verziója 0.56f.*



### Áttekintés

A ZX Evolution az egyik legnépszerűbb orosz ZX Spectrum klón, a Pentagon egy modern újragondolása, illetve továbbfejlesztése. A NedoPC csoport több klónt is kifejlesztett, eredetileg ATM Turbo 1, ATM Turbo 2 és ATM Turbo 2+ néven, de az elvileg soron következő ATM Turbo 3 kódnév helyett ez az új alaplap a ZX Evolution nevet kapta. Több változatot, illetve újratervezést is megért, az utolsó a C revíziós ZX Evolution alaplap, amelynek paraméterei illetve komponensei a következők:

- QFP tokozású, felületforrasztott Z80 CPU (Z84C0020FEC típusjelzéssel), amely maximálisan 20MHz-en képes futni. Ezen az alaplapon a ZX Spectrumhoz közeli, klasszikus 3,5MHz-en, illetve turbó módokban 7 és 14MHz-en működhet.
- A perifériák egy részét (pl. videó kimenet, IDE) egy Altera EP1K50 FPGA valósítja meg.
- Az Altera FPGA-val együttműködik egy ATMega128 AVR mikrokontroller is.
- WDC1793 floppy kontroller.
- Yamaha YM2149F hangvezérlő, a ZX Spectrum 128K-ban megtalálható Yamaha AY-3-8910 egy variációja. Az YM2149F volt megtalálható annak idején például az Atari ST-ben is.
- miniATX alaplap-formátum, melynek révén bármilyen MiniPC házba beépíthető a ZX Evolution.
- 512KB EEPROM
- 4MB RAM

Az alaplap hátlapi csatlakozói a következők:

- PS/2 billentyűzet, egér
- SD kártya foglalat
- VGA D/SUB csatlakozó
- miniUSB aljzat
- RS-232 soros csatlakozó
- sztereo 3,5mm-es jack, hang kimenet
- mono RCA, hang bemenet

Az alaplapi csatlakozók:

- ATX tápcsatlakozó ATX táphoz
- AT merevlemez tápcsatlakozó AT táphoz
- 20-tűs IDC csatlakozó az eredeti ZX Spectrum billentyűzetmátrix és Kempston joystick kivezetéshez
- 34-tűs floppy csatlakozó
- 40-tűs IDE csatlakozó
- 26-tűs csatlakozó párhuzamos port kivezetéshez Centronics nyomtatókhoz
- 3 darab 4-tűs CD-audio kábel csatlakozó (hangkártyákhoz vagy CD-hez)
- RGB videó panel csatlakozója

- két ZXBUS illesztőhely
- FPGA illetve AVR programozáshoz csatlakozók

A ZX Evolution egy csupasz miniATX alaplap. A működőképes számítógép összerakásához szükségünk lesz a továbbiakra:

- egy (fentebb már említett) MiniPC számítógépház
- PS/2 billentyűzet
- igazi PS/2 egér (Sajnos manapság a PS/2-USB átalakítóval működő USB egerek ritkák.)
- VGA D-SUB csatlakozós monitor, amely képes 15 vagy 31kHz-es sorfrekvenciát kezelni. Lásd a monitor-kompatibilitási listát a NedoPC weboldaláról. Retró számítógépekhez a régi CRT RGB monitorok helyett ajánlott pl. a Benq BL912 monitora, amely az 5:4-es képarány-nyal sokkal korhűbb képet ad, mint egy 16:9-es képarányú monitor. Sokszor egy 16:9 képarányú monitor nem is képes 4:3-as képet megjeleníteni.
- SD / SDHC memóriakártya
- opcionálisan floppy drive, merevlemez, vagy CD/DVD drive, utóbbiak IDE felületűek. SATA meghajtókat IDE2SATA adapter kártyával képes kezelni a rendszer

## BaseConf, Evo Reset Service

Amennyiben egy modern MiniPC számítógépházba építettük az alaplapot, az ATX tápegység Power vezetékét az alaplap soft reset csatlakozójához, a Reset vezetéket az alaplap hard reset csatlakozójához célszerű bekötni. Ehhez az alaplappal együtt érkező nyomtatott kézikönyv, illetve a NedoPC oldalról letölthető PDF nyújt segítséget. Ekkor a gépház bekapcsoló gombja, illetve annak 4 másodperces nyomva tartása a számítógép be- illetve ki-kapcsolását eredményezi. Működés közben a bekap-csoló gomb ún. „soft reset”-et, a reset gomb pedig „hard reset”-et eredményez. A hard reset ugyanazt eredményezi, mint a bekapcsolás folyamata, az AVR mikrokontroller resetel, és a saját flash EEPROM-jából újratölti az FPGA IC-be a programozó szekvenciát. Soft reset esetén csak a Z80 CPU indul újra.
Az alaplapot a NedoPC csoport az általuk kifejlesztett *BaseConf* konfigurációval szállítja. Bekapcsolás után ennek a bejelentkező képét, illetve főmenüjét látjuk. Ha nem látjuk, illetve fekete a képernyő, akkor ne essünk pánikba, hanem nyomjuk meg egyszer a Scroll Lock gombot a PS/2 billentyűzeten. Ha csatlakoztatva van a monitor és be is van kapcsolva, ezután jó esetben a ZX Evolution főmenüjét láthatjuk. Ez az Evo Reset Service, röviden ERS.

A NedoPC által megálmodott és megvalósított ZX Spectrum klón nagyon átfogó képet mutat, és sok olyan lehetőséget, amelyről a nyugati ZX Spectrum tulajdonosok nem is álmodtak. A képernyőn a billentyűzettel (a menüpontok betűjelével, illetve a fel/le gombokkal), illetve egérrel is navigálhatunk. A *H* gombbal egy súgót kaphatunk ezen gombokról,


illetve a *reset+billentyű* kombinációk esetén elérhető funkciókról.
A képernyő felső részén az Evo Reset Service saját verziószámát és további információkat láthatunk, például a firmware verziószámát. A képen a *BaseConf* és az *AVR Boot* feliratok mellett az *UnrealSpeccy 30.01.2016 beta* olvasható, ez valódi hardveren a firmware verziótól függ.
Például:

```
BaseConf: ZXEvo 4M 08.11.2013
AVR Boot: ZXEvoAVRBoot 23.01.2012
```

vagy TSconf firmware esetén (erről egy következő részben lesz majd szó):

```
BaseConf: ZXEvoTS&Base 05.04.2015
beta
AVR Boot: ZXEvoAVRBoot 23.01.2012
```

A jobb oldalon a dátumot és az időt láthatjuk. Igen, az alaplap tartalmaz egy valós idejű órát is, amelyet CR2032 gombelem lát el energiával kikapcsolt állapotban is. Az órát a CMOS tartalmának szerkesztésével állíthatjuk be, de erről később.
A képernyő közepe két részre van bontva. A bal oldalon a jelenlegi konfigurációról szóló információ található, amelyet a sor elejére kiírt számokkal illetve betűkkel megváltoztathatunk. A jobb oldalon a kék háttéren a főmenü (*Main menu*) elemei találhatók.

## NMI menü, monitor

Az alaplapon a CPU NMI (nonmaskable interrupt) vonala is kivezetésre került, ehhez egy nyomógombot is csatlakoztathatunk. Az NMI (vagy *Print Screen*) gombot megnyomva egy menübe jutunk, amely egy minimalista monitor programot tartalmaz. Jelenleg ez annyit tesz, hogy megtekinthetjük a CPU regisztereinek tartalmát, menüből megváltoztathatjuk a CPU órajelét, a memóriazárolási beállítást, illetve visszatérhetünk a futó programhoz. Ezt az almenüt láthatjuk lentebb.


Ha nincs extra nyomógomb a számítógépházon, és nem akarjuk magunk fúrni-faragni a gomb helyét, akkor használhatjuk a PS/2 billentyűzet *Print Screen* (eredetileg *SysRq*/*Print Screen*) feliratú gombját is.
Vannak, voltak a Pentagon gépekhez több lemezes játékok, amelyek feltételezték, hogy csak egy floppy van a géphez csatlakoztatva, és menet közben kérték az új lemezt. Sajnos, erre a ZX Evo-val nincs lehetőség. Évek óta ígérik, hogy az NMI menüből elérhető lesz az SD-kártya böngésző is, de ennek jelenleg semmi nyoma, ahogy POKE-okat sem tudunk ezen keresztül bevinni.

## Fájlok kezelése

A ZX Evolution tartalmaz egy teljes méretű SD-kártya foglalatot, amellyel SD (max. 2 GB) illetve SDHC (max. 32 GB) méretű memóriakártyákat képes kezelni. Az SD(HC) kártyát FAT32 fájlrendszerrel kell formázni, exFAT fájlrendszert nem kezel az Evo. Az IDE csatlakozót is használhatjuk merevlemez, illetve CD/DVD-ROM meghajtók csatlakoztatására. Sajnos, az IDE felületű eszközök kora lejárt, de léteznek IDE2SATA adapterek, amellyel SATA eszközeinkből kettőt (master, illetve slave meghajtóként) csatlakoztathatunk olyan, IDE eszközöket kezelő alaplapokhoz, mind amilyen a ZX Evolution is.

Az ERS főmenüjének második menüpontja a *File browse*, azaz fájlböngésző, itt az SD-kártya, illetve opcionálisan a merevlemez tartalmát böngészhetjük. Ezt a funkciót elindítva, bootolás után logikailag az első, illetve az utoljára kiválasztott lemez és könyvtár tartalma jelenik meg. A fájlböngésző ablakban a következő lehetőségeink vannak:

- kurzor mozgató billentyűk: fel/le egy fájlt lépünk, balra/jobbra egy lapnyit lépünk
- *D* billentyű: ha több lemezmeghajtónk is van (pl. SD-kártya és egy merevlemez), akkor ezzel választhatjuk ki a meghajtót
- *Enter*: fájl kiválasztása vagy futtatása
- *V* billentyű: Fájl megtekintése
- *X* billentyű, *EDIT* (*CapsShift+1*), *BREAK* (*Caps Shift+Space*): Kilépés a fájlböngészőből

Az ERS képes többféle fájltípust is felismerni. Ezek: TRD, SCL, FDI, $C, SPG, TAP, BMP.
A TRD, SCL és FDI fájlokról bővebben a következő fejezetben is szó esik. Ezek lemezkép fájlok, floppy lemezekről készült digitális másolatok. TRD kiválasztása esetén a rendszer egy új menüt kínál fel, amellyel a tartalmát RAM-diszkre (memóriába) másolhatjuk, illetve az A-D lemezcímkék alá felcsatolhatjuk. Az SCL és FDI tartalmát csak RAM-diszkre másolhatjuk. A megtekintés funkció a lemez tartalmát, a fájlok listáját mutatja.
A TAP fájlokról is esik szó később, ezek magnószalagok digitális másolatai. Ezeket, a magnóhoz hasonlóan, futtatásra betölthetjük.
A $C kiterjesztésű fájlok úgynevezett Hobeta fájlok. Ezt a fájlformátumot eredetileg az EmuZWin nevű ZX Spectrum emulátorhoz fejlesztették ki, és egy teljes TR-DOS (TRD) lemezkép helyett egyetlen fájlt tartalmaznak. $B (Basic), $C (bináris kód), $D (adattömb) fájlformátumok léteztek. A $C fájlformátumot általános célokra, azaz különböző, ZX Spectrumhoz kifejlesztett Commanderek és DOS-ok is elkezdték használni. A Hobeta fájlok elején egy 17 bájtos fejléc található, amely tartalmazza a főbb paramétereket: mi a fájl neve, típusa és hová töltődjön a memóriában.
Az SPG fájlok egy teljes memóriaállapotot (snapshot) tartalmaznak, így ezek gyakorlatilag indítható programfájlok.
A BMP fájlok Windows alól mint képfájlok ismerősek. BMP formátumú, mximálisan 320x200 felbontású és

maximum 16 színű képeket képeket tud az ERS megjeleníteni.

## Lemezek, lemezkép-fájlok kezelése, RAM-diszk

Az orosz (és több más, kelet-európai) klón az angol Technology Research Betadisk 128 floppy vezérlőjét vette alapul, ami négy DS/DD floppy drive-ot volt képes volt kezelni. Bár az alaplapon ennek a vezérlő IC-je is szerepel (WDC1793), az alaplappal együtt szállított kábel 1 drive kezelésére képes. Az 1-4 gombok segítségével a valódi drive címkéjét állíthatjuk be, de valójában nincs szükség valódi floppy meghajtó használatára.

Az SD kártyára másolt TRD (azaz TR-DOS) image-ek, lemezképek segítségével emulálja a rendszer a floppy meghajtókat a TD-DOS számára. A képernyő bal-alsó sarkában látható *Mount A: … Mount D:* információs sorokban látható, hogy melyik lemezcímkére melyik TRD fájlt csatoltuk fel. Egy TRD lemezkép felcsatolását a főmenü *File browse* menüpontjával kezdeményezhetjük. A fájlböngészőben a navigálás a fel/le gombokkal fájlonként történik, illetve a balra/jobbra gombokkal egy lapnyit ugorhatunk. Az Enter gombbal választjuk ki a fájlt, amikor is megnyílik egy almenü, amely megkérdezi, hogy melyik lemezcímkére kívánjuk csatolni a lemezképet. Az így felcsatolt lemezt a főmenü „TR-DOS boot" opciójával indíthatjuk el. A csatolást a „Service > Dismount image" menüpontjával szüntethetjük meg, amely szintén megkérdezi, hogy melyik lemezcímkénél hajtsuk végre a lecsatolást.

Az alaplap 4MB RAM-ja elegendő arra is, hogy egy teljes TR-DOS lemezképet memóriában tároljunk RAM-diszken. Ezt a RAM-diszket használja a rendszer arra is, hogy SCL illetve FDI (tömörített) formátumú TR-DOS lemezképet kicsomagoljon, ezután annak tartalma RAM-ból futtatható. A RAM-diszk lemezcímkéjét, már amennyiben használatban van, az *Y* gombbal változtathatjuk meg.
A RAM-diszk, jellegéből adódóan, nem maradandó, kikapcsolás vagy újraindítás után a tartalma elveszik. Ezzel szemben a teljes méretű TRD lemezképek tartalma az SD kártyán megmarad, és a rendszer írja-olvassa ennek tartalmát, tehát az elmentett munkánk, vagy egy játék high score listája nem illan el. Figyeljünk arra, hogy a TRD lemezképekből kétféle létezik: teljes méretű, illetve rövidített méretű. A rövidített (csonkolt, „truncated") TRD fájl csak a lemez elejéről származó, használatban lévő blokkokat tartalmazza. Ez utóbbi sem alkalmas arra, hogy munkát mentsünk rá.

## EVO DOS

A NedoPC csoport – a TR-DOS 5.03 kódjából kiindulva – kifejlesztett egy azzal kompatibilis DOS rendszert is, amely kezeli az ERS alól felcsatolt lemezképeket, és a RAM-diszk tartalmát is. Ezzel pl. valódi floppy meghajtók és lemezképek között másolhatunk fájlokat. Ezt a főmenü *TR-DOS* menüpontjából érhetjük el, vagy az S gombbal, illetve amikor 48K BASIC-ből a `RANDOMIZE USR 15616` parancsot hajtjuk végre.
A teljes Betadisk 128 leírást elérhetjük a http://www.worldofspectrum.org oldalról.

Azért a kompatibilitás nem teljes, két funkció eltávolításra került:

- a Betadisk „magic button" funkciója, ami egy RAM pillanatképet készített, és később a GOTO „programnév" CODE paranccsal lehetett betölteni
- az egyoldalas („SS") illetve 40 track-es („SD") floppy meghajtók kezelése.

Ennek fejében viszont kaptunk jó pár új parancsot:

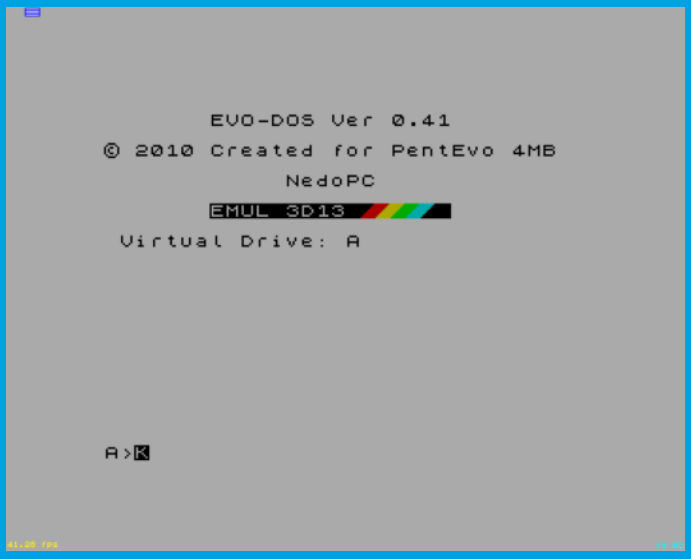
- .HELP – Kiírja a képernyőre az új parancsokat.
- .VER – Kiírja a képernyőre az EVO DOS aktuális verzióját és a build idejét.
- .VIRT [X:] - Kiírja az aktuális RAM-diszk lemezcímkét. Ugyanez a parancs, az opcionális lemezcímke megadásával, meg is tudja változtatni azt, és a CMOS-ba azonnal bekerül a változás, tehát ezután indított programok már az új lemezcímke alatt találják a RAM-diszk tartalmát, illetve reset után is az új beállítás lesz érvényben.

Amellett, hogy a TR-DOS szerinti *"X:" konvenciót is lehet használni alapértelmezett lemezcímke váltásra, az új *.X:* lemezcímke-megadást is használhatjuk. A „:" (kettőspont) jelenléte kötelező. Az X helyett írhatjuk a TR-DOS-ban megszokott A, B, C és D lemezcímkéket, de az EVO DOS képes a FAT32-es lemezeket is virtuális lemezekként mutatni, ezek az E, F, stb. lemezcímkéket kaphatják. A virtuális lemezekkel kapcsolatot további parancsok a következők:

- .DRIVE – Kiírja a jelenleg ismert FAT32 lemezeket.
- .DIR [/L] – A jelenlegi könyvtár tartalmát listázza. Alapesetben a 8.3-as rövid DOS fájlneveket, /L esetén a FAT32-es hosszú fájlneveket.
- .CD – Könyvtárváltás. Egyszerre egy szintet léphetünk, vagy egy alkönyvtárba vagy a .. használatával a szülő könyvtárba.

- .MOUNT fájlnév.TRD X: – Egy TRD fájlt csatol fel adott lemezcímke alá TR-DOS alatti használathoz.
- .UMOUNT X: – Egy lemezcímkéről leválasztja a csatolást.
- .CLRCMOS – A CMOS tartalmát alaphelyzetbe állítja.

Az EVO DOS képernyője:




## Hangbemenet, TAP fájlok

A mono RCA hangbemenettel a ZX Spectrumról ismerős szalagos magnóról is tölthetünk be játékokat, programokat.
Ha nem akarunk a magnóval bajlódni, az SD kártyára másolt TAP formátumú „szalagképeket" (angolul „tape image") fájlokat is használhatjuk a TRD fájloknál már említett *File browse* menüponttal, vagy a *Tape load* menüponttal.
Ezzel kapcsolatos két beállítás is használható. A főmenü bal oldalán az *L. Emu tape load*, és a későbbiekben olvasható *Setup* menü alatti *T. Autostart tape* opciók. A két beállítás kombinációjával elérhető üzemmódok a következő:

1. *Emu tape load: off* - ekkor a *Tape load* menüpont, 48K módban a *LOAD ""* parancs, illetve 128K módban a *Tape loader* az RCA hang bemeneten várja a betöltést. A *File browse* menüpontban egy TAP fájl kiválasztása az *Emu tape load* beállítást impliciten *on*-ra állítja.
2. *Emu tape load: on* és *Autostart tape: off* - ha a *File browse* vagy a *Tape load* menüpontok alól egy TAP fájlt választunk ki, akkor azt tudjuk majd betölteni 48K módban a *LOAD ""* paranccsal, vagy 128K módban a *Tape loader* menüponttal.
3. *Emu tape load: on* és *Autostart tape: off* - ha a *File browse* vagy a *Tape load* menüpontok alól egy TAP fájlt választunk ki, akkor az azonnal betöltődik.

## Memóriazárolás

Bizonyos régebbi 48K-s játékok már a 128K-s ZX Spectrumokon sem működtek rendesen, mivel a 128K-s gép I/O portokat használ memória lapozáshoz, ezért a 128K-s gépet is vissza kellett kapcsolni a 48K-s gépekkel kompatibilis módba. Ez itt sincs másképpen. A háromféle memóriazárolási üzemmód között az *M* gombbal válthatunk. A három üzemmód: *48K*, *128K* és *off*, utóbbi az, amikor a ZX Evo 4MB RAM-jából bármelyik 16KB-os szelet belapozható bárhová. A ZX Evolution a Pentagontól megörökölt memórialapozást bővítette ki maximálisan 4MB-ig (256 * 16KB). A ZX Spectrum +3 -hoz hasonlóan a ZX Evolution is képes a legalsó 16kB ROM helyére RAM-ot belapozni, ezzel pl. CP/M-et futtatni.

## Turbó módok

Az ERS képernyőjén *W* gombbal változtathatjuk a CPU órajelét, a standard 3,5 MHz, 7 MHz-es turbó mód, illetve 14 MHz-es turbó mód között.

## Perfect Commander

A főmenü *Perfect Cmd* menüpontjából (*X* gomb) egy TR-DOS lemezképeket kezelő fájlmenedzsert érhetünk el. Ehhez fel kell csatolni legalább az A: lemezt, különben a Perfect Commander nem működik megfelelően valódi hardveren. (Az Unreal Speccy emulátor alatt érdekes módon elindul, és üres fájl listát ad, ha nem csatoltunk fel lemezt.) A Perfect Commander felülete az MS-DOS őskorából származó *X-Tree Gold* programéhoz hasonlít, egyszerre csak egy lemez tartalmát mutatja.
A fájlmenedzser alaplehetőségei:

- *1* … *4* gombok: váltás A:, B:, C: és D: lemezek között
- *C* gomb: másolás
- *D* gomb: fájl törlés
- Q gomb: kilépés TR-DOS-ba

## 48K BASIC

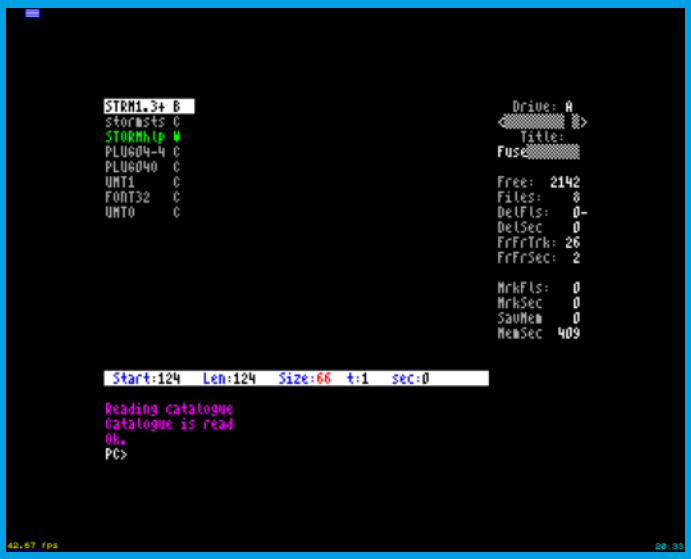
A jól ismert *(C) 1982 Sinclair Research Ltd.* feliratot az *I* gombbal, vagy a 48K BASIC menüpontból érhetjük el. A Betadisk 128 bővítést a `RANDOMIZE USR 15616` paranccsal aktiválhatjuk BASIC alól, egy korábban felcsatolt TAP fájlt a `LOAD ""` paranccsal tölthetjük be, stb. Programozhatunk is, ha valaki még nem unta meg harminc év alatt a BASIC nyelvet.

## 128K BASIC

Ezt az üzemmódot a 128K BASIC menüponttal (U gomb) érhetjük el. A 128K-s gép menüjéhez hasonlót kapunk induláskor, amely a 48K BASIC pont alatt egy TR-DOS menüponttal egészült ki, jelezve hogy a

Betadisk 128 bővítő kezelése integrálva van, ez Pentagon gépekről lehet ismerős. Sajnos a Betadisk funkciók NEM működnek 128K BASIC alól.

## Setup menü





Az ERS Setup menüje alól a következő beállításokat érhetjük el:

- *S. Sound on keys*: hang visszajelzés billentyűlenyomásra.
- *G. Access SD NeoGS*: Ha van NeoGS hangkártyánk, akkor annak az SD kártyáját is elérhetjük fájlok tárolására, TRD, SCL, FDI illetve TAP fájlok kezeléséhez.
- *K. DRV AY printer*: Megadható, hogy van-e párhuzamos nyomtató csatlakoztatva a számítógéphez.
- *T. Autostart tape*: Ha a főmenüben az *L.Emu tape load* be van kapcsolva, akkor ez a menüpont vezérli, hogy egy TAP fájlt a számítógép azonnal betöltsön, vagy csak BASIC üzemmódból lehessen ezt megtenni (lást fentebb).
- *H.Delay for HDD, secs*: Ha IDE (vagy IDE2SATA adapterrel SATA) merevlemezt csatlakoztattunk az alaplaphoz, akkor annak megfelelő felélesztéséhez szükséges lehet induláskor egy kis várakozási idő. Ezt adhatjuk meg itt, 0 és 9 másodperc között. 0 esetén az alaplap nem vár, azonnal megpróbálja detektálni az esetlegesen csatlakoztatott merevlemezt.

## Service menü

Az innen elérhető lehetőségek röviden:

- *R.Reset NeoGS*: ha van NeoGS hangkártyánk, és az esetleg lefagyott, vagy pl. egy játékból rosszkor léptünk ki resettel, és továbbra is szól a hang, akkor ezzel alaphelyzetbe állíthatjuk.
- *E.Reset CMOS*: a ZX Evolution tartalmaz egy 256 bájt méretű, elemmel védett CMOS órát, ami tartalmaz egy kis mennyiségű memóriát is. A korábban felsorolt beállítások értékei itt tárolódnak. Ennek a CMOS memóriának a tartalmát állítja vissza alaphelyzetbe ez a menüpont.


- *K.Format ramdisk 640k*: A RAM-diszket formázza meg üres, standard 640KB-os TR-DOS formátumra. Ez után a RAM-diszket hasonlóan használhatjuk pl. a Perfect Commander, vagy az Evo DOS alól, mintha egy SCL vagy FDI image-et csomagoltunk volna ki.
- *V.Basic 48 standart*: A standard 48K BASIC ROM-ot indíthatjuk, annak minden eredeti hibájával. A főmenüből elérhető 48K ROM egy javított ROM-ot tartalmaz.
- *B.Basic 128 standart*: A standard 128K BASIC-et érhetjük el. Összehasonlítva a főmenüből elérhető 128K-s ROM-mal, itt a TR-DOS nincs a menübe integrálva, helyette a 128K-s ZX Spectrum Tape Tester funkcióját kapjuk.

- *M.Edit CMOS*: kézzel szerkeszthetjük a CMOS memória tartalmát. Bizonyos meghatározott bájtokhoz információt is kiír a számítógép, így pl. beállíthatjuk az időt, dátumot.
- *S.Test PC keyboard*: A PS/2 billentyűzet gombjait tesztelhetjük ezzel a funkcióval.
- *C.ATM CP/M*: A ZX Evolution 512KB ROM-jában egy CP/M rendszer is helyet kapott, innen lehet elindítani. Ez önmagában is egy külön téma, talán egy másik cikkben visszatérünk rá.
- T.IS-DOS boot: IS-DOS-t indíthatunk. A CP/M-hez hasonlóan ez is megérdemel egy saját cikket. Türelmetlenek és angolul olvasni tudók számára ajánlott a http://www.pouet.net/prod.php?which=66207 címen elérhető *Info Guide #11* e-zine, vagy diskmag, amelyben egy háromrészes cikk szól az IS-DOS-ról, illetve annak *TASiS* néven futó továbbfejlesztéséről.
- *F. Fast update ROM*: A ZX Evolution teljes 512kB-os ROM-ját flash-elhetjük újra, a fájlnév kötelezően *zxevo.rom*.
- *U.Update custom ROM*: A ZX Evolution 512kB-os ROM-jában a második 64kB-os szelet eredetileg üres, ezt a fenti menüpont csak törli, de nem flash-eli. Ezzel a menüponttal tudunk egyéb ROM-okat is tesztelni.
- *N.Dismount image*: Korábban felcsatolt TR-DOS lemezeket tudunk itt lecsatolni.

## Egyéb beállítások

A főmenü bal oldalán maradt még három beállítás, amelyről nem esett szó. Ezek:

1. *G.Reset*: A ZX Evolution alapkonfigurációja az ERS (ebben a beállításban itt *EVO Service* néven szerepel), és a *Service* menüből indítható IS-DOS illetve CP/M mellett gyárilag további kétféle fő üzemmódban képes működni. Ezeket választhatjuk itt ki, amelyek csak további resetelés után érhetők el. Ez a két üzemmód a *GLUK Service*, és az *EVO ProfROM*. Ezekről részletesebben a ZX Evolutionról szóló cikk második részében írunk majd. Ha beégettük a *Custom ROM* tartalmát is (például egy *Gosh Wonderful* BASIC ROM-ot), akkor ezt a lehetőséget választva az a ROM indul el reset után. Fontos még megemlíteni, hogy a billentyűzet segítségével visszaállhatunk az ERS üzemmódba: a *0* számbillentyűt (USA kiosztás szerint!) kell nyomva tartanunk, miközben megnyomjuk a soft reset gombot, illetve az *F12*-t.
2. *E.Reload FONT*: Egyes esetekben (pl. ha játékok felülírják a RAM tartalmát) reset után a képernyőn szemetet látunk. Ezt előzi meg ez a beállítás, amely újratölti a ROM-ból RAM-ba a képernyőn használt karakterkészletet.
3. *J.Type FONT*: Kétféle karakterkészlet közül választhatunk, *CP866* (a 866-os DOS-os kódkészlet, az ASCII karaktereken túl cirill betűk is megjeleníthetők), illetve a szintén NedoPC csoport által készített *ATM* ZX Spectrum klón által használt *KOI* karakterkészlet.

## PS/2 billentyűzettel elérhető extra funkciók

A ZX Evolution-nek van néhány, csak PS/2 billentyűzetről elérhető funkciója is, ezek közül párat már említettünk a cikkben korábban, itt pedig összegyűjtöttük az összeset.

- *PrintScreen/SysRQ*: NMI menü
- *ScrollLock*: VGA/TV üzemmód váltás. Ez a gomb vált a TV 15kHz-es és a VGA 31kHz-es sorfrekvenciával dolgozó üzemmódjai között. Sok modern monitor nem képes 15kHz-es képet kezelni. Az első bekapcsoláskor a 15kHz-es üzemmód az alapértelmezett, így lehetséges, hogy fekete képernyőt, vagy a monitor által jelzett hibaüzenetet látunk. A *ScrollLock* LED világít VGA üzemmódban.
- *NumLock*: TAPE/BEEP átkapcsolás. Választható, hogy a sztereó 3,5mm-es jack vonalkimenetre az 1-bites beeper hangja, vagy BASIC alól a SAVE parancs esetén a mentés hangja menjen ki. A billentyűzeten a TAPE állás esetén a *NumLock* LED világít.
- *F12*: Z80 CPU reset (soft reset), azonos funkciót lát el, mint működés közben a bekapcsoló gomb rövid megnyomása.
- *Ctrl-Alt-Del*: teljes alaplap reset, azonos funkciót lát el, mint a számítógép reset gombja, illetve ugyanaz játszódik le, mint bekapcsoláskor.
- Ha csatlakoztattunk PS/2 egeret is, akkor a bal egérgomb lenyomva tartásával és a numerikus billentyűzeten a + illetve - gombokkal az egér érzékenységét növelhetjük, illetve csökkenthetjük.

## Reset funkciók

Bizonyos extra funkciók kiválthatók a soft reset (bekapcsoló gomb) és egy billentyű megnyomásával. Emlékezzünk rá, hogy a billentyűzet F12 gombja is reset gombként szolgál.

- H+reset: Kiírja a reset esetén használható funkciókat.
- *0+reset*: Ha a főmenüben a G billentyűvel megváltoztattuk, hogy melyik üzemmódban (*GLUK Service*, *ProfROM*, *Custom ROM*)

induljon újra reset után ZX Evolution-ünk, akkor ezzel a billentyűkombinációval állíthatjuk vissza az ERS indulását. A helyes gombnyomási sorrend: nyomjuk le (és tartsuk nyomva) a *0*-át, nyomjuk le és engedjük fel a resetet, végül engedjük fel a *0*-át.

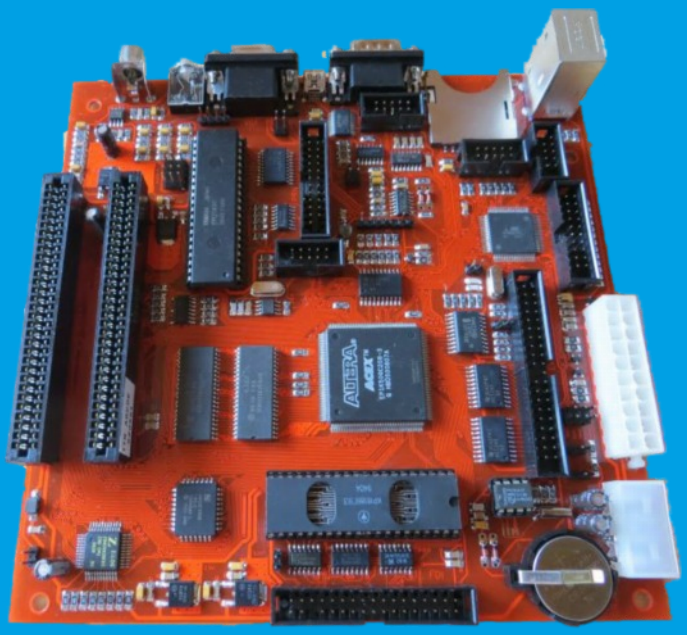
- CS+*reset*: A *CapsShift* és a reset egyidejű megnyomásával ZX Spectrum 128K üzemmódban indul újra a gép, ugyanúgy, mintha a főmenü *U* gombját nyomtuk volna meg.
- *SS+reset*: A *SymbolShift* és a reset egyidejű megnyomásával ZX Spectrum 48K üzemmódban indul a gép reset után. Mintha a főmenü *I* gombját nyomtuk volna meg.
- *D+reset*: Egy *GRASS* nevű demó indul el ROM-ból.
- *S+reset*: Egy vészhelyzeti CMOS editort indíthatunk el.
- C+*reset*: Egy színskálát rajzol a képernyőre, és a COVOX-on hangot is ad.

## Firmware és ROM frissítés

Az AVR egy programozható mikro-kontroller család, az FPGA pedig programozható elektronika illetve logika. A Z80 CPU mellett ezek alkotják a ZX Evolution hardver nagy részét. A ZX Evolution fejlesztői lehetővé tették, hogy – az AVR-ben található boot loader kivételével – a teljes rendszer frissítése az SD kártyáról egyszerűen megoldható legyen.
Az AVR/FPGA kódot tartalmazó *zxevo_fw.bin* fájlt az SD kártya gyökérkönyvtárába kell másolnunk PC-ről. Az SD kártyát az Evoba helyezve, a hard reset gomb nyomva tartásával nyomjuk meg a bekapcsoló gombot, majd engedjük el a reset gombot. Ekkor a boot loader először egy pár másodperc alatt frissíti a rendszert, majd már az új firmware-rel indul a számítógép.
ROM frissítéshez az ERS *Service* > *Fast update ROM* menüjéből a teljes 512kB-os ROM tartalmat újra flash-elhetjük. A *zxevo.rom* fájlt, hasonlóan egy TAP vagy TRD fájlhoz, a fájlböngészőn keresztül választhatjuk ki.
A *zxevo.rom* fájlban egy 64KB-os ROM terület üres, ezt az ERS *Service* > *Update custom ROM* funkciójával tölthetjük meg. Ezen keresztül tesztelhetünk pl. saját, kísérleti ROM-okat, vagy a *BaseConf* mellett a másik legnépszerűbb ZX Evo firmware, a *TSconf* ROM-ját is ezen keresztül flash-elhetjük.

***Böszörményi Zoltán (Zboszor)***
*(Folytatjuk …)*

***Dokumentációk:***
Eredeti orosz, illetve angol nyelvű leírás is elérhető a következő címekről:
http://www.nedopc.com/zxevo/rom/evo_reset_service.pdf
http://www.nedopc.com/zxevo/rom/evo_reset_service_eng.pdf
Az alaplappal együtt érkezik a kinyomtatott kézikönyv is, amely elérhető innen is:
http://nedopc.com/zxevo/zxevo_user_manual_revc_eng.pdf

# M/ZX Jelkezz, Jelkezz!

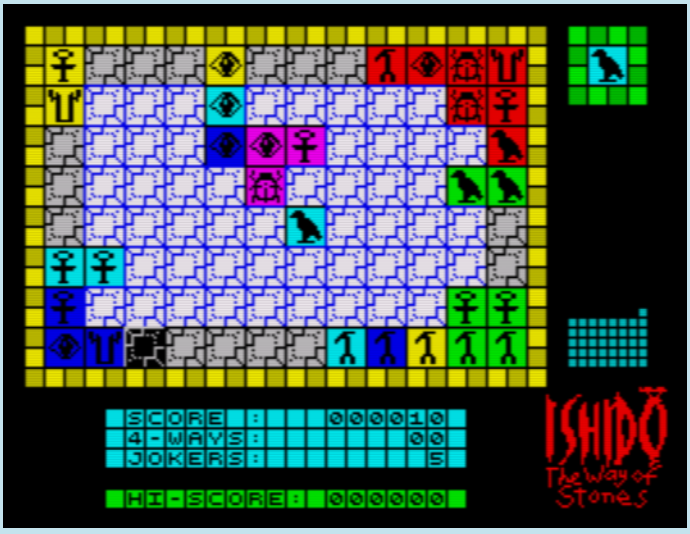
Azt már egy ideje megszokhattuk, hogy M/ZX rendre megörvendeztet minket színvonalas, tartalmas **FanZiX** kiadványaival - melyek terjedelemben sem szűkölködnek -, sőt azt is, hogy néha még más izgalmas kirándulásokra is invitál bennünket mondjuk egy **ZX SpeKtrum** magazinnal.

Azonban nagy meglepetés volt, amikor első logikai játékának, az *Ishido: The Way of Stones* megjelenéséről ovashattunk. Természetesen a tőle megszokott igényességgel készítette el legelső játék átiratát is, melyet ráadásul kisvártatva egy saját ötletből táplálkozó újabb logikai játék követett a *Save the Trees*.

Valószínüleg már szempillánk sem rebenne, mikor bejelentené legújabb hardver fejlesztését. De félre téve a tréfát, vegyük górcső alá az előbb említett Spectrum játékokat. Nem csak azért, mert a saját berkeinken belülről kerültek ki, de egyáltalán nem sárgák és savanyúak, hanem szép érett, igazi narancsok ám ezek.

Mindkét játék assembly-ben készült, az előbbi fejlesztéséről cikksorozatot is olvashattok hasábjainkon. Jellemzőjük a szépen kidolgozott képernyők, játékmenetek, némi Hewson utánérzéssel. Érdemes letölteni a teljes csomagokat, hisz minden részletre kiterjedő igényességel elkészített többnyelvű útmutatót, kazetta borítókat is találhatunk bennük, olyannyira, hogy még a kazettára ragasztható címke is megbújik a csomagban. Az Ishido estében nem is egy kazettaborítót lelhetünk, hanem több stílusban is készült verziók is megtalálhatóak.

## Ishido: The Way of Stones




Az Ishido egy logikai táblás játék, ami egy 96 mezős táblán játszható 72 kővel. Minden kőnek két jellemzője van: a színe és a szimbóluma. Hat szín és hat szimbólum van, így 36 különböző kő van. Mindegyik kőnek megvan a párja, így egy kőkészletben összesen 72 kő van. Az Ishidoban az elsődleges célod, hogy mind a 72 követ elhelyezd a táblán. A kihívás abban van, hogy a köveket csak olyan kövek szomszédságába helyezheted, melyeknek színe vagy szimbóluma a letett kőével egyező. Mikor a tábla kezd megtelni, nagyon megnehezíti ezt az egyszerű célt. A 4-Way a legértékesebb lépés, ahol egy követ négy másik közé raksz le, kettővel egyezik a lerakott kő színe és kettővel a szimbóluma.

## Save the Trees!


A Save The Trees! egy egyszerű logikai játék. Mentsd meg a fákat! Mentsd meg a parkodat! Veszélyes hernyók támadták meg a park fáit. Mentsd meg a kedvenc fáidat, a buhériai tölgyeket! Hogy ezt megtehesd, fel kell használnod a tölgyek legnagyobb ellenségeit, a katatón fabogarakat! Színek jelzik a hernyók sűrűségét. A fehér területek hernyómentesek, a kék területek a legfertőzöttebbek. Helyezd a bogarakat a fertőzött területekre, hogy csökkenjen a hernyók száma. A telepítési terület mérete 2x2, 2x3, 3x2 vagy 3x3 egységnyi lehet. A feladatod, hogy hernyómentesítsd a parkot. A játéknak 48 szintje van.


***Kardos Balázs (Balee)***

VADONATÚJ LOGIKAI JÁTÉK!

# SAVE THE TREES!


**”EZ A JÁTÉK EGY NAGYON VONZÓ TÉMÁT DOLGOZ FEL!”**
*(Hernyóellenes Liga)*

**”HA JÁTSZOTTAM VOLNA VELE, BIZTOS TETSZETT VOLNA!”**
*(Pistike)*

**”AZOK A KUKACOK NAGYON RONDÁK, VALAMI HASONLÓT MÁR ETTEM EGY MŰSORBAN”**
*(VV Alfonz, celeb)*

**”KÁR, EZEK A FÁK TÚL LOMBOSAK, SZÉP ÉPÍTÉSI TERÜLETET ALAKÍTHATTUNK VOLNA KI A PARK HELYÉN HA ELPUSZTULNAK!”**
*(egy szimpatikus építési vállalkozó)*

**”KÖVETELJÜK A KATATÓN FABOGARAK KÁRTALANÍTÁSÁT!”**
*(Az Elnyomott Bogarak Jogvédelmi Szervezetének közleményéből)*

FanZiX

www.fanzix.hu

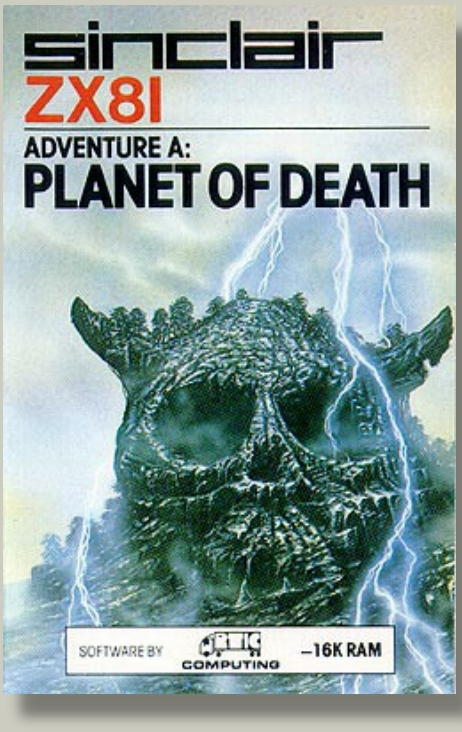
**NEM ÉRTED MIRŐL VAN SZÓ? TÖLTSD LE INGYENESEN A JÁTÉKOT ÉS MEGTUDOD!**

# ZX81 KLASSZIKUSOK - 2. RÉSZ

Ebben a számban is szeretnénk folytatni a sort a fontosabb ZX81 opuszok áttekintésével. Gyanítom tartogathat meglepetést is ez a kis sorozat, de ha másra nem is, hát egy kis nosztalgiázásra remek lesz.

## PLANET OF DEATH
### (1981 - ARTIC COMPUTING)

Több apropója is van, hogy ebben a részben szót ejtek erről a játékról. Az első, hogy ez volt az első gyári programom életemben, melyet a ZX81-hez kaptam. Már a címe is izgi, a „Bolygó neve, halál!" A kívülről rendkívül tetszetős borító valójában egy kevésbé, helyesebben egyáltalán nem látványos szöveges kalandjátékot rejt, bár ZX81 esetében talán ez minden darabról elmondható, ez alól talán némi kivételt jelenthetnek a nagyfelbontású játékok, de ezekről majd egyszer máskor értekezem.




A másik ok, amiért érdemes foglalkozni vele, hogy ez volt az első kalandjátékok egyike, ami megjelent a ZX81-re még annak megjelenési évében. Amúgy a sztori is felkeltheti az érdeklődünket iránta. Az alaptétel, hogy egy idegen bolygóról kellene elmenekülnünk, bár homály fedi, hogy miként is kerültünk oda. De végül is mindegy, kutassuk csak fel az űrhajót, meg egy számítógépet, miközben a zöld emberkékkel is érdemes barátkozni. A játék gépi kódban íródott és egy egyszerű IGE-FŐNÉV alapú parszerrel bír. Nagyjából húsz helyszínt és tárgyat tartalmaz és elég sok, több mint 1000 szót ismer. Mindez kb. 12 kbájtban elfér. Igazából egy sorozat első darabja is, ez ez Adventure A. Eleinte az A, B, C darabokat reklámozták, de idővel bővült a sor. Az Adventure B az Inka átok *(Inca Curse)* egy Inka templomban játszódik, már sokkal több, ötven helyszínen zajlik az élet, 25 tárggyal kell mókolni, ebben a kincsek nincsenek benne, ráadásul ahogy a többi epizódban is, ezeket a megfelelő helyen és időben kell felhasználni.

Sorban a harmadik a Végzet hajója *(Ship of Doom)* a legnagyobb ezek közül a maga 35 helyszínével és 40 tárgyával. Itt a feladat, hogy az űrhajónkat kiszabadítsuk egy gravitron sugárnyaláb fogságából és elhúzzuk a csíkot. Ehhez egy gombot kell megnyomnunk valahol és nyilván visszatalálnunk a hajónkhoz. Ha lehet hinni a szóbeszédnek, egy rejtett ajtón átjuthatunk egy másik szintre.

Egy szó mint száz, egy próbát azért megérnek…

```
(C)1981 ARTIC COMPUTING

WELCOME TO ADVENTURE A
YOU CAN HAVE AN ADVENTURE IN
YOUR OWN HOME.YOU COMMAND ME
WITH SHORT SENTENCES.

SOME USEFUL WORDS ARE :-
INVENTORY - TELLS WHAT YOU HAVE
REDESC - REDESCRIBE LOCATION
QUIT - TO RESTART THE GAME
ALSO GET,PUT,USE,WITH AND MANY
MORE.(OVER 100 IN FACT).
HIT ANY KEY TO START
```

***A sorozat darabjai:***

- Adventure A: *Planet of Death* (1981)
- *Adventure B: Inca Curse* (1981)
- *Adventure C: Ship of Doom* (1982)
- *Adventure D: Espionage Island* (1982)
- *Adventure E: The Golden Apple* (1983)
- *Adventure F: The Eye of Bain* (1984)
- *Adventure G: Ground Zero* (1984)
- *Adventure H: Robin Hood* (1985)

Természetesen szépen megjelentek ZX Spectrumra is változatlan formában.


**Kardos Balázs (Balee)**
*Folytatjuk…*

# ISHIDŌ

## The Way of Stones

2007 óta az első magyar ZX Spectrum játék!


**”EGYSZERŰEN ISHIDO KELL MINDENKINEK!”**

*(Amiga Joker, 1990. december)*

**”SZUPER! NEM LEHET SZABADULNI TŐLE!”**

*(Power Play, 1989. szeptember)*

**”EZ EGY NAGYON, NAGYON JÓ JÁTÉK... ÉS ELFÉR 16K-BAN.”**

*(Pegaz, WOS fórum, 2015. október)*

**”SZERETEM! NAGYON BOLDOG VAGYOK, HOGY VALAKI SPECTRUMRA KONVERTÁLTA - RÁADÁSUL ILYEN JÓL!”**

*(retromad, WOS fórum, 2015. október)*

**”NAGYON JÓL SIKERÜLT KONVERZIÓ - GYÁRI MINŐSÉG”**

*(csory, Speccyalista Kerekasztal, 2015. október)*

**”FANTASZTIKUS! NEMCSAK A JÁTÉK, HANEM A KÖRÍTÉS IS”**

*(Zozosoft, Speccyalista Kerekasztal, 2015. október)*

FanZiX

www.fanzix.hu

**INGYENESEN LETÖLTHETŐ!**

# PROGRAMOZÁSTECHNIKA - BASIC

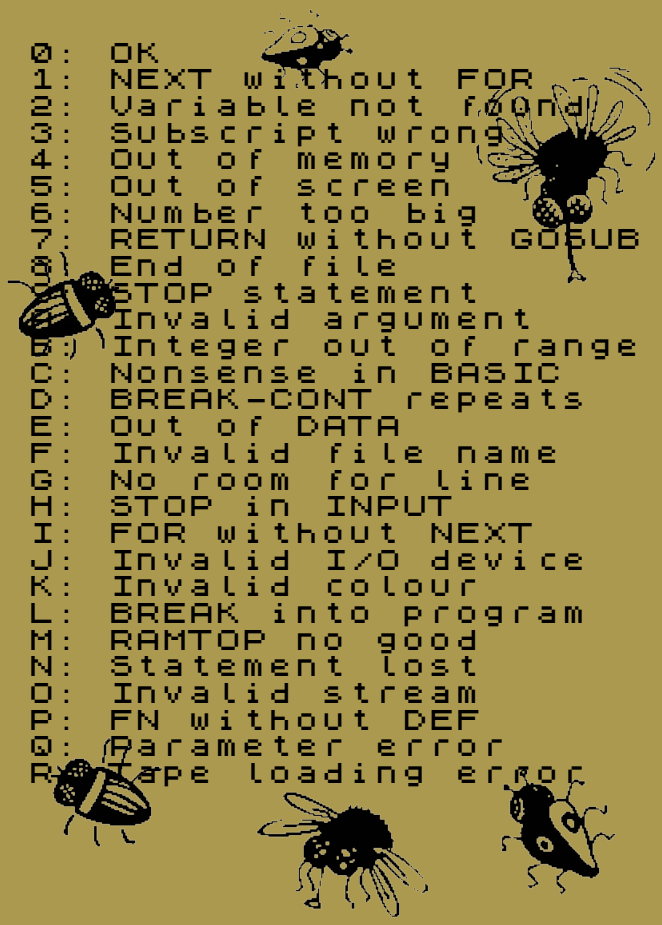
## ZX SPECTRUM TV-BASIC KÜLÖNKIADÁS - 2. RÉSZ

**Kedves Bézikül Beszélő Olvasó!**


Sorozatunk második részében feleleveníthetnénk az előzőekben megismert BASIC utasításokat és érdekességeiket, de helyette folytassuk a következő, minden kezdő BASIC programozót érintő lényeges résszel, a ZX Spectrum hibakezelésével.

**A hibaüzenetek**

Gyakorlásaink során már láthattunk jópár hibaüzenetet megjelenni, amelyek szerencsére szöveges formában íródtak ki utalva a hibára (nem úgy, mint kezdő ZX81-es társainknak, akiknek a **2/10** sokat nem segített), de valóban olyan „szerencsések”?




Talán még soha nem olvastuk így ezeket végig. Már a sorrendjük is sokat elárul a Spectrum fejlesztési csoportmunkájának menetéről, felosztásáról, mert keverednek rendesen össze-vissza (akinél épp mi kellett), de most ne ezt elemezzük. Alaposan megnézve, illetve megfigyelve kapcsolatukat egymással és az utasításokkal, tolongnak az értelmetlenségek:

Minek a **No room for line**, amikor van **Out of memory**? Értelmük ugyanaz, miszerint elfogyott a szabad memória.

```
PRINT 100/0
```

A 0-val osztáshoz rendelt **Number too big** kicsit tréfásnak hat.

```
PLOT 180,180
```

PLOT, DRAW és egyebek esetén helytelen koordináták esetén miért **Integer out of range** a szép **Out of screen** helyett?

```
FLASH 2016
```

**BRIGHT 2, FLASH 3, INVERSE 4, OVER 5**-nél is kézikönyvhöz nyúl az emberfia az **Invalid colour** hibaüzenet megfejtéséhez. Miért nem egy **Integer out of range**, ha már 255 felett úgyis az jön? Egyáltalán minek ez az üzenet, hiszen lehetett volna úgy értelmezni, hogy 0 esetén „nem”, a többi értéknél „igen”.

**FOR without NEXT**? Elég nehéz ám előidézni, próbálja csak meg bárki, mielőbb tovább olvas!

Megoldása, hogy csak akkor jelenik meg, ha a kezdőérték nagyobb a záróértéknél és van **STEP** megadva és nincs **NEXT** valahol a következő sorokban ... de minek is ezért üzenni? Ha növekvő a ciklus, akkor

hiba nélkül elfogadja **NEXT** nélkül is!

```
10 FOR F=1 TO 100
20 PRINT "ELFOGADVA."
```

```
10 FOR F=100 TO 1
20 PRINT "ELFOGADVA."
```

```
10 FOR F=100 TO 1 STEP -1
20 PRINT "EZ NEM MEGY!"
```

A **GO TO** 1-től 32767-ig majdnem rendben, mivel nincs hibaüzenet, bár megkérdőjelezhető, hogy miért is fogad el 9999-nél nagyobb értéket, hiszen beírni nem lehet ilyen sorszámú sort, de aztán jön a 32768-61440 tartomány **Statement lost**-ja, ami felettébb elgondolkodtató, majd e felett a meglepő **Integer out of range.**

A **Nonsense in BASIC** ritka, mint a barna szín Spectrumon, de gondolkozzunk el, miért nem írhattuk be Interface I nélkül azokat a kedves kis utasításokat, ha már rákerültek a billentyűzetre. Szegény kézikönyvet búvó kezdők ezerszer is megnézhették a **CAT 1** utasítás helyes beírását, mindig csak villogó kérdőjelet kaptak.

Ha meg olyan programot töltöttünk be, amibe egy szerencsés Interface I tulajdonos beírta azokat, jött a fenti kézrecsapás (hát ez mégiscsak nonszensz!), amiről pedig végleg nem tehettünk. Talán jobb lett volna simán átlépni, mintegy **REM** utasításként kezelve.

Halkan megkérdezhetnénk, hogy akkor miért működik az **LLIST, LPRINT** nyomtató nélkül? A **COPY** utasítás meg végleg összezavarja a kezdő program-kalózokat, de vajon miért nem **LCOPY**?

Az **End of file** csak a programozó képzeletében létezett, mert a Spectrum BASIC-ben dísznek van, soha nem jön elő, de szintén lehetetlen **Invalid I/O device** vagy **Invalid stream** hibát megkapni. Mire vagy mihez is készülhettek, hacsak a nyomokban sem létező Microdrive-hoz, de végül az sem ezeket használta.

Zárásként a tréfa kedvéért egy soha nem látott, titkos hibaüzenet, amelyet a véletlenek hoztak össze:

```
POKE 23610,28
```


**D: BREAK-CONT repeats**
de ismétlés helyett inkább

**FOLYTATJUK!**

***Pgyuri***

## IDÉZET

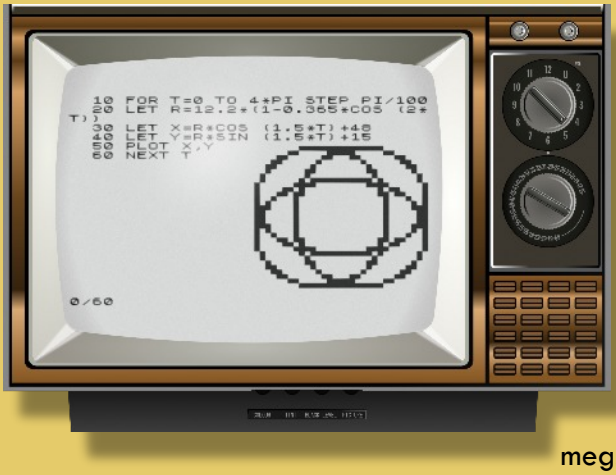
**Sir Clive Sinclair** egy 1989-es PCW-nek adott nyilatkozatából:

“I can certainly see the day when one might regret games machines - when they become so realistic that they seem just as real as life and that people can become addicted to them as they are addicted to drugs."
azaz

“Tisztán látom magam előtt azt a napot, amikor egyesek már bánni fogják a játékgépek [létrejöttét] — amikor már annyira élethűvé válnak, mint maga a való élet, és a játékosokat legalább annyira függővé tudják majd tenni mint a drogok.”

# A ZX81 KÉPALKOTÁSA

A Speccyalista Világ második számában megismerhettük a televízió képalkotás alapjait a multicolor effektusok kialakítása kapcsán Pgyuri programozástechnikai rovatában.
A programozástechnikát kiegészítve némi hardverrel, ismerjük meg a teljes képalkotás folyamatát a ZX81 működésében.




## Először nézzük meg tüzetesebben a gép kapcsolási rajzán fellelhető néhány érdekességet.

Az ULA és CPU között az adat és címvezetékeken (A0-A8) leválasztó ellenállásokat találunk, mégpedig olymódon, hogy az ULA a ROM-mal van egy oldalon a címvezetékeken, viszont az adatvezetékeken az ULA a CPU-val van közvetlen és a ROM-RAM-tól leválasztott kapcsolatban. Tehát az ULA a CPU által a memóriákból kiolvasott adatot módosíthatja, és a CPU-tól függetlenül címezheti a ROM egyes tartományait.
Másik érdekesség, hogy a CPU megszakítás bemenete (INT) közvetlenül össze van kötve az A6 címvezetékével. Tehát, ha az INT engedélyezve van, minden olyan CPU írás olvasás tevékenységnél ahol A6 alacsony szintre kerül, egy megszakítási ciklus indul. Látható még egy egyszerű, tranzisztorral megvalósított kapuáramkör a CPU WAIT bemenetére a HALT és NMI jelekről.
Ez várakozási ciklusokat generál a HALT utasítás végrehajtásakor, amíg az NMI bemenet jele visszatér magas szintre.

## Néhány szakirodalmakból ismert adat

- A RAM nem csak a 4000h-tól, hanem C000h-tól is látszódik a memória területen. Gépikódú programok futtatása a felső területen nem lehetséges.
- A karakterek grafikus képe a 8K ROM-ban 1E00h-tól 512 bájt hosszon található.
- A Z80 frissítési ciklusa az utasítás lehívás után történik, ilyenkor az adatvezetékekre az R (A0-A7) és I regiszter (A8-A15) értéke kerül.
- Az ULA 64 mikroszekundumonként állítja elő a videójel sorszinkron impulzust, ami a CPU NMI bemenetén megszakítást generál. Ezt a jelet a CPU egy OUT utasítással ki és bekapcsolhatja.
Kikapcsolt állapotban nincs megszakítás, nincs képalkotás, a CPU teljes sebességgel működhet, ez a FAST üzemmód.

## A képalkotás folyamata

Az NMI bemenetre 64 mikroszekundumonként megszakítás érkezik. A megszakítás programrészben növeli sorszámlálónak használt AF' regiszter értékét és ha még nincs képterület (border), visszatér a megszakításból. Az AF' regiszter értéke szerint a kép végén OUT utasítással képszinkron jelet generál a videojel kimeneten.

## Képterület megjelenítése

NMI és AF' vizsgálat után ugrás a nem teljesen dekódolt videó memória felső 32k „árnyék" lapjára.
Érdekesség, hogy itt tulajdonképpen a képterület gépikódú utasításként értelmezhetetlen végrehajtása történne az ULA beavatkozása nélkül.
A karakter kód utasítás lehívásakor M1 aktív, A15 magas, ha a beolvasott adat D6 bitje=0, akkor az ULA az adatbuszt leválasztó ellenállások segítségével a CPU-ra 00 adatot (NOP utasítást) kényszerít, tehát nem hajtódik végre a „karakterkód". D6=1 esetén végrehajtható az utasítás (minden sor végén 76H-HALT található).
Az utasításkód beolvasás után a NOP frissítési periódusában a CPU kiteszi a buszra az R és I (karaktergenerátor kezdet cím(ROM) felső bájt) regisztereket.

Az ULA a korábban megjegyzett (előzőekben lehívott „utasítás" valójában karakter) kódból és egy saját 3 bites (karakteren belüli sor-) számláló értékével módosítja a címbusz ellenállásokkal leválasztott alsó 9 bit-jét (A0-A2 ULA karakter sorszámláló, A3-A8 ULA karakterkód, A9-A15 CPU I regiszter), és ezáltal a ROM-ból megkapja az ábrázolandó karakter éppen megjelenítendő sorának bitmintáját, amit pontonként megjelenít a videó kimeneten.
A sorvégi HALT utasítás az előzőek miatt (D6=1) végrehajtódik, és a CPU egy megszakítás beérkezéséig NOP utasításokat hajt végre, ami az

előzőek szerint a space karakter folyamatos megjelenítését eredményezi.

Mivel a CPU A6 bitje össze van kötve az INT bemenettel, a HALT-onkénti NOP frissítési periódusa alatt az R regiszter megjelenik a címbusz alsó felén (amikor a CPU mintát vesz az INT bemeneten), és a képernyőterületre ugrás előtt az INT engedélyezve lett, az R regiszter értéke be lett állítva úgy, hogy a 32. frissítési periódus (sor vége) után egy INT megszakítás hajtódik végre.
Az INT programrészben a CPU regisztereiben számolt sorszámlálók állása szerint folytatódik tovább az előzőek ismétlésével a képalkotás a 24. sor befejezéséig.

A teljes 32 karakteres sor esetén az egyes karakterek megjelenítése alatt kényszerített NOP esetén is ugyanúgy növekszik az R regiszter, mint a sorvégi HALT okozta NOP esetében, így a sor végére érve ugyanannyi idő telik el az INT megszakításig.
Mivel a CPU különböző hosszúságú és ezáltal végrehajtási idejű utasításokat használ, az WAIT-HALT-NMI kapu kialakítás gondoskodik arról, hogy a sor megjelenítő rutin indulási ideje mindig azonos időre legyen a sorszinkron jelhez képest az NMI megszakítás elfogadása után.

A módszer segítségével elérhető, hogy a szűkös memória miatt, a képen az üres sor vagy a karakterek utáni üres hely a sor végéig csak 1 bájtot foglal a memóriában. A teljes üres képernyő csak 24 bájt (soronként egy HALT kód)!
Memóriabővítés esetén a rendszer már nem használja a tömörített képernyő memóriát, ezzel jelentős időt takarít meg a BASIC értelmező azzal, hogy a képernyőmemória utáni területet nem kell folyamatosan áthelyeznie a képtartalom (méret) változásakor.

Sajnos ennek ára van, mert a CPU a szoros ULA-val való együttműködés miatt más programot csak a képterület alatt és felett levő üres terület „rajzolásakor" tud futtatni.
A FAST üzemmód lekapcsolja a teljes képalkotást (megszünteti az NMI jelet), így lényeges sebesség-növekedést eredményez.

A rendszerváltozók és a képalkotásban működő program lecserélésével nagyobb felbontású grafikus képernyő is megjeleníthető, természetesen ehhez memóriabővítés is szükséges a nagyobb képernyőterület miatt.
A képernyő területnek mindenképpen látszódnia kell a felső 32k lapon is, ezt a feltételt 64k memória-bővítés esetén a felső 32k RAM M1 inaktív szintjére engedélyezésével oldják meg, ebben az esetben adat tárolásra használható a felső 32k.

A minimalista áramkör kialakítás miatt a videó/szinkron jelek minősége miatt az újabb televíziókon és monitorokon nem kapunk használható képet. Ezen segíthetünk a neten fellelhető egyszerű jelformáló, erősítő, szinkron utóváll előállító áramkör utólagos beépítésével.

**További információk és felhasznált irodalom:**


http://problemkaputt.de/zxdocs.htm
https://www.youtube.com/watch?v=1irH3KuGyI0

The ZX81 Video Display System - Wilf Rigter
ZX81 ROM Disassembly - Dr. Ian Logan & Dr. Frank O' Hara



***Gondos Csaba (Csaba, GoCom)***

## IDŐGÉP

### 1980.

A nyolcvanas évek legelején kezdi el az angol Ferranti cég az egyéni igények szerinti chipek gyártását, Sinclair rögtön ki is használja ezt. A ZX80-ból egy egész sor kis chipet felcserélik a "nagy ULA" chipre. A megspórolt pénzen kibővítik a ROM memóriát, áttervezik az alaplapot és 1981 márciusában piacra dobják a ZX81-et, mely ténylegesen megteremtette az angol személyi számítógép gyártást. A billentyűzetet is lecserélik és az előző szenzorai helyett, ebbe az új gépbe már fólia érintkezőket tesznek. Továbbra is egy gombnyomásra egy BASIC parancs kerül beírásra. Már nem minden billentyűleütésre ellenőrzi a beírt sort, hanem a NEW LINE lenyomása után. A BASIC már tud decimális számokkal is számolni. A számítási sebessége nem túl nagy, de egy új BASIC utasításpár (SLOW, FAST) segítségével növelhető, a FAST üzemmódban ugyanis nem kezeli a képernyőt! A külső csatlakozói azonosak a ZX80-nál látottakkal. Ez a gép már 64k-ra volt bővíthető és blokkgrafikára is képes volt.

## Egyszerű ütközés észlelés attribútum vizsgálattal

### Az attribútumok meghatározása

Bizonyára mindenki jól emlékszik az **ATTR** függvényre, aki egy kevés időt is eltöltött a Sinclair BASIC-ben történő programírással. E függvény segítségével tudtuk meghatározni az attribútumait bármely karaktercellának a képernyőn. Habár megértése elég nehézkes a BASIC programozó számára, nagyon jó szolgálatot tesz egyszerű ütközés észlelés készítésekor. Tulajdonképpen ez a módszer annyira jól használható, hogy a gépi kódú megfelelőjének alkalmazását számos piaci szoftverben megtalálhatjuk, valamint - ami nekünk fontosabb -, rendkívül jól használható a kezdő Spectrum programozó számára is.

Kétféleképpen határozhatjuk meg egy adott karaktercella színattribútum értékeit Spectrumon. Ha gyorsan végigpörgetjük a Spectrum ROM visszafejtését, találunk egy rutint a **9603**-as címen, ami elvégzi a munkát helyettünk, de ha akarjuk, magunk is kiszámíthatjuk a megfelelő memóriacímet.
A legegyszerűbb módszer egy attribútum érték megtalálására néhány ROM rutin használata:

```
ld    bc,(ballx) ; x és y értéke a bc
                   regiszterpárba.
call 9603        ; ROM hívás, hogy(c,b)
                   attribútum a verembe
                   kerüljön.
call 11733       ; attribútumok az
                   akkumulátorba.
```

Ezzel szemben sokkal gyorsabb, ha magunk végezzük el az attribútum kalkulációt! Szintén hasznos lehet, ha nem csak az attribútum értékét határozzuk meg, hanem a címét is a memóriában – például arra az esetre, ha módosítani szeretnénk.

### Attribútumcímek meghatározása

A Spectrum rendellenes pixel elosztásával szemben a színcellák a **22528** és **23295** zárt intervallumon belül helyezkednek el a memóriában sorfolytonosan, ahogy azt elvárnánk. Más szóval: a képernyő legfelső 32 attribútumcellája a **22528**-as címen kezdődik és a **22559**-es címig tart, balról jobbra. A második sor a **22560**-as címtől a **22591**-es pozícióig tart, és így tovább. Hogy megkapjuk az (x,y) kiíratási pozícióhoz tartozó cella attribútumcímét, egyszerűen megszorozzuk x-et 32-vel, majd hozzáadjuk y-t, és még 22528-at. Ha megvizsgáljuk az eredményül kapott címen található értéket, megkaphatjuk az általunk kiválasztott pozícióban megjelenített színadatot. A következő példa kiszámolja a (b,c) pozícióhoz tartozó attribútum helyét, és a címet elhelyezi a **HL** regiszterpárban.

```
; (b, c) helyen lévő karakter attribútumcímének
; meghatározása
atadd:
   ld   a,b        ; x pozíció
   rrca            ; szorzás 32-vel
   rrca
   rrca
   ld   l,a        ; eltároljuk l-ben
   and 3           ; bitek maszkolása a magas
                     bájton
   add a, 88       ; 88*256=22528, az
                     attribútumok kezdőcíme
   ld   h, a       ; magas bájt kész
   ld   a, l       ; x*32 újra
   and  224        ; alacsony bájt maszkolása
   ld   l, a       ; l-be be
   ld   a, c       ; y eltolás meghatározása
   add  a, l       ; az alacsony bájthoz
                     hozzáadjuk
   ld   l, a       ; hl=az attribútum címe
   ld   a, (hl)    ; attribútum visszaadása
                     a-ban
   ret
```

Ha megvizsgáljuk a hl címen található bájtot, megkapjuk az attribútum értékét. Amennyiben módosítjuk a hl címen található értéket a memóriában, megváltozik a megfelelő négyszög színe a képernyőn. Hogy értelmezni tudjuk a kiszámolt címen lévő értéket, ismernünk kell az attribútum értékét alkotó 8 bit elrendezését:

```
d0-d2   tinta szín 0-7,
        0=fekete, 1=kék, 2=piros, 3=magenta,
        4=zöld, 5=cián, 6=sárga, 7=fehér
d3-d5   papír szín 0-7,
        0=fekete, 1=kék, 2=piros, 3=magenta,
        4=zöld, 5=cián, 6=sárga, 7=fehér
d6      világosság,
        0=tompa, 1=világos
d7      villogás,
        0=változatlan, 1=villog
```

Ha például azt szeretnénk megvizsgálni, hogy zöld papírszín található-e a megadott helyen, az így nézne ki:

```
   and   56          ; a papírbiteken kívül
                       minden bitet maszkolunk
   cp    32          ; zöld(4) * 8?
   jr    z, green    ; igen, jöhet a zöld teendő
```

A sárga tinta tesztje e képpen alakul:

```
   and   7           ; csak a tinta biteket
                       hagyjuk meg
   cp    6           ; sárga (6)?
   jr    z, yellow   ; igen, jöhet a sárgaság
```

## Használjuk is fel a tanultakat a gyakorlatban

Például kibővíthetjük a Százlábú játékunkat attribútum-ütközésészleléssel. Ahogyan eddig is, az új részeket aláhúzással jelöltük:

```
; példa 5.1
; Fekete képernyőt szeretnénk

   ld    a, 71       ; fehér tinta (7) fekete
                       papíron (0),
                     ; világosan (64)
   ld    (23693), a  ; képernyő színek
                       beállítása
   xor   a           ; akkumulátor nullázásának
                       gyors módja
   call  8859        ; maradandó keretszín

; Grafika beállítása

   ld    hl, blocks  ; felhasználói grafikus
                       elemek címe
   ld    (23675), hl; az UDG ide mutasson

; Rendben, kezdődjön a játék!

   call  3503        ; ROM rutin - képernyő
                       törlése, 2-es csat.
                       nyitása

; Koordináták inicializálása

                     ; kezdeti koordináták a
                       HL-be
   ld    hl, 21+15*256
   ld    (plx), hl   ; játékos koordinátái

   call  basexy      ; x és y pozíciójának
                       beállítása
   call  splayr      ; játékos törzsének
                       megjelenítése

; Ez a főhurok

mloop:
  equ $

; Játékos törlése

   call  basexy      ; x és y pozíciójának
                       beállítása
   call  wspace      ; üres hely a játékos
                       pozíciójába

; Törölve van a játékos, átmozgathatjuk az új
; pozícióba mielőtt újra megjelenítjük

   ld    bc, 63486   ; billentyűk 1-5/joystick
                       port 2
   in    a, (c)      ; kiolvassuk a megnyomott
                       gombokat
   rra               ; legkülső bit = 1-es gomb
   push  af          ; megjegyezzük
   call  nc, mpl     ; ha megnyomva, mozgás balra
   pop   af          ; akku helyreállítása
   rra               ; következő bit (2-es
                       helyiérték) = 2-es gomb
   push  af          ; megjegyezzük
   call  nc, mpr     ; ha megnyomva, mozgás
                       jobbra
   pop   af          ; akku helyreállítása
   rra               ; következő bit (4-es
                       helyiérték) = 3-es gomb
   push  af          ; megjegyezzük
   call  nc, mpd     ; ha megnyomva, mozgás
                       lefelé
   pop   af          ; akku helyreállítása
   rra               ; következő bit (8-es
                       helyiérték) = 4-es gomb
   call  nc, mpu     ; ha megnyomva, mozgás
                       felfelé

; Az átmozgatás megtörtént,
; újrameglejeníthetjük a játékost

   call  basexy      ; x és y koordináta
                       beállítása
   call  splayr      ; játékos megjelenítése

   halt              ; várakozás

; Visszaugrás a főhurok elejére

   jp    mloop

; Játékos balra mozgatása

mpl:
   ld    hl, ply     ; Emlékezzünk, y a
                       vízszintes koordináta!
   ld    a, (hl)     ; Mi a mostani érték?
   and   a           ; Nulla?
   ret   z           ; Ha igen, nem tudunk tovább
                       balra menni!

; leellenőrizzük nincs-e gomba az útban

   ld    bc,(plx)    ; aktuális koord
   dec   b           ; 1 négyzettel balra nézünk
   call  atadd       ; annak a pozíciónak a címe
   cp    68          ; a gombák világosak (64) +
                       zöldek (4)
   ret   z           ; ha gombát találunk, nem
                       mehetünk arra

   dec   (hl)        ; különben y = y-1
   ret

; Játékos jobbra mozgatása

mpr:
   ld    hl, ply     ; Emlékezzünk, y a
                       vízszintes koordináta!
   ld    a, (hl)     ; Mi a mostani érték?
   cp    31          ; A jobb szélén vagyunk
                       (31)?
   ret   z           ; Ha igen, nem tudunk tovább
                       jobbra menni!

; leellenőrizzük nincs-e gomba az útban

   ld    bc, (plx)   ; aktuális koord
   inc   b           ; 1 négyzettel jobbra nézünk
   call  atadd       ; annak a pozíciónak a címe
```

```
   cp   68          ; a gombák világosak (64) +
                      zöldek (4)
   ret  z           ; ha gombát találunk, nem
                      mehetünk arra

inc (hl)            ; különben y = y+1
ret

; Játékos felfelé mozgatása

mpu:
   ld   hl, plx     ; Emlékezzünk, x a
                      függőleges koordináta!
   ld   a, (hl)     ; Mi a mostani érték?
   cp   4           ; a pálya tetején vagyunk
                      (4)?
   ret  z           ; Ha igen, nem tudunk tovább
                      felfele menni!

; leellenőrizzük nincs-e gomba az útban

   ld   bc, (plx)   ; aktuális koord
   dec  c           ; 1 négyzettel fentebb
                      nézünk
   call atadd       ; annak a pozíciónak a címe
   cp   68          ; a gombák világosak (64) +
                      zöldek (4)
   ret  z           ; ha gombát találunk, nem
                      mehetünk arra

   dec  (hl)        ; különben x = x-1
   ret

; Játékos lefelé mozgatása

mpd:
   ld   hl, plx     ; Emlékezzünk, x a
                      függőleges koordináta!
   ld   a, (hl)     ; Mi a mostani érték?
   cp   21          ; A képernyő alján vagyunk
                      (21)?
   ret  z           ; Ha igen, nem tudunk tovább
                      lefele menni!

; leellenőrizzük nincs-e gomba az útban

   ld   bc, (plx)   ; aktuális koord
   inc  c           ; 1 négyzettel lentebb
                      nézünk
   call atadd       ; annak a pozíciónak a címe
   cp   68          ; a gombák világosak (64) +
                      zöldek (4)
   ret  z           ; ha gombát találunk, nem
                      mehetünk arra

   inc (hl)         ; különben x = x+1
   ret

; A játékos törzsének, x és y koordináta érté-
; kének beállítása, ez a rutin kerül meghívás-
; ra a törzs törlése és megjelenítése előtt

basexy:
   ld   a, 22       ; AT pozícionáló kód
   rst  16
   ld   a, (plx)    ; játékos függőleges koord
   rst  16          ; beállítjuk
   ld   a, (ply)    ; játékos vízszintes koord
   rst  16          ; ezt is beállítjuk
   ret

; Megjelenítjük a játékost a jelenlegi PRINT
; pozícióban

splayr:
   ld   a, 69       ; cián tinta (5) fekete
                      papíron (0),
                    ; világosan (64)
   ld   (23695), a  ; beállítjuk az ideiglenes
                      színeket
   ld   a, 144      ; 'A' UDG ASCII kódja
   rst  16          ; játékos kirajzolása
   ret
wspace:
   ld   a, 71       ; fehér tinta (7) fekete
                      papíron (0),
                    ; világos (64)
   ld   (23695), a  ; beállítjuk az ideiglenes
                      színeket
   ld   a, 32       ; SZÓKÖZ karakter
   rst  16          ; üres hely megjelenítése
   ret

; Egyszerű pszeudo-véletlen szám generátor.
; Egy mutatót léptet a ROM területen (a seed-
; ben tárolva), visszaadva a megcímzett bájt
; tartalmát

random:
   ld   hl, (seed) ; Mutató
   ld   a, h
   and  31          ; Az első 8 KB-on belül
                      tartjuk
   ld   h, a
   ld   a, (hl)     ; "Véletlen" szám a mutatott
                      helyről
   inc  hl          ; Mutató léptetése
   ld   (seed), hl
   ret
seed:
   defw 0

; Kiszámolja a (dispx, dispy) koordinátánál
; lévő karakter attribútum értékének címét

atadd:
   ld   a, c        ; függőleges koordináta
   rrca             ; szorzás 32-vel
   rrca             ; Jobb léptetni maradékkal
                      3-szor gyorsabb, mint
   rrca             ; balra léptetni 5-ször
   ld   e, a
   and  3
   add  a, 88       ; 88x256=attribútumok
                      kezdőcíme
   ld   d, a
   ld   a, e
   and  224
   ld   e,a
   ld   a, b        ; vízszintes pozíció
   add a, e
   ld   e, a        ; de=attribútum címe
   ld   a, (de)     ; visszaadjuk az attribútum
                      értékét az akku-ban
   ret

plx     defb 0      ; játékos x koordinátája
ply     defb 0      ; játékos y koordinátája

; UDG grafika
blocks:
; játékos törzse
   defb 16,16,56,56,124,124,254,254
; gombák
   defb 24,126,255,255,60,60,60,60
```


***Tanács Imre (Kapitány)***

# PROGRAMOZÁSTECHNIKA

## HOGYAN KÉSZÜLT AZ ISHIDO: THE WAY OF STONES 2.

### *Avagy a kezdő programozó buktatói*

A rend kedvéért, a program fejlesztésével ott tartottam, hogy voltak SevenUp-ban készült rajzaim a játékban használt szimbólumokhoz és egy saját betűkészletem, amit Mopi fontszerkesztőjével készítettem, aztán volt egy egyre inkább megszokott, nagyon használhatónak tűnő alkalmazáscsokrom a fejlesztéshez. A szimbólumokat be tudtam építeni a betűkészletbe, négy-négy másra nem használt betű helyére, ezeket ki tudtam írni, rajzolni a képernyőre. Assembly programozás szempontjából: sok dologra rájöttem, sokra emlékeztem és még többet kísérletezgettem. Ezek eredménye majdnem 100 'asm' kiterjesztésű fájl lett, teljesen áttekinthetetlenül egy kesze-kusza könyvtárstruktúrában. Akkor éreztem úgy, hogy rendet kell tenni a tudományomban.

### *Hogyan is néz ki egy program?*

Az előző részben már szó volt róla, hogy használhattam volna a 'FONT' labelt, magyarul címkét, ezt ekkor már meg is tettem. De mi is az a címke? És egyáltalán hogyan is néz ki egy program? Erről az alapról megfeledkeznek azok, akiknek ez túl egyértelmű, pedig ez a tudás szükséges az áttekinthető programozás elkezdéséhez. Tehát egy egyszerű, átlátható assembly program valahogy így néz ki (pontosvessző után a megjegyzések vannak):

```
   org   25000       ; ha el akarom indítani a
                       programot, akkor az
                     ; 'org' utáni értéket
                       használom: RANDOMIZE USR
                       25000
   jp    MAIN        ; jp, tehát ugrás a MAIN
                       címkére, ahol a program
                     ; fő része kezdődik

; ÁLTALÁNOS ELJÁRÁSOK, FÜGGVÉNYEK

; például a KEYPRESS, a billentyű leütését
; figyelő függvény, itt a KEYPRESS egy
; címke, de a függvényen belül a KP1 is az

KEYPRESS
   push hl
                     ; 'hl' regiszter eltárolása
   ld    hl, 23560   ; a 23560-as címen van
                       tárolva az utoljára
                       lenyomott
                     ; billentyű ASCII kódja,
                       nulla, ha nem történt
                       lenyomás
   ld    (hl), 0     ; 'hl' regiszterbe 0-t tölt

KP1
   ld    a, (hl)     ; majd egy ciklusban addig
                       nézegeti 'hl'-t,
                     ; míg más értéke nem lesz,
                       tehát gombnyomásig
   cp    0           ; összehasonlítja nullával
                       'a' regiszter értéket
   jr    z, KP1      ; ha még mindig nulla, akkor
                       folytatja a ciklust
   pop   hl          ; egyébként 'hl' visszakapja
                       eredeti értékét
   Ret               ; a függvény vége, visszatér
                       a program a KEYPRESS
                     ; meghívása utáni sorra, 'a'
                       regiszter felvette a
                     ; lenyomott billentyű kódját

; A PROGRAM FŐ RÉSZE

MAIN
   ld    a,2
   call  5633
   ld    de, STRING1
   ld    bc, 5
   call  8252        ; szöveg kiírása
   call  KEYPRESS
   Ret

; MINDENFÉLE VÁLTOZÓK, AMIKRE SZÜKSÉG VAN A
; PROGRAMBAN

STRING1
   defb 'HELLO'
```

A megjegyzések néha feleslegesnek tűnhetnek, de egy saját forráskód későbbi megértéséhez sokszor mégis nélkülözhetetlenek, mert ami most egyértelműnek látszik, az akár néhány hét elteltével is zavarossá, érthetetlenné válhat.

Előbbi program lehetne az Ishido alapja is, ahonnan minden elkezdődik. Aztán folytatódhatna azzal, hogy nem 'HELLO'-t írna ki, hanem azt, hogy 'ISHIDO: ', gombnyomás után meg mellé, hogy 'THE WAY OF STONES'. Ez valahogy így nézne ki, csak a program fő része változna:

```
; A PROGRAM FŐ RÉSZE

MAIN
   ld    a, 2
   call  5633
   ld    de, STRING1
   ld    bc, 8
   call  8252        ; 1. szöveg kiírása
   call  KEYPRESS
```

```
   ld    a, 2
   call  5633
   ld    de, STRING2
   ld    bc, 17
   call  8252        ; 2. szöveg kiírása
   call  KEYPRESS
   ret

; MINDENFÉLE VÁLTOZÓK, AMIKRE SZÜKSÉG VAN
; A PROGRAMBAN
STRING1
   defb  'ISHIDO: '
STRING2
   defb  'THE WAY OF STONES'
```

Érezhető és látható is, hogy ez így nincs rendben: majdnem ugyanaz a programrész ismétlődik meg kétszer, sőt, ha még valamit ki akarnék írni ugyanilyen módon, akkor még többször. Hogyan tehetném hatékonyabbá a programot? Úgy, hogy az ismételgetés helyett egy karaktereket kiíró eljárást használok, aminek legyen a neve mondjuk PRINTSTR. A program így kisebb és áttekinthetőbb is lesz, az általános függvények, eljárások rész bővül, a KEYPRESS után így néz ki a program:

```
PRINTSTR
   push  bc          ; a szokásos utasítások
                       kerülnek ide, de mivel ez
                       egy
   push  de          ; általánosan használt
                       eljárás, nem kell megadnom
                       'de'
   ld    a, 2        ; és 'bc' regiszter értékét,
                       azokat az eljárás
   call  5633        ; meghívása előtt adom meg,
                       ugyanezen okból
   pop   de          ; viszont szükség van 'bc'
                       és 'de' regiszterek
   pop   bc          ; eltárolására és
                       visszaállítására, különben
   call  8252        ; a 'call 5633' utasítás
                       miatt menet közben
   ret               ; elállítódnának

; A PROGRAM FŐ RÉSZE

MAIN
   ld    de, STRING1
   ld    bc, 8
   call  PRINTSTR    ; 1. szöveg kiírása
   call  KEYPRESS
   ld    de,STRING2
   ld    bc, 17
   call  PRINTSTR    ; 1. szöveg kiírása
   call  KEYPRESS
   ret

; MINDENFÉLE VÁLTOZÓK, AMIKRE SZÜKSÉG VAN A
; PROGRAMBAN

STRING1
   defb  'ISHIDO: '
STRING2
   defb  'THE WAY OF STONES'
```

Innentől, ha szükségem lenne erre az egyébként túl egyszerű szövegkiíró eljárásra, akkor használhatnám anélkül, hogy emlékeznem kellene például arra, hogy a ROM melyik címeit kell közben meghívnom. Mindössze annyit kell tudnom, hogy 'de'-be meg kell adnom a szöveg címét, 'bc'-be pedig a hosszát, majd meghívnom az eljárást. Így bővül egy program, ami akár az Ishido is lehetne. A lényeg az átláthatóság és a hatékonyság.

### *Hogyan tovább? Egy kicsit Ishido specifikusan…*

Az előző részben elkezdett és türelmesen, sokáig folytatott módon felépült egy sor eljárásból és függvényből a programom egyre dagadó első fele. Lett egy használható véletlenszám generátorom, egy sokkal jobb karakterkiíró eljárásom, ami adott képpozícióba is tud írni, számok kiírására eljárásom, ami nem is olyan egyértelmű, mint amilyennek tűnik, vagy egy függvényem, ami kiolvassa egy kiírt karakter színét, aztán dallamsor megszólaltatására egy eljárás, meg még sok más, szépen egymás után, rendben, kommentelve.
Következő lépésben ki kellett találnom, hogy milyen folyamattal írható le a játék:
Az alapok lerakása: egy 12 mező széles és 8 mező magas tábla kirajzolása, valamint kell egy változó, ami tárolja, hogy melyik mezőn milyen kő található. Ez a változó dolog kis magyarázatra szorul, nem egészen úgy működik, mint más programozási nyelveknél. A változó, gyakorlatilag egy címke, neve bármi lehet, én TILES-nek neveztem el, 96 mező, az 96 bájt, egyszerűen lefoglalva a memóriából a defb-vel (a PASMO defb-ként szereti, más fordító db-ként), ami a 'define byte' rövidítése. A defb után vesszővel elválasztva vannak a számok, ezeket nem kötelező csoportosítva írni, ehelyett lehetne minden sorban az, hogy 'defb 0', csak az áttekinthetőség miatt van ebben az esetben 12 db nulla egy sorban. Ha értékadáskor azt írom például, hogy 'ld bx,TILES', akkor 'bx' regiszter a TILES címke memóriacímét veszi fel. Ha viszont azt írom, hogy 'ld a,(TILES+2)', akkor a címkét értékadáshoz használtam és az 'a' regiszter TILES 'változó' harmadik bájtjának értékét veszi fel. Így a tábla utolsó mezőjének értékét a (TILES+95) adja, de mi van, ha (TILES+96)-ot adok meg? Semmi különös, akkor a TILES 96 bájtja után következő első bájt lesz az eredmény, lenti programnál ez egyenlő a (TILESET)-tel. A TILESET címke után a kövek generálása következik: mind a 6 szín és 6 figura variációjából van két-két darab, tehát összesen 72 darab kő. Hogy néz ki ez egy assembly programban:

```
; Alaphelyzet: nincs egyik mezőn sem semmi, így
; a TILES 8x12 bájtjának értéke nulla egyébként
; a játék futása közben ezek az értékek változ-
; nak annak megfelelően, hogy milyen kő kerül
; egy-egy pozícióba, értékük szín*16+figura
; lesz

TILES
   defb  00,00,00,00,00,00,00,00,00,00,00,00
   defb  00,00,00,00,00,00,00,00,00,00,00,00
   defb  00,00,00,00,00,00,00,00,00,00,00,00
   defb  00,00,00,00,00,00,00,00,00,00,00,00
   defb  00,00,00,00,00,00,00,00,00,00,00,00
   defb  00,00,00,00,00,00,00,00,00,00,00,00
   defb  00,00,00,00,00,00,00,00,00,00,00,00
```

```
   defb 00,00,00,00,00,00,00,00,00,00,00,00


; a kövek készlete minden színből és figurából
; tartalmaz kettőt-kettőt, minden követ egy
; bájt képvisel, aminek az értéke
; szín*16+figura, a szín és a figura egy 1 és 6
; közötti érték lehet

TILESET
defb 17,17,18,18,19,19,20,20,21,21,22,22
defb 33,33,34,34,35,35,36,36,37,37,38,38
defb 49,49,50,50,51,51,52,52,53,53,54,54
defb 65,65,66,66,67,67,68,68,69,69,70,70
defb 81,81,82,82,83,83,84,84,85,85,86,86
defb 97,97,98,98,99,99,100,100,101,101,102,102
```

A kiinduló táblához még a kövek közül ki kell választani hatot kezdő kőnek úgy, hogy minden szín és minden figura képviseltesse magát. Ez mindjárt nem is olyan egyszerű feladat. Legegyszerűbb módja, ha a program véletlenszerűen húzogat ki a kövek közül egyet-egyet és ellenőrzi, hogy a kő megfelel-e a feltételeknek, azaz a színe és a figurája szerepel-e az addig kihúzott kövek között. Ha szerepel, akkor új kő kihúzása történik. Ez egy egyszerű, de az ötödik, hatodik kőnél már hosszadalmas eljárás. Le lehet a hatodik kő húzásával rövidíteni a folyamatot, hiszen akkor már úgyis csak egy szín és egy figura a lehetséges választás. De van egyszerűbb mód is, ha generálok egy-egy változót a hat színnek és a hat figurának:

```
COLORS
   defb 1,2,3,4,5,6
FIGURES
   defb 1,2,3,4,5,6
```

Majd a hat-hat elemet összekeverem egy megfelelő eljárással, ahol egy véletlenszerűen kiválasztott elemet megcserélek egy másik véletlenszerűen kiválasztott elemmel, és ezt megcsinálom még sokszor. Ezután megvan az első hat kő, amit ki is helyezek a táblára pl. egy karaktereket kiíró eljárással a megfelelő pozícióba. Valamint a TILES változóban felülírom a megfelelő értéket az első hat kőnek megfelelően, és a TILESET-ben az első hat kőnek megfelelő értéket kicserélem az első hat értékkel. A játék megkezdésének utolsó lépése, hogy a maradék 66 követ összekeverem, egy véletlenszerűen kiválasztott elemet megcserélek egy másik véletlenszerűen kiválasztott elemmel. A TILESET első hat eleme már kint van a táblán, a hetediket pedig kirakom a tábla mellé, hogy tudja a játékos, hogy milyen követ kell elhelyeznie a táblán.
Következő megoldandó feladatom, hogy a játékos egy kurzor segítségével le tudja rakni a következő követ a táblára. Erre kellett egy változó a kurzor X és Y pozíciójához, figyelni kellett a billentyűleütéseket és megfelelő billentyű lenyomása esetén változtatni X és Y értékét. Figyelni kellett rá, hogy X értéke csak 1-12, míg Y értéke csak 1-8 lehet. A kurzor megjelenítésének legegyszerűbb módjának azt találtam, ha villogtatom a kurzor alatt lévő táblamezőt. Természetesen nem elég az új pozícióban jelezni a kurzort, törölni kell az előző helyzetéből is (ehhez máris kell két új változó). Ezután már csak azt kellett vizsgálni, hogy a tűz gomb megnyomásakor kijelölt helyzetbe a következő kő lerakható-e a szabályokat betartva. Ehhez a következő lépésekre volt szükség:
A kijelölt mező négy szomszédja közül mennyiben van kő, tehát hány szomszédja van?

1. ha egy sem, akkor szabálytalan a lépés
2. ha egy, akkor abban az esetben szabályos a lépés, ha a szomszéd színe vagy figurája megegyezik a lerakandó kő színével vagy figurájával
3. ha kettő, akkor az egyik szomszéd színével, a másik szomszédnak pedig a figurájával kell megegyeznie a lerakandó kő színének vagy figurájának
4. ha három, akkor mindhárom szomszéddal szemben kell lenni vagy szín, vagy szimbólum egyezésnek, és kell lennie legalább egy egyezésnek színből és szimbólumból is
5. ha négy, akkor mind a négy szomszéddal szemben kell lenni vagy szín, vagy szimbólum egyezésnek, és kell lennie legalább két egyezésnek színből és szimbólumból is

Ha sikeres volt a lerakás, tehát a 2-5. esetek valamelyike teljesült, akkor növelni kell a játékos pontszámát. Egyszerűnek tűnik ez az öt feltétel, de assemblyben nem is olyan könnyű megfogalmazni. Először is nem minden mezőnek van négy szomszédja, tehát, ha a kiválasztott mező (X,Y), akkor lehetséges, hogy pl. (x-1,Y) a táblán kívül van. Aztán a pontozás szempontjából is szükség van a szomszédok számára, mert a szélső mezőkbe való lerakásért nem jár pont. Minden lépés után ki kell helyezni az új követ, de csak akkor ha az szabályosan elhelyezhető a táblán. Az ellenőrzéshez az előző 5 lépés használható, odapróbálva a következő követ minden üres mezőre. Az első esetben, mikor szabályosnak bizonyul a lépés, abba lehet hagyni az ellenőrzést, hiszen akkor már biztos van legalább egy mező, ahová el tudja helyezni a játékos a követ. Ha nincs szabályos lépésre lehetőség, akkor befejeződik a játék. Ha elfogytak a kövek, akkor szintén befejeződik a játék. Ezután az utolsó bekezdésben szereplő pár dolgot kell ismételgetni a játék végéig.

***És hirtelen kész is lett a játék***

Ami ezen kívül van, a zsák, tehát a TILESET maradék elemeinek kijelzése, a következő kő lehetséges helyeinek megmutatása segítségként, a menü, a betöltőkép, stb. már csak extra, csak hab a tortán. Hogy mi a tanulság? Az, hogy magának a játéknak az elkészítése, pláne egy logikai játéké, nem nagy durranás, sőt, meglepően egyszerű. Az idő nagy

részét az előkészületek, főleg a rajzolgatás, tervezgetés, kisebb részben pedig a megfelelő eljárások, függvények csiszolgatása veszik el. Menet közben pedig sokszor kell próbálgatni a félkész programot. Volt, hogy nem tettem és alig találtam meg egy-egy nagyon primitív hibát.
Gyakori hiba, amit könnyű benézni, mikor pl. egy érték 1-100 lehet, de a kódban egyszerűbb 0-99-ként alkalmazni (pl. ha egy címhez adom hozzá: az első érték, maga a cím, tehát nem cím+1, hanem cím+0). Ilyenkor, amíg nem lesz 100 adott érték, addig fel sem tűnik, hogy valami gixer van. Rövid próbálgatással, csak úgy felületesen, mikor pont nem generál oda 100-at a program, ahova nem kéne, elsikkad a hiba. Tovább halad a program fejlesztése, aztán a sokadik lépés utáni próbánál bejön az a bizonyos 100-as érték, a program elkezd hülyeségeket csinálni, a programozó vizsgálja az utoljára fejlesztett részeket, hiszen eddig jó volt minden, akkor biztos az új részek a rosszak. Mire megtalálja az egyszerű hibát, addig órák, napok telhetnek el.

A fejlesztés végére valami ilyen képet mutatott az Ishido:

```
   org   24200
   jp    MAIN

; ÁLTALÁNOS ELJÁRÁSOK, FÜGGVÉNYEK

KEYPRESS
   …
   ret

PRINTSTR
   …
   ret
   …

; ISHIDOHOZ TARTOZÓ FÜGGVÉNYEK, ELJÁRÁSOK

START
   …
   ret

CURSOROUT
   …
   ret
   …

; A PROGRAM FŐ RÉSZE

MAIN
   …
   ret

; MINDENFÉLE VÁLTOZÓK, AMIKRE SZÜKSÉG VAN A
; PROGRAMBAN

X       defb        5
Y       defb        5
SCORE   defb        0,0,0,0,0
   …
```

Végül a kezdőknek, vagy a kezdés előtt állóknak saját tapasztalataim alapján a következőket tanácsolom:

1. kezdj el ismerkedni az alapokkal
2. ehhez legyen egy olyan asztali környezeted, amit célszerűnek, kényelmesnek vélsz, én csak a Crimson Editor+PASMO+Spectaculator szentháromságot ismerem, nekem az bevált, de biztos vannak alternatívák
3. mindent, amit megismersz, próbálj is ki, ne foglalkozz kizárólag elméletekkel, de ne akarj mindjárt mindent tudni, amiről úgy érzed, hogy már megértetted, azt próbáld ki így is, amúgy is, változtass rajta kisebb-nagyobb dolgokat
4. kezdj el felépíteni magadnak egy könyvtárat olyan eljárásokból, függvényekből, amik általánosságban hasznosnak tűnnek, egy következő projektnél nagyon jól jöhetnek majd, és nem veszik el az értékes időd, nem kell újra meg újra megcsinálnod ugyanazt a rabszolgamunkát
5. gondold végig, hogy amit meg akarsz valósítani, az egyáltalán lehetséges-e Spectrumon, megvalósítható-e nyolc színnel, 256x192-es képernyőn, korlátozott memóriával, stb. Ha igen, akkor próbáld meg logikusan végiggondolni leendő programodat, rajzolj, készíts grafikai terveket, képernyőterveket, lásd magad előtt a végeredményt
6. próbáld meg logikai szakaszokra bontani a fejlesztést, nem kell minden egyes lépést előre látnod, csak nagy vonalakban, aztán boncolgathatod a nagyobb szakaszokat kisebbekre, mikor odaérsz a fejlesztésben, írj megjegyzéseket, hogy később is tudd, mit miért csináltál
7. ne görcsölj, ne akard előre látni az összes lehetséges problémát, menet közben úgyis módosítanod kell a programodon többször is. Amin előzőleg agyaltál, az lehet, hogy feleslegessé válik később
8. lépésenként haladj, és minden szakaszt próbálgass, ne sajnáld rá az időt

A végére egy idézet Pgyuritól, jobban úgysem tudnám megfogalmazni:
„A játék-programozás legszebb és sajnos legrövidebb része a programkód készítés, alig lesz 10 %-a a ráfordított időnek. Az összes többi a grafika, hang elkészítésére, a tesztelésre és hangolásra megy, ami viszont a legunalmasabb az egészben. Nem véletlen, hogy annyi hiba volt, van és lesz a játékokban…”


***Mezei Róbert (m/zx)***

# HARDVER SIMOGATÓ

## KLAVIA-TÚRA: ZX81 MEMOTECH BILLENTYŰZET

Kedvenc Sinclair gépeinknek az egyik hendikeppje bizonnyal a billentyűzet, ezért aztán szép számmal készültek a különféle megoldások. Ez a cikksorozat ezen a porondon próbál egy kis útmutatást nyújtani.
A Memotech billentyűzetének már most megelőlegezhető egy csomó, fokozó jelző, legyen szó akár a megjelenésről, akár a praktikumról, akár a belbecsről.

### Külcsín

„Ebben még van anyag!" mondhatnánk, lévén a Plug-in vázát, egy közepes atomrobbanást is minden bizonnyal túlélni képes alumínium ház adja. Jó nehéz is, de az csak jó, hiszen nem kell attól félni, hogy elszánkázna az ember keze alól. A megjelenés hűen követi az egyéb Memotech termékek - memóriabővítők, printer illesztők, Hi-Res grafikus interfa ce (!) -, hűvös-elegáns „dizájnját" és anyagát is, ugyanis azokat sem átallottak alumíniumból készíteni! A 41 billentyű elhelyezkedése hűen követi a ZX81 eredeti elrendezését, egyedül a felső sorban, a nulla mellett helyet foglaló „Shift" a kakukktojás, mely az egyetlen plusz az eredetihez képest. Nagyra értékelem, hogy a billentyűsapkákon a feliratok nyomtatottak és nem csak matricákat biggyesztettek rájuk.


### Belbecs

Töredelmesen be kell valljam, hogy nem volt lelkem szétszedni. Erre két jó okom volt. Egyrészt a fémházat „színrefújt" csavarok tartják egyben, és még ha oly óvatosan is bánik velük az ember, nyomot hagyna rajtuk a csavarhúzó. Lévén látszott, hogy az elmúlt több mint húsz évben senki sem szegezte nekik a kereszthornyos Philipset, felettébb nagy ostobaság lett volna csak pár fotó kedvéért „elvenni a szüzességét". Másrészt pont ez a Plug-in legnagyobb attrakciója! Nem kell szétszedni és beleszerelni a ZX81 alaplapját, hanem elegendő rádugni az alapgép buszcsatlakozójára a billentyűzetből kijövő szalagkábel végén fityegő interface-t! A fantasztikus állapotú billentyűzettel érkezett, éppen olyan döbbenetesen jó állapotú doboz és leírás joggal büszke erre a tulajdonságra. Ma már kevés jelentősége van, hogy „így nem vész el a garancia" és az is megmosolyogtató kicsit, hogy „mivel a gép eredeti billentyűzete is párhuzamosan működik a Plug-in-nel, ideális az olyan játékokhoz, melyben két ellenfél játszhat egymással". De tény ami tény, annál egyszerűbb, hogy „felbiggyesztem a gép hátuljára az interface-t és már használhatom is" nincs!

### Jó vétel?

Bár még nem sok ZX81 billentyűzetet láttam, azt hiszem kijelenthetjük, hogy a legjobb vétel ehhez a géphez! Nem bocsátkoznék ismétlésekbe a „miért" kifejtése okán.

### Summázás

Az értelmező szótárban az „igényes" mellé azt hiszem bőven elegendő lenne, egy Memotech Plug-in képét másolni. Zseniálisan egyszerű üzembe helyezni, a megjelenése fantasztikusan elegáns, örökéletű fém házban van. Mi kell ennél több?

***Samu József (Sam. Joe)***

## BASIC TUDOR

### ATTR

ATTR(X,Y)

az adott 8x8-as terület színkódját adja vissza.

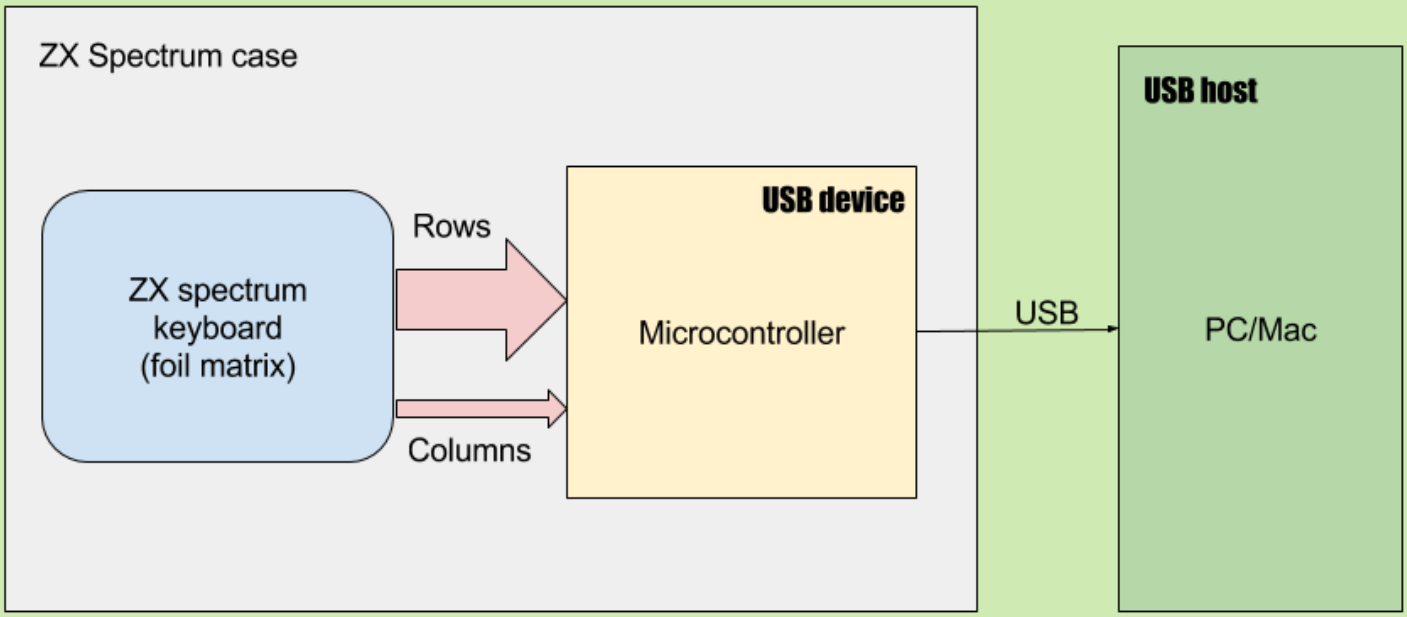
Koordináták: X: 0-21 és Y: 0-31
Eredmény:
FLASH*128+BRIGHT*64+INK*8+PAPER

Egy pixel színét koordinátáinak 8-cal osztásával kapod meg.

# HARDVER ÖTLETEK - ZX SPECTRUM

## USB BILLENTYŰZET ZX SPECTRUMBÓL - 1. RÉSZ

A projekt célja, hogy egy ZX Spectrum dobozából - amely valójában a billentyűzetet is tartalmazza - egy olyan eszközt készítsünk, amit a PC, mint egy **szabványos USB eszközt** felismerjen. Ez az eszköz egy **ZX Spectrum emulátor**ral együtt használva autentikus játékélményt nyújt.




A megvalósításhoz olyan mikrokontroller szükséges, amely képes USB Device módban működni, mintegy „emulálni" a normál PC billentyűzetet.

A billentyűzet kezelése mellett a mikrokontroller egyéb feladatok ellátására is képes lehet. Játékok esetében az első felmerülő igény, hogy hagyományos digitális (mikrokapcsolós) joystick csatlakoztatására is legyen lehetőség, illetve, hogy ez a joystick a billentyűzet bármely nyomógombját meg tudja valósítani (keyboard mapping), ezáltal univerzális irányítóeszközt kaphatunk. Az eszköz nagy előnye, hogy szabványos USB billentyűzetként viselkedik, ezért hardver és operációs rendszer független, bármilyen rendszerrel használható, ami képes az USB HID eszközök kezelésére (Windows, Mac, Linux, Android).

A megvalósításhoz a Microchip egyik 16 bites kontrollerét választottam, amely USB eszközként tud működni és megfelelő erőforrásokkal rendelkezik a feladat megoldásához. A PIC24FJXXXGB család bármelyik tagja megfelel, valójában csak a rendelkezésre álló FLASH és SRAM méretében különböznek egymástól. Ugyancsak szempont volt a választásnál, hogy az USB kezelés megfelelő szoftveres támogatással rendelkezzen, a fejlesztőkörnyezet legyen ingyenes és az eszköz ne csak a Spectrum, hanem adott esetben más klasszikus 8 bites gép billentyűzetét is képes legyen kezelni (pl. Enterprise, C64). Mindezekért a PIC24 család egy 64 kivezetéssel rendelkező tagját választottam, ezen elegendő kivezetés van egy 8x16-os mátrix és két független digitális joystick kezeléséhez. A megvalósításban használható eszközök tehát: **PIC24FG128GB106** vagy **PIC24FJ256GB106.**

A Microchip szoftveres környezete példa-programokkal és C fordítóval ingyenesen használható, az egyetlen eszköz, amire ezen kívül szükség van, az egy dedikált programozó, ezt egyszer kell megvásárolni. Megfelelő programozó a Pickit2 vagy Pickit3 eszköz. (A Pickit2 régebbi modell, ehhez a családhoz megfelel, de ha esetleg később más PIC mikrokontrollert is programozni akarunk, akkor a Pickit3 a jobb választás)

A hardver megtervezése a Microchip dokumentációi alapján viszonylag egyszerű. A billentyűzet „oldalán" egy egyszerű mátrix kialakítást kell megoldani, ahol a sorokat kapcsolgatva az oszlopok értékét le tudjuk olvasni, megfelelő gyakorisággal ismételve ezt a billentyűzet épp aktuális állapotát a mikrokontroller memóriájában tárolni tudjuk. A stabil működés érdekében az oszlopok kivezetéseire külső felhúzó ellenállást terveztem, így az inaktív 1-es állapotot

ezek biztosítják, míg az aktív lenyomott „0” akkor áll elő, ha a megfelelő soron a mikrokontroller 0-t ad ki, a billentyűt pedig megnyomják. A megoldás teljesen hasonló az eredeti Spectrumos megoldáshoz, annyi különbség van csupán, hogy a sorok tri-state állapotát (csak 0-t kapcsoljon vagy nagyimpedanciás Z-t, de 1-et soha) nem külső diódákkal, hanem az I/O vonalak bemenetbe kapcsolásával oldja meg a firmware.

Az USB oldalon a mikrokontroller „USB HID device”-ként viselkedik. Ennek megvalósításához a Microchip kész prototípusalkalmazásokat ad, így csak annyi volt a feladat, hogy az USB példaprogramot össze kellett illeszteni a mátrix lekérdező funkciókkal.

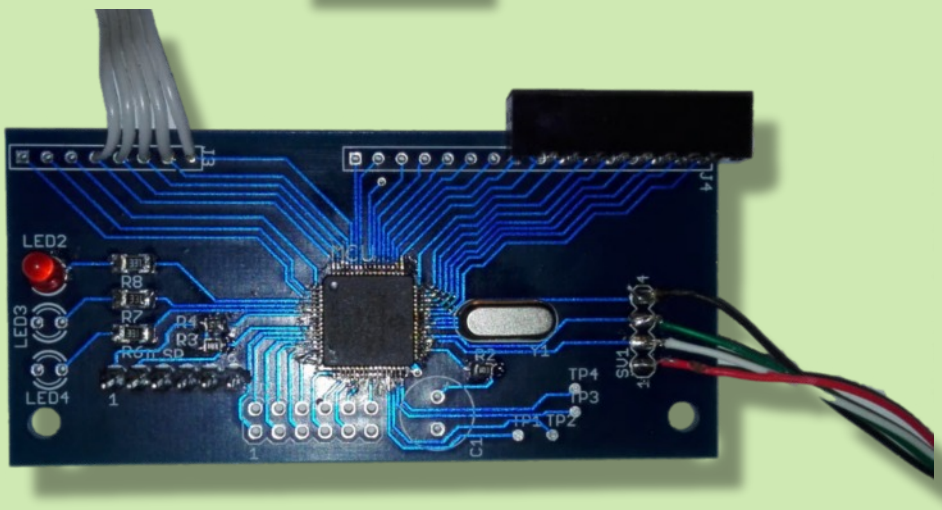
A mátrixról beolvasott billentyű-kombinációt bizonyos esetekben módosítani kell, ha pl. a ZX Spectrum billentyűzetet a PC-n hagyományos gépelésre szeretnénk használni. Pl. a Caps Shift+0 a Spectrumon a „Backspace”-t jelenti, vagy a Caps Shift+5 a kurzor balra, és így tovább. Ugyanakkor más esetben, ha az eszközt emulátorral használjuk, ez a funkció felesleges. További nehézséget okozhat, hogy az eltérő emulátorok a PC billentyűit másképp használhatják. Van, ahol a Ctrl gomb a Symbol Shift, de máshol meg az Alt, esetleg a két Shift is eltérő funkciókat lát el, vagy a Tab kapcsolja az E módot, és ehhez hasonló szituációk. Mindezekért az eszközt úgy kell kialakítani, hogy ezen változtatásokat kezelni tudja. A firmware ezért képes arra, hogy egy megadott formában beírt parancs alapján a működését megváltoztassa. Ezeket a parancsokat egyszerűen, pl. egy *Notepad* szerkesztőt megnyitva lehet beírni, a firmware felismeri és végrehajtja őket. A legutolsó beállításokat pedig minden esetben saját Flash memóriájában el is tárolja, tehát kikapcsolás után ugyanebben az állapotban lesz a billentyűzet.

**Parancsok:**

- **ssctrl** symbol shift a Ctrl billentyűre mappelve
- **ssalt** symbol shift az Alt billentyűre mappelve
- **keymapon** bekapcsolja a speciális kombinációkat
- **keymapoff** kikapcsolja a speciális kombinációkat
- **joy1map** az 1. Joy port mappelése a megadott billentyűkre
- **joy2map** a 2. Joy port mappelése a megadott billentyűkre

**A kész panel (prototípus) képe:**
Az eszköz kialakításának köszönhetően más 8 bites gép házába is beépíthető. A megfelelő firmware-rel lehetőség lenne pl. egy C64 vagy Enterprise házába beültetni, a szükséges „mappelés” után emulátorral használva nagyszerű „klasszikus” játékérzést nyújthat a 8 bites korszakot kedvelők népes táborának.

***László József (FPGAJoco)***
*Folytatjuk...*


## BASIC TUDOR

## USR

`LET X=USR <memória cím>`
Visszatéréskor az X változóba beteszi a BC regiszterpár értékét. Arra tervezték az USR függvényt, hogy assembler rutint (függvényt) hívjon meg és legyen visszatérő értékre lehetőség.

`RANDOMIZE USR <memória cím>`
Visszatéréskor BC-vel inicializálja a véletlenszámot.

`PRINT USR <memória cím>`
Kiírja a BC értékét.

Ugye RUN futtatja a BASIC programot az x. sortól.

`RUN USR <memória cím>`
Tréfás megoldás, ha visszatér, akkor amit visszaad, onnan RUN.

# Ne keresd az újságárusnál, inkább töltsd le!


www.fanzix.hu