SPECCYALISTA VILÁG

A SPECCYALISTA BARÁTI KÖR LAPJA

SINCLAIR.HU

Rick Disckinson interjú

A ZX Spectrum ULA története

ZX Spectrum ULA típusok

LOAD "1984"

Assembly ovi 8. rész

OlvaSOKKoló

vDrive^{ZX} ismertető

Klavia-túra

Játékújdonságok

2018. 2. SZÁM

Tartalom

Beköszöntő

Kedves olvasó!

Virtuálisan ismét kézbe vehetitek a várva várt Speccyalista Világ legújabb számát!
Ezentúl idén még számíthattok egy dupla számra is december végén.

Most is mint mindig, igyekeztünk az egykori Spectrum Világ szellemiségét megőrizni és minden olvasónknak adni valami számára érdekes olvasmányt.
A szerkesztőségünk címére ugyan most is kevés visszajelzés érkezett, de ezt a vonalat igyekszünk továbbra is képviselni.

A Sinclair.hu portálon már rengeteg hasznos tudást halmoztunk fel az elmúlt években, amelyekből leportalanítjuk az arra érdemes írásokat és időről-időre ezekből az anyagokból is beválogatunk cikkeket, melyek így talán még több olvasóhoz juthatnak el.

Szeretnénk továbbra is, betekintést engedni a berkeinken belül folyó projektekbe. Legyen az akár hardver- vagy szoftverfejlsztés, archiválás, régészkedés.
Most Szilágyi György és Náray József osztja meg ötleteit.

Nagy örömünkre szolgál, hogy ismét sikerült újabb szerzőket megnyerni az ügynek, így már Nagy Dániel remekbeszabott írásaira is számíthatunk, reméljük a speccyalisták táborát is erősíti hamarosan. Hiába, akiknek Z80 dobog a szíve helyén, azokra mindig számíthatsz!

Ismét ollózunk nemzetközi kapcsolatunktól. Nemrégiben hunyt el Rick Dickinson, így most a vele készített interjú magyar fordítását közöljük a ***Villordsutch*** név alatt publikáló brit szakírótól, aki egyszemélyben blogger, gamer, reviewer. A továbbiakban is számíthattok magyar fordításban megjelenő interjúira a brit sinclaires világ neves alakjaival, természetesen a szerző engedélyével.

Reméljük a lap továbbra is elnyeri tetszéseteket és lesznek, akik szívesen bekapcsolódnának 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 lapunk szerzőinek és munkatársainak azt a sok munkát, melyet befektettek ezen újabb szám elkészítésébe.

2018. október 31.

Kardos Balázs (Balee)

Impresszum

Főszerkesztő: Kardos Balázs (Balee)
Lektorálás: Sári Gábor (sAGA)
Szerkesztés, tördelés: Kardos Balázs (Balee)
Grafika: Molnár Péter (Mopi)

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

Szerzők: Barabás Péter (z0d), Buzogány Csaba (Makranc), Horváth Péter (Hpeter), Kardos Balázs (Balee), Kovács Péter (kpbendi), Nagy Dániel, Mezei Róbert (M/ZX), Szilágyi György (Pupos)

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

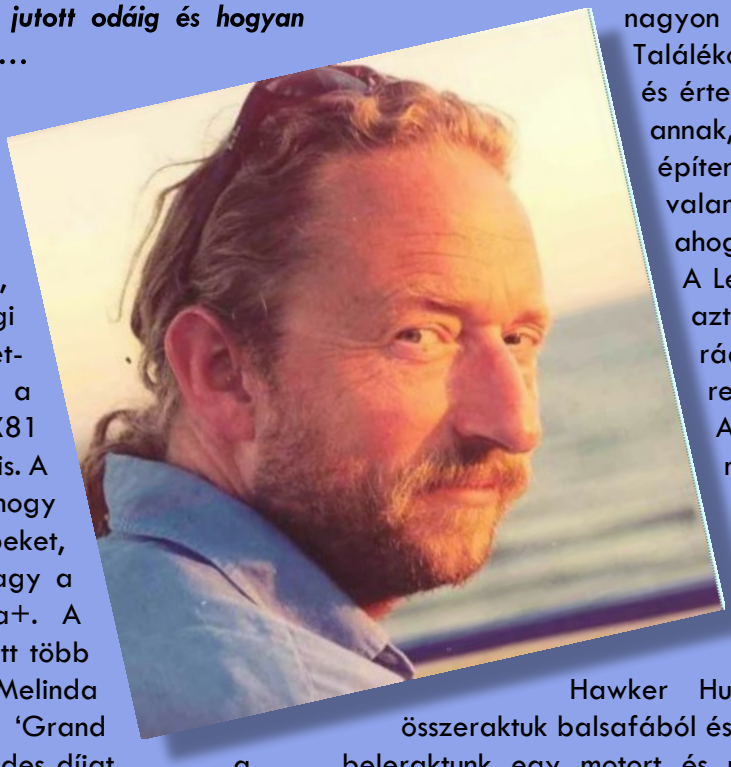
Kiadó: Speccyalista Baráti Kör http://sinclair.hu

Facebook: https://www.facebook.com/Speccyalista-Világ-1860907330696250/ 2018. október

ARCKÉPCSARNOK

INTERJÚ RICK DICKINSON-NAL

Az alábbi interjú a FlickeringMyth.com-on jelent meg 2016. augusztus 19-én, Villordsutch interjúja Rick Dickinsonnal: a pillanat, amikor a Sinclair Research Ltd.-nél kezdett dolgozni, hogyan jutott odáig és hogyan tervezte az új „retro" gépeket…



Rick Dickinson az egyik legismertebb brit tervező azok számára, akik 1990-es évek mikroszámítógép lázában nőttek fel illetve azoknak, akik részt vesznek a jelenlegi hatalmas retrogaming őrületben. Neki köszönhetjük a díjnyertes Sinclair ZX81 kinézetét és a ZX Spectrumot is. A mai napig megkeresik, hogy tervezzen modern „retro" gépeket, mint a ZX Spectrum Next vagy a Sinclair ZX Spectrum Vega+. A Dickinson Associates-el együtt több díjat is nyert, köztük a Bill&Melinda Gates Alapítvány 'Grand Challenges'-et és az Archimedes díjat a mérnöki kiválóságáért.

Rick Dickinson.

Villordsutch: Amikor felnőttél a személyi számítógépek és persze a Sinclair gépek, még nem léteztek az álláskiírásokban. Mivel foglalkoztál 1979 előtt és mi szerettél volna lenni, mielőtt átlépted a Sinclair Research küszöbét?

Rick Dickinson: Természetesen a számítógépek nem voltak elterjedve akkor, és bár tudtam, hogy léteznek, leginkább csak kifinomult misztikumként gondoltam rájuk, amik a MOD és a NASA jellegű cégeknél léteznek. Őrülten szerettem volna feltalálni és készíteni dolgokat. Legóval kezdődött és mivel a nagyszüleim Németországban éltek, ezért évekkel korábban hozzájutottam, mint ahogy az Egyesült Királyságban meg lehetett venni. Lego házakat csinálni jó volt, de építettem magamtól is és TV műsorokból is, mint a Thunderbirds (köszönet Herry és Silvia Andersonnak, hogy annyi fiatal elmét inspiráltak). Csináltam függőhidakat, űrhajókat, tengeralattjárókat, rakétákat, plafonig érő tornyokat, kastélyokat, James Bond autókat, amik anyám szárítókötelén autóztak, hajókat, amiket a kertünkben levő tóban úsztattam és így tovább. Hihetetlenül tanulságos termék volt, lefedve a kreatív folyamat sok aspektusát, azt, hogy hogyan kell megvalósítani egy terméket és mindezt nagyon kevés erőforrásból. Találékonynak kellett lenned és értened kellett a lényegét annak, amit meg akartál építeni, hogy mitől működik valami és miért néz ki úgy, ahogy.

A Lego után az Airfix jött, aztán felfedeztem a rádió-vezérlésű repülőket és csónakokat. Apám légtechnikai mérnök volt és akkoriban a Blackburn Bucaneeren és a TSR2-n dolgozott. A tervező táblájára rajzoltunk egy Hawker Hurricane-t és nulláról összeraktuk balsafából és zsebkendőkből, aztán beleraktunk egy motort és megreptettük. Ez egy szuper első lecke volt tervezési folyamatban, starttól a finisig. Mindig építőmérnök akartam lenni - utak, hidak, alagutak -, de szenvedtem a matekkal. Az utolsó pillanatban, amikor már éppen építész akartam lenni, felfedeztem az ipari tervezést és tudtam, hogy ez lesz a nekem való munka. Az ipari tervezésben benne van a villától és a késtől kezdve a targoncáig minden (ahogy a rajztanárom mondta). Minden tömegtermelésre tervezett dolognál szükség van ipari tervezőre - fogkefe, okostelefon, 13A-es csatlakozó, orvosi eszközök - bármi és majdnem minden. A Sinclair volt az első munkahelyem a diploma megszerzése után. Nem tudtam, hogy Clive Sinclair tervei között van a számítógép, simán csak az „a világon az első" dolog vonzott, pl. a zsebszámológép, digitális karóra, zseb TV stb.

Gerry es Silvia Anderson inspirációi.

Villordsutch: Emlékszel arra, hogy mit szóltak a barátaid és a családod, amikor megtudták, hogy a dicséretes első osztályú művészeti diplomádat arra használod, hogy számítógépeket tervezz, annak az embernek, aki digitális karórákat és elektronikus számológépet gyárt?

Rick Dickinson: Ahogy én is, ők is nagyon izgatottak voltak. Clive már akkor is elismert volt a tömegeknek

szánt innovatív termékeiről. Azonban senki sem (még én sem), tudta hogy mi az az otthoni számítógép.

Villordsutch: Bár a ZX81 volt az első számítógépem, úgy gondolom, hogy a ZX Spectrum volt a legjobban kinéző géped, utána pedig a ZX80 (egyébként mindhárom előttem van, miközben gépelek). 1981-ben megnyerted a British Design Council díjat a ZX81-gyel, majd a Sinclair QL-lel a Smau Industrial díjat. A klasszikus Sinclair munkáid közül melyiket mondanád a kedvencednek és miért?

Rick Dickinson: A Spectrum az, amivel a legtöbb ember szerelembe esik. Töröm a fejem, hogy megértsem az összefüggést az esztétika és a termék iránti különböző szintű vágy - illetve később a végső vonzódás - között. Nem tudom távolról és objektíven nézni a munkámat és biztos más preferencia alapján választok, de nekem a ZX81 az. Miért? Nem tudom. A kreatív tevékenység sok-sok kávé, csoki… és néha puszta kín egyvelege, főleg amikor nincs másik termék, amihez viszonyíts. Talán azért, mert ez volt az első és teljesnek éreztem - nem tudtam tovább fejleszteni és úgy gondoltam, hogy abban az állapotában már elég jó lesz. Ehhez képest a Spectrummal soha nem jutottam el ilyen szintre és nemrég, ahogy néhányan láthattak is, újragondoltam az eredeti tervet. A QL jobb is lehetett volna, bár nekem tetszett és tartalmazott néhány bátor újítást az ipari tervezésben, volt néhány alternatívám a végső változatot illetően, de már nem volt idő kidolgozni.

A díjnyertes ZX81.

Villordsutch: Te tervezted a Sinclair ZX Spectrum Vega+-t és a ZX Spectrum Next-et. Milyen érzés tudni, hogy a „Rick Dickinson tervezte" szlogennek ilyen súlya van a rajongók között.

Rick Dickinson: Nos, kicsit zavarba ejtő, mert nem vagyok kimondottan egocentrikus és általában elkerülöm a nyilvánosságot, csendben teszem a dolgom. Persze azért jólesik, hogy szép emlékeket vittem az emberek életébe, csak úgy, mint Anderson-ék az enyémbe.

***Villordsutch*: Hogyan kezdted el kidolgozni a Vega+ és a ZX Spectrum Next terveit? A készítőknek már volt elképzelésük, amit egy zsebkendőn odavittek neked, hogy dolgozd ki, vagy hagytak, hogy egy kényelmes szobában, sok teával, szendviccsel és egy rakás papírral és ceruzával dolgozz?**

Rick Dickinson: A készítőknek nem voltak vázlataik, de tudtak, hogy mit szeretnének. A gyakorlat azt mutatja, hogy sokszor a tervezők nincsenek tudatában fontos információknak és emiatt ki vannak téve a mélyebb szintű változtatásoknak a termékben. Ezért abszolút biztosnak kellett lennem abban, hogy tudom mi jár a fejükben, miért választottak egy adott technológiát illetve gyártási módot és nem utolsó sorban ott van a marketing faktor. Például mit jelent az, hogy valami „retro" és miért akarna bárki újracsinálni valamit, amit már egyszer megcsináltak? Jogos kérdés, és mint a legtöbb terméknél, fontos, hogy el tudd adni. Így először a kérdéseket kellett kitalálni, mielőtt hozzákezdhettem volna a tervezéshez. Tudnom kellett, hogy egy olyan terméket tervezek-e újra a jelen kornak, ami látványban hasonlít az eredeti Spectrumra, mintha a Sinclair Research még mindig létezne és úgy döntött volna, hogy csinál egy hordozható Spectrumot. És ha ez így van, mi a referencia, mivel a régi Spectrum valami mássá változott, mint ami volt. A másik lehetőség, hogy olyat tervezek, amit az eredeti Spectrum idejében terveztek volna. Mindkét lehetőséget megvitattuk; a Vega Plus esetében a második verziót választottuk, viszont a Spectrum Next-nél pont az ellenkezője volt - megtervezni úgy, mintha a Sinclair még létezne.

A Vega Plusnál megbeszéltük a szükséges gombok számát és logikus elrendezését, az akksi kapacitást, a képernyő méretet, a NYÁK csatlakozóit stb., aztán elkezdtem a tervezést. Elképzeltem, hogy a Kings Parade irodában vagyok a 80-as évek elején, és hogy hogyan terveztem volna. A legnehezebb rész a megfelelő forma kitalálása volt. Az eredeti Spectrum egy asztali számítógép volt, a Vega Plus pedig egy hordozható játékkonzol - nagyon eltérő, ahogyan használják a kettőt, ezért ergonómiailag is

különböznek és emiatt végül kinézetben is. Hogyan lehetne egy hordozható eszköz olyan, mint egy Spectrum? Izgalmas volt elővenni a Spectrumot, a saját munkámat, és kiszedni belőle a kinézete lényegét és alkalmazni valami egészen másra, remélve, hogy jó lesz. Igazából nagyon élveztem a folyamatot és jó volt a csapattal dolgozni, ahogy végigmentünk sok formán, aztán kiválasztottunk egyet és továbbfejlesztettük a vezérlő felületet, majd több verziót is kipróbáltunk mielőtt döntöttünk a végleges mellett. A Spectrum Next nagyon más volt, sok tekintetben egy teljesen más termék, sok lehetséges kimenetellel, de hasonlóan gondolkodtunk. Mindkét termék tervezési folyamatának újratekintése érdekes lenne.

Villordsutch: A polcos számítógépek elég egyformák lettek az elmúlt évtizedekben és csak a konzolok néznek ki kicsit máshogy. Ez tetszik neked vagy még nem láttál olyat, amire elismerően bólogatnál?

Rick Dickinson: A legtöbb termék elég unalmas és nem inspiráló, de időről időre kijön valami nagyon izgalmas, ami megváltoztatja a játékszabályokat és előremozdítja a design evolúciót abban a kategóriában. A konzoloknál ez a Sony PSP volt. A 80-as években az Olivettit, az Apple-t, a Grid-et és az Apricot-ot csodáltam az asztali/hordozható termékeik miatt, a hagyományos elektronikus fogyasztói cikkek között pedig a Braun-t. Manapság, a hihetetlen mennyiségű termék, fantasztikus tervezések és ipari gyártástechnológia ellenére a legtöbb termék - legalábbis ipari tervezési szempontból - elég kiábrándító. Az Apple az igazi úttörő és nagyon magasra rakták a mércét nagyon gyorsan és nagyon következetesen. Mások nagyon közelről követik az ötleteiket és megoldásaikat, néha egyenesen lemásolják. Elég kiábrándító, hogy nem tudják kitalálni a saját stílusukat. Azt gondolom, hogy az emberek felismerik a „jó designt", ha látják.

A Sony PSP

Villordsutch: Az oldaladat (The Product Designers) nézve az eszközök, amiket terveztél mind fantasztikusan és futurisztikusan néznek ki. Először azt hittem, hogy a vezetéknélküli magzatfigyelő csak egy koncepció, de aztán a linkre kattintva ott volt a gyakorlatban is. A díjaid közt szerepel a Bill&Melinda Gates Alapítvány „Grand Challenges" és az Archimedes díj a mérnöki kiválóságért, ami bevallom nagyon impresszív. Zavar, hogy néhány ember csak úgy ismer, mint aki a Speccy-t tervezte vagy inkább büszke vagy rá?

Rick Dickinson: Nagy megtiszteltetés, hogy közöm volt a Spectrumhoz és annak ellenére, hogy a többi termék, amin dolgoztam többé-kevésbé hasznosabb bizonyos szempontból, nagyon örülök, hogy én csináltam meg a Spectrumot.

A Flickering Myth és Villordsutch szeretné megköszönni Rick Disckinsonnak, hogy időt szakított az interjúra. Rick jelenlegi projektjei megtekinthetőek a weboldalán: http://www.theproductdesigners.com/

@Villordsutch

Fordította: Barabás Péter (z0d)
A szerző engedélyével közöltük a magyar fordítást.

Villordsutch YouTube csatornája, ahol szerzőtől további érdekes tartalmakat találhattok:

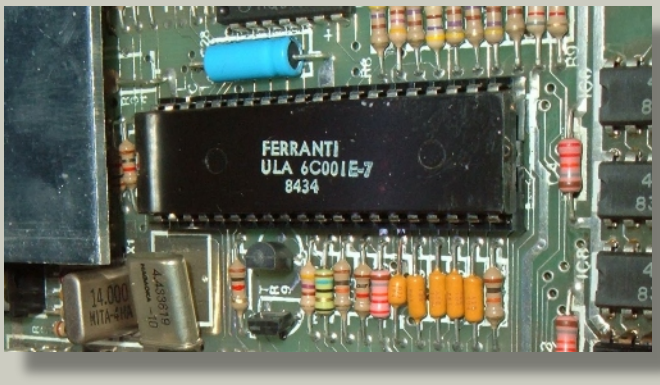
Amennyiben az érdeklődéseteket felkeltette a szerző, akkor ne felejtsétek el követni őt a Facebook-on:

SPECTRUMOLÓGIA

A ZX Spectrum ULA története 1982-től napjainkig

Bevezető

A Sinclair gépek sikerének egyik titka a ZX81 óta minden eredeti Sinclair gépben (illetve a szürke ZX Spectrum +2-ben) megtalálható ULA áramkör. A hárombetűs rövidítés az „összekötetlen logikai hálózat” rövidítése, ami jól megragadja a lényeget: a tömeggyártott félvezető legutolsó, kizárólag összeköttetéseket tartalmazó rétegét a megrendelő határozza meg, a félvezetőben lévő logikai áramkörök változatlanok. Emiatt a Sinclair gépeknek nem volt szüksége drága egyedi chipekre, mint a C64-ben lévő VIC-II vagy az Enterprise-ban lévő Nick és Dave, hanem elég volt a tömeggyártott ULA összeköttetéseit egyedi módon konfigurálni és ilyen módon lehetett bőségesen alákínálni az egyedi chipeket használó konkurenciának.



Az ULA technológia már az 1980-as évek végére elavulttá vált, így az Amstrad a ZX Spectrum +2A, +2B és +3 gépeiben már más kapumátrixszal helyettesítette azt, és csak szerviz célokra tartott raktáron ULA-kat a régebbi modellekhez. Napjainkban ULA-t már senki nem gyárt, ezért pótlása, amellett, hogy az egyik legsérülékenyebb chip a gépben, csak más, az ULA-t funkcionálisan helyettesítő áramkörrel lehetséges. Azonban éppen a modern áramkörök rugalmassága miatt a sima helyettesítés mellett képességjavító fejlesztések is történtek. Ezeket a cikkben részletesen bemutatom.

Hogyan rajzol az ULA?

A Spectrumokban az ULA legfőbb feladata a videokép generálása a videó RAM-ból, amit elég bizarr módon old meg (bár nem annyira bizarr, mint a ZX81). Közös 14MHz-es órajelből állítja elő a 7MHz-es pixelórajelet (sima felezéssel) és a 3,5MHz-es processzor órajelet oly módon, hogy ha a processzor olyankor akarna a video RAM-hoz hozzáférni, amikor az ULA, akkor egy-egy ütemre felfüggeszti a negyedelő órajelét, ami előállítja az órajelet a Z80 számára. Videó RAM-nak számít a 16k-s gépek teljes memóriája, a 48k-s gépek memóriájának a harmada (a $4000..$7FFF címtartomány), illetve a 128k-s gépek memóriájának fele (a páratlan számú memórialapok).

Az ULA általában a saját 3-bites tárolójában tárolt keretszínt (BORDER) küldi a videó áramkörnek, de amikor eljön a képrajzolás ideje, akkor a következőt csinálja: 8 pixelenként beolvas 2 bájtot a videó RAM-ból, amiből az egyik bájt bitjei egy nyolcbites tárolón végiggörgetve a pixelórajellel egymás után kiválasztják, hogy a másik bájt 0-2 vagy 3-5 bitjei kerüljenek kijelzésre, attól függően, hogy az adott bit 1 vagy 0.

A 6. bit minden esetben megnöveli a szín fényességét, a 7. bit pedig kapuz egy 16/50 másodpercenként alternáló bitet, ami megfordítja a kiválasztó bitet a másik bájtból.

A kiválasztó bájt minden vízszintes 8-pixeles csoport számára azonos, azonban a másik, ún. attribútum bájt 8 egymás alatt elhelyezkedő csoporthoz azonos, így 64db pixel színeire van hatással. Minden 8×8 pixeles karakterpozícióban két színnel lehet rajzolni (amiket önkényesen papír- és tinta színeknek neveznek).

Vegyük észre, hogy az attribútum bájtot az ULA minden egyes 8-pixeles csoport megjelenítésekor újra kiolvassa, ezért ha azt megváltoztatjuk olyankor, amikor némelyik hozzá tartozó csoport már megjelent, némelyik még nem, akkor a később megjelenítendőek már az új értéknek megfelelően kerülnek a képernyőre. Ez, valamint a tény, hogy az ULA és a processzor órajele közös forrásból származik, lehetőséget nyújt arra, hogy a program pontos időzítésével a karakterpozíciók 8 sorának akár 8 különböző attribútumértéket adjunk. Ez sajnos eléggé lefoglalja a processzort, ezért viszonylag ritkán látjuk ezt a lehetőséget kihasználva. Néhány demo, valamint a Nirvana és Bifrost keretrendszereket használó játékok „multicolor” megjelenítésében érhetjük tetten.

A ZX Spectrum ULA fejlődése

1984-ig az összes ULA változat az előző változatok hibáinak javításán túl semmilyen más változtatást nem hozott. Élesebb kép, megbízhatóbb betöltés a magnóról, valamint a hibákat utólag korrigáló áramkörkiegészítők elmaradása a fejlődés látható jelei.

1984-ben megjelent a Timex Computer 2048 nevű, ZX Spectrum kompatibilis gép, amelyet leginkább Portugáliában és Lengyelországban forgalmaztak.
Egy javított, ám gyártásba sohasem került ZX Spectrum prototípus (Timex Sinclair 2048) alapján készült, lényegében minden ZX Spectrumra írt szoftver változtatás nélkül vagy apró változtatásokkal futtatható rajta. Jelentősége abban van, hogy tartalmaz egy pár bővítést az ULA, amiket a manapság készülő ZX Spectrum ULA helyettesítőkbe is bele szoktak rakni. Érdekessége, hogy lényegében csak az ULA chipben különbözik a 48k-s ZX Spectrumtól.

A bővítések a következőket használták ki:

1. Az ULA két bájtot (azaz 16 bitet) olvas minden 8-pixeles csoport megjelenítéséhez.
2. Az ULA órajele a pixelfrekvencia duplája.
3. A videó RAM több, mint duplája a kép megjelenítéséhez ténylegesen használt tárhelynek.

Így a környező elektronika minden módosítása nélkül a következő új videómódokat lehetett bevezetni:
1. A kép átcímzése $4000-es címről $6000-es címre. Ez lehetőséget adott arra, hogy amíg a program az egyik képet rajzolja, addig a másikat mutassa, s így a képrajzolásból adódó idegesítő villódzás megszűnjön, a képváltás pillanatszerűen történhessen.
2. Az egyik bájtot $4000-től, a másikat $6000-től kezdődő memóriablokkból olvasva minden 8-pixeles csoportnak külön attribútuma lehet, így sokkal kevesebb megkötés van a képen a színek használatára, bár még mindig nem színezhetünk bármilyen pixelt az elérhető 15 szín bármelyikére.
3. A fenti módon kiolvasott két bájtot, dupla sebességgel közvetlenül kizavarva a videó áramkörre monokróm, 512 pixel széles képet kapunk, amely lehetővé teszi teljesen olvashatóan 85 karakteres sorok megjelenítését (karakterenként 6 pixellel), s így a 80-karakteres kijelzést elváró CP/M üzleti szoftverek portolását.

Sinclair azonban - utólag úgy gondolom, helyesen - úgy döntött, hogy ezt a javított ZX Spectrum változatot nem forgalmazza, illetve a Timex általi forgalmazáshoz nem adja a nevét.

Ehelyett 1985-ben kijött a ZX Spectrum 128, amelyben az ULA-ra újabb feladatok hárultak, úgy mint a Z80 64 kilobájtos címtartományát meghaladó 128k RAM és 32k ROM memória kezelése (lapozása) és a hanggenerátor chip címdekódolása. Így a fenti bővítések már „nem fértek bele”, kivéve a két kép váltogatásának lehetőségét, amely azonban a TC2048-cal nem kompatibilis módon, az 5. és a 7. memórialapok elejére helyezte a két képet. Ezek közül az előbbi fixen a $4000..$7FFF címekre van lapozva, biztosítva az eredeti ZX Spectrummal való kompatibilitást, de mindkettő belapozható a $C000...$FFFF címtartományba, így ugyanazokkal a rutinokkal lehet mindkettőbe rajzolni, illetve az éppen mutatott képet teljes egészében ki lehet lapozni a Z80 által látott memóriából, hogy feleslegesen ne foglalja azt. Ez mindenképpen jobb megoldás, mint a TC2048-asé, bár sajnos elég kevés szoftver használta ki. Külön feketepont Sinclairnek, hogy maga a 128k ROM (ami egyébként is össze lett csapva) sem használja. Érdemes azonban megjegyezni, hogy a TC2048-ra egyetlen kereskedelmi forgalomba került szoftver sem használta a két kép puffer lehetőségét, míg a ZX Spectrum 128-ra azért írtak egy pár játékot, ami használja.

Elbán innen, ULA-n túl

Kelet-Európa COCOM-lista által sújtott országaiban olcsósága, egyszerűsége és nagy szoftverválasztéka miatt viszonylag hamar felfigyeltek a ZX Spectrumra, amelyben a többi nyugati géppel ellentétben csak egy, a keleti blokkban nem gyártott alkatrész volt: az ULA.

A másik „soklábú”, nagy bonyolultságú alkatrész, a Z80A processzor pótolható volt a Kelet-Németországban gyártott U880 processzorral, amely a Z80 dokumentációja és visszafejtése alapján készült, így minden dokumentált és majdnem minden nem dokumentált tulajdonságában azonosan viselkedett az eredetivel. Az U880-t kicsit később lemásolták és sorozatban gyártották a Szovjetunióban is. Nem ismert olyan ZX Spectrumra írt szoftver, amely ne működne tökéletesen U880 processzoron.

A Sinclair ULA ugyan nem volt COCOM-listás, de nyugati importja a pénzügyi csőd szélén táncoló szocialista országokba még így is komoly probléma volt, így mindenképpen érdemes volt más megoldás után nézni.

A legjobb megoldásnak a legfapadosabb bizonyult: egyszerűen nem használni semmilyen nagyintegráltságú áramkört, hanem szabványos diszkrét logikai (ún. TTL) áramkörökből összehozni az ULA funkcionalitását. Többek között ezt az utat választotta a magyar HT-3080C nevű (sikertelen) iskolaszámítógép-jelölt is, valamint sok nagyszériában gyártott szovjet és csehszlovák klón.

A korai klónok, mint pl. a HT-3080C számára még nem állt rendelkezésre az ULA részletes visszafejtése, legfőképp a bizarr, órajelet felfüggesztő logika működése. Emiatt azt a problémát, hogy ugyanahhoz a memóriához a processzornak és a videoáramkörnek is hozzá kell férnie, úgy kerülték meg, hogy megkettőzték a video RAM képrajzoláshoz használt részét, és a processzor mindkettőbe írt, de csak az egyikből olvasott, míg a videó áramkör a másikból. Ez megspórolt ugyan pár logikai áramkört, de cserébe az akkor még jóval drágább RAM-okból kellett több. Később alaposabban sikerült visszafejteni az ULA-t és TTL áramkörökből tökéletesen rekonstruálni a működését. A volt Szovjetunió területén még az 1990-es években is gyártottak sorozatban ZX Spectrum klónokat ilyen módon.
A sors különös fintora, hogy ezek a kelet-európai szükségmegoldások később igen fontosnak és hasznosnak bizonyultak nyugaton is a ZX Spectrum közösség életbentartásához.

Az ULA utáni idők

Miután a QL (16-bites üzleti számítógép) és a C5 (villanyautó) kudarcai után Sir Clive Sinclair eladta cégét a konkurens Amstrad-nak, az még rendelt néhány szállítmány ULA-t a már eladott Sinclair gépek szervizeléséhez. Az első, ZX Spectrum +2 néven gyártott számítógéptípusba, amely lényegében nem különbözött a ZX Spectrum 128-tól a beépített magnót leszámítva, még került is ULA, de a technológia felett az 1980-as évek második felére végleg eljárt az idő.

Az utolsó ZX Spectrum márkanév alatt gyártott Amstrad gépek, az egymástól csak a beépített háttértárolóban különböző ZX Spectrum +2a, +2b és +3 (az első kettő magnóval, az utóbbi 3 hüvelykes floppymeghajtóval épült egybe) típusokban már nincs ULA. Ennek szerepét egy speciális Amstrad kapumátrix tölti be, amely azonban műszaki vakvágánynak bizonyult. Ezek a gépek koruk színvonalán is elavultnak számítottak, keveset adtak el belőlük, s ezért szinte semmilyen szoftver nem használta ki plusz képességeiket, így a későbbi klónok és emulátorok számára sem fontos a velük való teljes kompatibilitás, míg ők maguk csak részlegesen kompatibilisek a korábbi ZX Spectrumokkal.

Érdekességük, hogy ezekben video RAM-nak a 4, 5, 6 és 7-es lapok számítanak (ami nem okoz különösebb gondot, hiszen az 5. és a 7. lapok, ahol tényleges videótartalom van, ebben a halmazban is benne vannak), illetve hogy a bizarr órajelfelfüggesztés helyett a „tankönyvi” megoldást választva a Z80 WAIT jelét használják a processzor várakoztatására. Ez egy kicsit több processzoridőt vesz el, viszont mivel precízebb a tényleges versenyhelyzet logikája, ritkábban folyamodik hozzá, mint a régebbi típusok ULA-ja.
További tulajdonságuk (amiért picit irigyeltem őket sima +2 tulajdonosként), hogy képesek a $0000...$3FFF címtartományba is RAM-ot lapozni, így a Z80 teljes címtartománya lefedhető RAM-mal, ami megnyitja az utat a CP/M és más Z80 alapú rendszerek szoftvereinek komoly változtatás nélküli futtatásához.

ULA-t 1988 óta nem gyárt senki. Az Amstrad készleteinek kiárusítása után, ha elromlott egy ZX Spectrumban az ULA, akkor azt igen nehezen lehetett pótolni, leginkább egy másik gépből kannibalizálva. Ez a ZX Spectrumot hobbiként használó közösségnek négy lehetőséget hagy:

1. Lemondani a hardverezésről és csak emulátorban folytatni a platform használatát.
2. A kelet-európai klónokhoz hasonlóan TTL áramkörökkel helyettesíteni az ULA-t és így készíteni utángyártott gépeket, amelyek ZX Spectrumként működnek és nem csak szoftver, de hardver szinten is kompatibilisek a ZX Spectrummal.
3. IC foglalatba helyezhető, modern alkatrészekből készült áramkör használata, amit be lehet helyezni az ULA helyére az eredeti ZX Spectrum nyomtatott áramkörökbe.
4. Modern alkatrészbázisból újraépíteni a ZX Spectrumot.

Az utóbbi két megoldás számára természetes választás a korszerű FPGA áramkörök használata, melyek az ULA-nál is rugalmasabban konfigurálható logikai áramkörök. Teljes konfigurációjuk elektronikusan újraírható Flash memóriában van, illetve nem csak a logikai elemek összeköttetései, de maguk a logikai elemek egy része is konfigurálható. Ezen kívül még a legkisebbekbe is sokkal-sokkal több logika fér, mint a legnagyobb ULA-kba.

Konkrétan az egész ZX Spectrum, processzorostul, RAM-ostul, ROM-ostul, ULA-stul, minden népszerű változatában és alváltozatában belefér egyetlen FPGA-ba. Viccként akár egy komplett C64 is elfér mellette. De persze ha csak az ULA kiváltása a cél (ld. 3. pont), akkor elég az ULA-t belerakni, de a megmaradt FPGA kapacitást bűn lenne nem kihasználni valami hasznosra.

Még a második megoldást választók, illetve az eredeti, működőképes ULA-val ellátott ZX Spectrumok XXI. századi felhasználói is szembekerülnek egy egyre égetőbb problémával: a modern monitorokon és TV-készülékeken egyre nehezebb megjeleníteni a ZX Spectrum képét, pláne élvezhető minőségben. Ezért piaca lett a ZX Spectrumhoz készített HDMI illesztőknek, amik a ZX Spectrum élcsatlakozójára illeszkedve olvassák az ULA (vagy ami helyette van) kommunikációját és ebből HDMI képet állítanak elő. Tipikusan ezt is FPGA-val szokták megoldani és itt is felmerül, hogy mi hasznosat lehetne még belerakni abba az FPGA-ba?

Az ULAplus szabvány

Bárki, aki a ZX Spectrumot tovább akarta fejleszteni (Sinclairt is beleértve) már az 1980-as években szembetalálkozott azzal a problémával, hogy azokat a lehetőségeket, amik csak az ő termékén érhetők el, a szoftverfejlesztők nem fogják kihasználni, hiszen ezzel a saját szoftverük piacát szűkítenék.

Így maradtak javarészt kihasználatlanul az Enteprise és SAM Coupé gépek impresszív képességei, miközben e gépek árát jelentősen megnövelték, siker helyett piaci kudarcot okozva. Sinclair döntését a 2048-as prototípus gyártásba engedése ellen is ez igazolta, s a TC2048 ennek megfelelően meg is bukott. Még a ZX Spectrum 128 plusz memóriáját és kettős képernyőjét sem igazán használták ki a játékfejlesztők. Amire viszont ugrottak, az az AY-3-8912-es hanggenerátor, hiszen attól, hogy azt használják, legfeljebb annyi történik, hogy a régebbi Spectrumokon néma marad a játék, viszont a 128-as tulajdonosok szívesen megveszik a gépük képességeit megcsillogtató új kiadásokat. Ebből a szempontból érdemes megfigyelni az egyik legsikeresebb játék, a „Commando” megoldását: detektálta, ha Spectrum 128-asba próbálták betölteni és ebben az esetben a plusz memóriába a zene, illetve az azt lejátszó kód került. Így a végsőkig ki lehetett használni a 48k-s Spectrum memóriáját és nem foglalta benne hasztalan ballasztként a helyet a 128-as zene. De akinek 128-asa volt, az élvezhette ennek előnyeit.

A ZX Spectrumot hobbiként napjainkban tovább használó és fejlesztő közösség ezekből a történetekből tanult. Ezért a különböző fejlesztéseket igyekeznek úgy csinálni, hogy azok egyrészt egymással kompatibilisek legyenek, másrészt pedig hogy ne vegyék el a kedvét az amúgy is kisszámú programozónak attól, hogy használják ezeket azzal, hogy ezáltal az amúgy is szűkös felhasználói bázisukat tovább csökkentik. Ezeknek a célkitűzéseknek a megvalósulása az ULAplus szabvány.

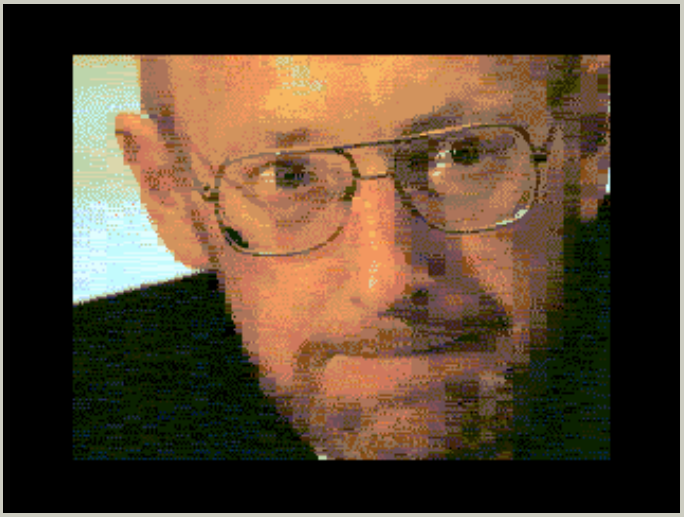
A szabvány legfontosabb eleme a 64-színű paletta. Ennek segítségével a Spectrum eredeti 15 színénél jóval többre van lehetőség. Mind emulátorok, mind ULA-helyettesítő áramkörök, mind Spectrumok élcsatlakozóira húzható HDMI illesztők támogathatják és támogatják is ezt a lehetőséget. A dolog lényege, hogy az attribútumbájt felső két bitje (eredetileg BRIGHT és FLASH) kiválasztja a 4 paletta közül az egyiket, míg a másik 3-3 bit az adott palettából a megfelelő előtér- és háttérszínt definiálja, méghozzá egymástól függetlenül, ugyanis egy paletta 16 színből (8 előtér és 8 háttér) áll.
Így minimális munkával (akár egy palettát beállító BASIC elő-betöltővel) újraszínezhetünk már meglévő programokat (ld. a Commando átszínezett változatát), de akár egyszerre 64 színt is varázsolhatunk a képernyőre a megfelelő attribútumbeállításokkal, amihez már egy kicsit bele kell túrni a kódba, illetve eleve így kell megírni. Ha a FLASH-t nem használjuk (ezzel 32-re csökkentve az elérhető színek számát, ami még mindig jó sok), akkor a színeket ügyesen megválasztva el tudjuk érni, hogy ULAplus-t nem támogató Spectrumokon is élvezhető a program, ám sokkal szebb ULAplus-szal.

A Commando játék akkor készült, amikor az ULAplus még ötlet szintjén sem létezett. Ennek ellenére minimális munkával sokkal „katonásabb” lett.

Ugyan opcionális része a szabványnak, de detektálni is lehet az ULAplus meglétét, és akkor a program el tudja dönteni, hogy használja a teljes 64-színű palettát, vagy sem.

A Nixy the Glade Spirit c. játékot már két változatban készítették el, eredeti ULA-ra és ULAplusra.

A következő, szintén opcionális része a szabványnak a TC2048-as videó módok támogatása. Erre az eredeti ULA-val és annak TTL helyettesítéseivel működő gépeket már nehezebb rávenni, ezért a jelenleg kapható HDMI illesztők ezt egyelőre nem támogatják, de a keleti klónoknál említett második video RAM trükkel még ez is megoldható.



Ha bekapcsoljuk a TC2048 multicolor módot és palettát is használunk mellé, akkor már látványosan túlmutat az eredeti ULA képességein az ULAplus. A képen Sir Clive Sinclair látható.

Hiszen nem kell a képet feltétlenül az eredeti video RAM-ból olvastatni az ULA-val, elég ha az illesztő a saját RAM-jába is elteszi, amit a processzor a video RAM-ba ír. És az 1980-as évek kelet-európájával ellentétben ma már az a plusz RAM nem hogy nem drága, hanem további plusz költség nélkül már ott van benne az FPGA-ban, ezért egyszerű firmware frissítéssel támogathatóvá lehet tenni a HDMI illesztők által ezeket a videó módokat. Ráadásul a minden 8-pixeles blokkhoz a video RAM-ból 2 bájtot kiolvasó szabály betartása miatt még időzítésbeli inkompatibilitás sem lesz a kétféle megoldás között.
Ezzel már majdnem eljutott a Spectrum az Enterprise vagy a SAM Coupé színvonalára, de sokkal-sokkal fejlesztőbarátabb módon. Ha így csinálták volna az 1980-as években, aminek igazándiból semmilyen technikai akadálya nem volt, akkor a bukás helyett nagy siker lehetett volna a jobb grafikájú Spectrum-verő gép.

Hogyan tovább?

A ZX Spectrum grafikájának legfőbb rákfenéjét, a színütközést (britül colour clash néven emlegetik) ezek egyike sem küszöböli ki. Arról, hogy ennek mi lenne a legmegfelelőbb módja, még folyik a vita és több különböző megoldási javaslat van forgalomban, ezek egyikét-másikát már implementálták is FPGA-ban és emulátor szoftverben, de ezen a téren még nincs konszenzus.
Szerencsére az FPGA-k utólag is konfigurálhatóak, ezért az ilyen döntéseket utólag is lehet módosítani.
Egyelőre a legelterjedtebb megoldás az ún. Radastan mód, aminek a felbontása mindkét irányban fele a ZX Spectrum felbontásának (azaz 128×96 pixel), egy pixel az első paletta 16 színének bármelyikét felveheti. Minden bájt 2 pixel színét határozza meg, a videó puffer mérete 6144 bájt, azaz még picit kisebb is, mint a Spectrum eredeti kép puffere. Implementálja egy pár emulátor, valamint egy pár FPGA alapú gép, de vannak vele problémák. Az egyik, hogy nem tölti ki teljesen a Spectrum által biztosított kereteket, azaz csak feleannyit olvas a RAM-ból, 12,5 %-kal kevesebb helyről. A másik, hogy egy pixel színének a módosítása vagy egy sprite kirakása szoftveresen meglehetősen lassú, mert bonyolult maszkolást igényel. Emiatt a Spectrumtól megszokott sebességű szoftvereket írni rá nehézkes. Ezért egyelőre a Radastan nem része az ULAplus szabványnak, bár páran szeretik és használják.

Más megoldások is terítéken vannak, bár mivel azok még ennyire sem elterjedtek, részletes ismertetésüktől itt eltekintek. Mindenesetre az látszik, hogy az ULAplus szabvány előtt még vannak kihasználatlan lehetőségek, amelyek a kicsi ám lelkes célközönség körében való elterjeszthetőség feltételeinek megfelelve remélhetőleg meg is fognak valósulni. Hogy még sokáig éljen a ZX Spectrum.

Nagy Dániel

HARDVER SIMOGATÓ

vDrive^{ZX}

Bevezető

Amikor 13. születésnapomra megkaptam életem első számítógépét egy ZX Spectrum 48k személyében (a nagybátyám hozta be Ausztriából „magán importként"), teljesen természetesnek tűnt, hogy a számítógépek egyetlen adatrögzítési lehetősége a kazettásmagnó. Később az iskolai számítógép szakkörben - ahol C64-ek voltak floppy-val -, értettem csak meg, hogy egy modern adattároló mennyire megnövelheti a gép funkcionalitását.

A Sinclair világban - az akkori trendekkel ellentétben -, floppy helyett egy apró, végtelenített szalaggal dolgozó kazettás eszköz, a Microdrive szolgált adatrögzítési megoldásként. Hazai elterjedtsége és a ZX Interface 1 magas ára miatt azonban a többségünk csak a nyálát csorgathatta utána.

Amikor év(tized)ekkel később engem is elkapott a 8 bit nosztalgia, hatalmas adósságot törlesztettem a gyerekkori énem felé, amikor beszereztem egy ZX Interface 1-et és hozzá egy Microdrive-ot.

Ekkor kellett szembesülnöm azzal a sokak számára ismerős jelenséggel, hogy az idő bizony megszépíti az emlékeket és a Microdrive annak idején nem véletlenül szorult háttérbe a floppy-kkal szemben. A kazetták nehezen beszerezhetőek és drágák, a többségük nem működik, beragad, szakad és általában kiszámíthatatlan a működésük. Erre jött még rá, hogy idővel a távtartó szivacs rendre szétporlad bennük, és ha nem cseréljük, akkor a szalagra tapadó darabkái teljesen tönkretehetik a kazettákat.

Többszöri próbálkozás és kudarc után már-már úgy tűnt, hogy a Microdrive számomra nem lesz több, mint egy használhatatlan technikai érdekesség a gyűjteményemben.

Ekkor találtam rá az új zélandi Charles Ingley fejlesztésére, a vDrive-ra.

Mi is az a vDrive^{ZX}

A vDrive egy SD-kártya alapú, hardveres Microdrive emulátor.

A vDrive-al lecserélhetjük egy eredeti Microdrive szalag-mechanikáját egy SD-kártya olvasóra, miközben mindenben ugyanúgy működik, mint az eredeti kazettásegység: az eredeti szalagkábelen csatlakoztathatjuk a ZX Interface 1-re és a Spectrum felé ugyanolyan Microdrive-nak látszik, mint egy kazettás meghajtó.

Tulajdonságok

Mivel a vDrive magát a Microdrive hardvert emulálja, így 100%-osan kompatibilis az eredeti Sinclair féle ZX Interface 1 - Microdrive rendszerrel, a működéséhez sem memóriában futó kódra, sem alternatív ROM-ra nincs szükség.

A vDrive a +3 kivételével minden ZX Spectrum típussal működik és Sinclair QL-hez is elérhető. (A vDrive^{QL} önálló termék, jelen leírás csak a ZX verzióra szorítkozik.)

Kompatibilis a ZX Interface 1 mindkét hivatalos ROM verziójával, továbbá a Multiface 1 és Multiface 128 kiegészítőkkel.

A vDrive a Spectrum emulátorok körében szabványnak számító .mdr kazettakép fájlokat használ a Microdrive kazetták emulálásához. Egyszerre több Microdrive egység emulálására is képes, összesen 8 virtuális meghajtót tudunk a Spectrumhoz csatlakoztatni a segítségével (mivel az Interface 1 maximum 8 meghajtót támogat).

A vDrive együttműködik az eredeti kazettás egységekkel is, azaz a vDrive mellett további Microdrive-okat is csatlakoztathatunk az Interface 1-hez.

A vDrive szabvány FAT16 és FAT32 formátumú SD-kártyákkal működik 32 GB maximális kapacitásig és a kártya tartalma PC-n is módosítható.

Beszerzés

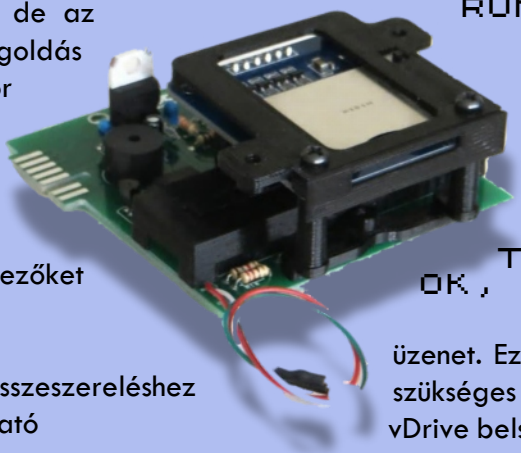
A vDrive^{ZX} közvetlenül a fejlesztőtől, Charles Ingley-től rendelhető. Mivel Charles hobbiként, elsősorban szórakozásból és a kihívás miatt fejlesztette (mint a legtöbb Spectrum kiegészítőt szokás manapság), csak kis tételben, közvetlen megrendelésre gyártja a vDrive-ot. A legegyszerűbb, ha személyesen lépünk kapcsolatba vele a vDrive hivatalos weboldalán vagy emailben , esetleg elkaphatunk egy-egy példányt a SellMyRetro oldalon is.

Az ára 49.95£, a postaköltség Új Zélandról 10£ sima légipostával (4-6 hét, nincs csomagkövetés) vagy 28£ nemzetközi csomagküldővel (2 hét, van csomagkövetés).

Hardver

A vDrive használatához a következőkre van szükség:

- 1 db. eredeti Microdrive meghajtóra. Nem kell működőképesnek lennie, de a doboza legyen ép.
- 1 db. eredeti, működő ZX Interface 1-re
- 1 db. szalagkábelre, ami a Microdrive-ot az Interface 1-hez köti
- 1 db. SD-kártyára
- és természetesen egy működőképes ZX Spectrumra, amely kompatibilis a ZX Interface 1-el.
  (Figyelem! Sajnos nem minden Spectrummal működik az Interface 1. Egyes modellekbe olyan Z80 processzor került, aminek az M1 vonala hibás, ez a Spectrum mindennapi működését nem zavarja, de az Interface 1-ét igen. A megoldás ilyenkor a Z80 processzor cseréje, ami ma már pár ezer forintból megoldható.)



A vDrive csomag a következőket tartalmazza:

- a vDrive egység
- csavarok és alátétek az összeszereléshez
- rövid összeszerelési útmutató

Összeszerelés

Elsőként szét kell szednünk az eredeti Microdrive-ot és eltávolítani a szalag-mechanikát. A mellékelt leírás szépen végig vezet a folyamaton, ábrák segítségével pontosan megmutatja, hogy milyen sorrendben melyik csavart kell eltávolítanunk.
A következő lépés a vDrive alaplap rögzítése a doboz aljához a mellékelt távtartó segítségével, majd az alaplapra kerül rá az SD-kártya foglalat.
Ezután a vDrive LED-jét kell az eredeti Microdrive dobozon lévő apró lyukba rögzíteni. Ez trükkös lehet, mert nem mindegyiken elég nagy a LED nyílás, ilyenkor nagyon óvatosan snitzerrel, ollóval, esetleg reszelővel ki kell szélesítenünk a lyukat, hogy az új egység LED-je kényelmesen beleférjen. Végül nincs más dolgunk, mint az eredeti borítást rácsavarozni a drive-ra.
Az egész művelet nem tart tovább pár percnél és még egy olyan barkács antitalentumnak, mint én vagyok sem okozott több bonyodalmat, mint egy távirányítós játék összeszerelése.
A vDrive használatba vételéhez ugyanúgy össze kell raknunk a hagyományos ZX Interface 1 - Microdrive rendszert, mintha nem emulált meghajtóval dolgoznánk: kikapcsolt állapotban a Spectrumra illesztjük az Interface 1-et, majd a szalagkábellel hozzá csatlakoztatjuk a Microdrive-ot.

Első lépések

Ha a mechanikai szerelés kész, ideje beüzemelnünk a vDrive-ot.
(A speciális utasításokat egyelőre nem részletezem, de ígérem, hogy a későbbiekben visszatérek rájuk.)

Vegyük ki az SD-kártyát a meghajtóból és kapcsoljuk be a gépet. Ha a hardver megfelelően működik, akkor egy apró csippanás és a drive-on folyamatosan világító zöld LED jelzi, hogy a vDrive szolgálatra kész.
Adjuk ki a

```
RUN
```

BASIC parancsot. Pár másodperc elteltével a drive ismét csippan egyet és megjelenik a képernyőn a

```
Toolkit installed
OK, 0:1
```

üzenet. Ez jelzi, hogy a vDrive beállításához szükséges Toolkit program betöltődött a vDrive belső flash memóriájából. Rakjuk be az SD-kártyát, majd adjuk ki a

```
.sdinit
```

parancsot. Ez formázza az SD-kártyát és létrehoz rajta egy a vDrive működéséhez szükséges konfigurációs fájlt. (Figyelem! A formázással a kártyáról minden adat el fog veszni!)
Most hozzunk létre egy „teszt” nevű kazettakép fájlt az SD-kártyán.

```
.mkimg "teszt"
```

Ez lesz az a fájl, ami a Spectrum felé a Microdrive kazettánkat emulálja.
Csatlakoztassuk a kazettaképfájlt egy virtuális meghajtóhoz:

```
.ld 1 "teszt"
```

Innentől a rendszer használatra kész, minden úgy működik, mintha egy eredeti Microdrive meghajtó lenne a géphez csatlakoztatva benne egy üres, formázatlan kazettával.
Nincs tehát más dolgunk, mint a hagyományos Spectrum BASIC utasítással formázni a „kazettánkat”:

```
FORMAT "m";1;"teszt"
```

Ha lefut a formázás, akkor a

```
CAT 1
```

paranccsal ellenőrizhetjük az eredményt. Ha mindent jól csináltunk a következő üzenet jelenik meg:

```
teszt
126
```

Azaz a kazetta neve és a felhasználható terület mérete kilobájtban (ez a ZX Interface 1 ROM verziójától függően lehet 127 - ROM v1 vagy 126 kilobájt - ROM v2).

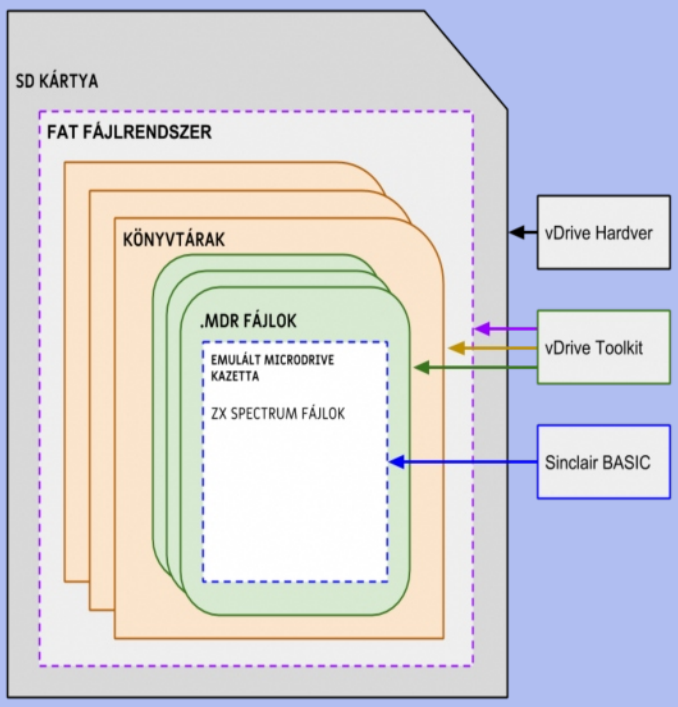
Toolkit

A bevezetőben említettem, hogy a vDrive tökéletesen kompatibilis az eredeti Microdrive-al és hogy semmilyen egyedi szoftver nem kell a működéséhez. Most mégis rögtön azzal kezdtük, hogy a drive beüzemelésekor betöltöttünk egy programot!
Hogy is van ez?

A válasz egyszerű: magához a Microdrive emulációhoz valóban nincs szükség semmilyen extra szoftverre, a vDrive beállítására és az SD-kártyán végzendő műveletekre ugyanakkor a Spectrum BASIC önmagában nyilvánvalóan nem alkalmas, erre való a Toolkit.
A Toolkit a Sinclair BASIC-et egészíti ki egyedi, vDrive kezelő utasításokkal.
Ha már beállítottunk mindent (létrehoztuk a kazettakép fájlokat, hozzárendeltük őket a meghajtókhoz, stb.), bármikor újraindíthatjuk a Spectrumot és a vDrive a Toolkit nélkül is a legutolsó beállításoknak megfelelően fog működni, pontosan úgy, mintha egy eredeti fizikai drive lenne a géphez kötve, benne egy eredeti kazettával. (Megjegyzés: ezeket a beállításokat nem is feltétlenül kell a Spectrumon végeznünk, az SD-kártyát csatlakoztathatjuk egy PC-hez is és egy sima szövegszerkesztővel módosíthatjuk a konfigurációs fájlt.)

A vDrive Toolkit használatakor élesen el kell különítenünk az SD-kártyán lévő fájljaink kezelését a virtuális Microdrive kazettákon lévőktől.
Az oldal felső sarkában látható ábrán megpróbálom bemutatni a vDrive logikai felépítését.

A hardver kezeli magát az SD-meghajtót, a Toolkittel tudjuk formázni a kártyát és létrehozni vagy módosítani a könyvtárakat és a kazettakép fájlokat, a Sinclair BASIC pedig az emulált kazetták



tartalmáért felelős. A Sinclair BASIC a Toolkit nélkül nem látja az SD-kártya tartalmát, a Toolkit pedig nem lát bele a kazettakép fájlokba.

A Toolkit betöltése
A Toolkitet a vDrive belső Flash memóriájából tudjuk betölteni. Távolítsuk el az SD-kártyát a drive-ból, Reseteljük a Spectrumot majd adjuk ki a BASIC **RUN** parancsot.

A Toolkit korrekt betöltését a

```
Toolkit installed OK, 0:1
```

üzenet jelzi.

Fontos, hogy SD-kártya ne legyen a drive-ban és hogy semmilyen más műveletet ne végezzünk a Spectrumon a RUN kiadása előtt.
Ha az SD-kártya bent maradt, akkor a

```
File not found, 0:1
```

hibaüzenetet fogjuk kapni.
Ha pedig csináltunk már valamit a Spectrummal (mondjuk **CAT 1** paranccsal megnéztük az aktuális kazetta tartalmát), akkor hiába vesszük ki az SD-kártyát a meghajtóból, a **RUN** nem a Toolkitet fogja betölteni, hanem sima BASIC utasításként fog lefutni és az eredménye a szokásos:

```
Program finished, 0:1
```

üzenet lesz.

Toolkit parancsok

Lássuk milyen műveleteket tudunk elvégezni a Toolkit segítségével.

Fájlkezelés

`.mkimg "image"` - létrehoz egy kazettaképfájlt a megadott néven, a névnek a FAT16 szabványnak kell megfelelnie: maximum 8 betű lehet, a kis és nagybetűket nem különbözteti meg egymástól, a fájl kiterjesztése mindig ***.mdr*** lesz

`.ren "file", "file"` - átnevezi a fájlt
`.rm "file"` - törli a fájlt
`.ro "file"` - írásvédetté teszi a fájlt *(a Spectrum úgy fogja látni, mint egy írásvédett kazettát)*

`.rw "file"` - írhatóvá teszi a fájlt

Könyvtár műveletek

A vDrive támogatja a könyvtárakat, de az alkönytárakat nem. Azaz az SD-kártyára tetszőleges számú könyvtárat létrehozhatunk, de azokba további alkönyvtárakat már nem tudunk. (Illetve de, mert a FAT megengedi, csak a vDrive nem fogja látni őket.)

`.mkdir "dir"` - könyvtár létrehozása
`.rmdir "dir"` - könyvtár törlése
`.cd "dir"` - belépés a könyvtárba, mivel a vDrive-on nincsenek alkönyvtárak, ezért nem kell a könyvtárból a DOS/Unix-hoz hasonlóan visszalépni (*cd ..*), hogy könyvtárat váltsunk, hanem az egyik könyvtárból közvetlenül ugorhatunk a másikba

`.ls` - listázza a könyvtár tartalmát
`.li` - listázza a könyvtárban lévő kazettakép fájlokat *(tehát csak a .mdr kiterjesztésűeket mutatja)*

`.cp "file", "dir"` - átmásolja a fájlt a megadott könyvtárba

`.mv "file", "dir"` - átmozgatja a fájlt a megadott könyvtárba

Drive műveletek

Ahhoz, hogy a kazettakép fájlokat használatba vegyük, virtuális meghajtókhoz kell rendelnünk őket. Egy kazetta behelyezése a virtuális drive-ba az

`.ld n "image"`

utasítással történik. Ezzel az ***n*** jelű meghajtóhoz csatoljuk a megadott kazettaképfájlt.

A vDrive összesen 8 meghajtót tud egyszerre emulálni, hogy éppen mennyire van szükségünk, azt a Toolkittel tudjuk beállítani.

`.mkdrv n` - ***n*** számú meghajtó hozzáadása *(n 1 és 8 között)*

`.rmdrv n` - ***n*** számú meghajtó elvétele
`.lv` - kilistázza a meghajtókat

Fontos, hogy ***n*** ez esetben nem a meghajtó jelét, hanem a meghajtók maximális számát jelöli, azaz ha 8 meghajtót csatolunk a rendszerhez, akkor az **.rmdrv 5** utasítás nem az ötödik meghajtót fogja leválasztani, hanem 5 meghajtót a legnagyobbtól kezdve, azaz az utasítás hatására csak az 1. 2. 3. meghajtó lesz a rendszerhez csatlakoztatva. Ugyanígy, ha már van 3 meghajtónk, akkor az **.mkdrv 5** utasítás végeredménye összesen 8 meghajtó lesz.

A vDrive együttműködik eredeti Microdrive egységekkel is, ilyenkor a vDrive-ot kössük össze elsőként az Interface 1-el és a vDrive után csatlakoztassuk a további meghajtókat. Fontos, hogy ilyenkor az emulált meghajtók száma annyival kevesebb lehet csak, ahány eredeti Microdrive-ot csatlakoztattunk a rendszerhez. A meghajtók fizikai címe folytonos, azaz a vDrive utáni meghajtó száma mindig nagyobb lesz, mint az emulált meghajtóké. Azaz, ha a vDrive 3 meghajtót emulál, akkor a vDrive után csatlakoztatott Microdrive egység száma 4 lesz, két emulált meghajtó közé nem tudjuk „beékelni”.

Bank műveletek

A „bank” meghajtók és a hozzájuk rendelt kazettaképfájlok csoportja, amelyek között egyetlen paranccsal válthatunk.
A bankok használata a vDrive egyik igen hasznos tulajdonsága. Ha egyszerre több meghajtót használunk különböző kazettaképekkel és gyakran akarunk váltogatni közöttük, akkor a különböző számú meghajtók hozzáadása és a kazettaképek meghajtókhoz rendelése előbb-utóbb kényelmetlenné válik. Ezt a folyamatot egyszerűsíthetjük le a bankok használatával.

Nézzük meg a bankok működését egy konkrét példán keresztül.

Hisoft assemblerrel (ami szerencsére támogatja a Microdrive-ot) akarunk fejleszteni gépi kódú programot. Három drive-ot szeretnénk egyszerre használni: az elsőre kerül maga az assembler, a másodikra a kód szövege és a harmadikra a lefordított program.
A kazettakép fájlok nevei az SD-kártyán: hiasm.mdr, pr01text.mdr, pr01code.mdr lesznek.

Hozzunk létre egy bankot:

```
.mkbank "testbank"
```

aktiváljuk:

```
.sb "testbank"
```

adjuk hozzá a drive-okat:

```
.mkdrv 2
```

(A bank létrehozásakor automatikusan létrejön egy drive is, azaz ha a példánkban 3 meghajtóval akarunk dolgozni, akkor további kettőt kell még a bankhoz adnunk.)
Végül a meghajtókhoz rendeljük a kazettakép fájlokat:

```
.ld 1 "hiasm"
.ld 2 "pr01text"
.ld 3 "pr01code"
```

Ezzel kész is vagyunk. Ahányszor kiadjuk az

```
.sb "testbank"
```

utasítást, a vDrive létrehozza a három meghajtót, bennük a megfelelő kazettaképpel és a rendszer máris használatra kész.
Ezeket a beállításokat csak egyszer kell elvégeznünk minden banknál, utána az **.sb** „*banknév*” utasítással kedvünkre váltogathatunk közöttük. Így létrehozhatunk külön bankokat játékokhoz, archiváláshoz, fejlesztéshez, stb.

Toolkit bank utasítások:

`.mkbank "name"`	- bank létrehozása
`.rmbank "name"`	- bank törlése
`.lb`	- kilistázza a bankokat
`.sb "name"`	- bank aktiválása

BASIC rövidítések

Mondjuk ki őszintén: a Sinclair BASIC Microdrive műveletei túlkomplikáltak.
Egy BASIC program betöltéséhez a klasszikus és egyszerű

```
LOAD "programnév"
```

utasítás Microdrive használatakor már így néz ki:

```
LOAD *"m";1;"programnév"
```

A Toolkit ezért tartalmazza a leggyakrabban használt Microdrive kezelő BASIC utasítások rövidített formáját:

`.C n`	- CAT n
`.L n "name"`	- LOAD *"m";n;"name"
`.S n "name"`	- SAVE *"m";n;"name"
`.V n "name"`	- VERIFY *"m";n;"name"
`.F n "name"`	- FORMAT "m";n;"name"
`.E n "name"`	- ERASE "m";n;"name"
`.M n "name"`	- MERGE *"m";n;"name"

Egyéb Toolkit utasítások
A Toolkit tartalmaz továbbá néhány utasítást, melyek a vDrive hardver és a Toolkit működéséhez szükségesek.

`.sdinit`	- formázza az SD-kártyát és létrehozza rajta a vDrive konfigurációs fájlt
`.mkcfg`	- felülírja az SD-kártyán az aktuális vDrive beállításokat az alap beállításokkal
`.tkon`	- Toolkit bekapcsolása
`.tkoff`	- Toolkit kikapcsolása
`.sndon`	- hang (csipogás) bekapcsolása
`.sndoff`	- hang kikapcsolása
`.vinfo`	- vDrive firmware és Toolkit verzió megjelenítése
`.update`	- vDrive firmware frissítése
`.help`	- Toolkit parancsok listája

Összefoglaló

Mire jó a vDrive^{ZX}?
Manapság a Spectrum-hoz fejlesztett modern adattárolási megoldások közül alighanem a vDrive biztosítja a legjobb kompromisszumot a hatékonyság, megbízhatóság és az eredeti „Sinclair életérzés” visszaadása között.
Látványra a vDrive változtat a legkevésbé az eredeti Spectrum design-ján, így ha fontos számunkra, hogy minél kevesebb modern eszközt aggassunk a Spectrumunkra és a gépünk használata során minden úgy történjen, ahogy annak idején, akkor a vDrive ideális megoldás.
Ha pedig az eredeti eszközökkel, eredeti hardveren akarunk Spectrumra fejleszteni, a vDrive-nál keresve sem találhatnánk jobb eszközt, mivel az összes Microdrive-ot kezelő szoftver és hardver garantáltan működik a vDrive-val.
Ezen kívül, ha komolyabb Microdrive kazetta gyűjteménnyel rendelkezünk, akkor a vDrive kitűnő eszköz az archiválásukra. Mindössze egy második, működőképes Microdrive egységre lesz szükségünk és

a kazettáinkat SD-kártyára menthetjük a vDrive segítségével, a lementett kazettaképfájlokat pedig PC-n tárolhatjuk, vagy megoszthatjuk őket a neten.

Mire nem jó a vDrive^{ZX}?

Ha magával a Spectrum BASIC-el és az eredeti hardverrel csak annyit akarunk bajlódni, hogy betöltjük a WOS-ról letöltött kedvenc játékainkat, arra léteznek a vDrive-nál sokkal praktikusabb SD-kártya alapú megoldások.

Ami a vDrive előnye: a 100%-os kompatibilitás az eredeti Sinclair rendszerrel... nos ez egyben a hátránya is: a Microdrive és az Interface 1 már a maga idejében is sok játékkal inkompatibilis volt, a gyér elterjedtsége miatt alig adtak ki Microdrive-on szoftvereket, játékokat meg szinte semmit, így a játékosoknak több gondot okozott, mint hasznot. A trükkös loaderekkel, turbókkal és másolásvédelemmel ellátott programokat nehéz Microdrive-ra menteni, így hiába kényelmes és modern megoldás az SD-kártya, a vDrive szűk keresztmetszete maga az eredeti Sinclair féle Microdrive rendszer.

Jó hír ugyanakkor, hogy a vDrive kompatibilis a Multiface 1-el (és a 128-al), így végső soron lehetséges a teljes memóriamentés direktben a vDrive-ra, ami egyfajta megoldásként szolgálhat a játékok Microdrive-ra mentésére is.

A PC-s Fuse emulátor ráadásul a Microdrive-ot és a Multiface-t is egyaránt támogatja, így ha csak a WOS-ról letöltött játékok .mdr-be konvertálása a cél, azt a Fuse-vel a PC-n is elvégezhetjük és az elkészült .mdr fájlokat átmásolhatjuk az SD-kártyára, így végső soron a Multiface hardverrel nem rendelkező vDrive felhasználók sincsenek teljesen elvágva a játékoktól.

Személyes tapasztalatok

A kb. egy év alatt, amióta lehetőségem volt használni, nagyon megszerettem a vDrive-ot. Egy olyan eszközt sikerült beszereznem, ami pont azt adta a Spectrumhoz, ami számomra nagyon hiányzott: egy megbízható, modern adattárolót, ami ugyanakkor nem rondít bele a Spectrum eredeti atmoszférájába.

A Microdrive kezelése BASIC-ből pont ugyanannyira nehézkes és kényelmetlen, mint annak idején volt, de ez számomra pozitívum, mert semmiképpen sem akartam egy olyan rendszert, ami alig különbözik egy PC-s emulátortól.

Az élmény tökéletességéhez egyetlen dolog hiányzik csak a vDrive-ból: a motor búgása és a kazetták jellegzetes, surrogó hangja. Ki tudja, Charles talán a következő szériába még ezt is beépíti.

Függelék

A Toolkit `.help` fordítása:

`.LD n "image"`	- fájl betöltése az n meghajtóba
`.LB`	- bankok listája
`.LI`	- kazettakép fájlok listája *(az aktuális könyvtárban)*
`.LS`	- minden fájl és könyvtár listája (az aktuális könyvtárban)
`.LV`	- a virtuális Microdrive egységek listája
`.CD "dir"`	- könyvtár váltás
`.CP "file","dir"`	- fájl másolás
`.MV "file","dir"`	- fájl mozgatás
`.REN "file","file"`	- fájl átnevezés
`.RM "file"`	- fájl törlése
`.SB "bank"`	- aktív bank beállítása
`.MKIMG "image"`	- új kazettakép létrehozása
`.MKBANK "bank"`	- bank létrehozása
`.RMBANK "bank"`	- bank törlése
`.MKDRV n`	- n vMdrive hozzáadása
`.RMDRV n`	- n vMdrive törlése
`.MKDIR "dir"`	- könyvtár létrehozása
`.RMDIR "dir"`	- könyvtár törlése
`.RO "image"`	- kazettakép fájl írásvédetté tétele
`.RW "image"`	- kazettakép fájl írhatóvá tétele
`.SND [ON/OFF]`	- a vDrive hang (csipogás) ki- és bekapcsolása
`.TK [ON/OFF]`	- Toolkit ki- és bekapcsolása
`.SDINIT`	- SD-kártya formázása
`.MKCFG`	- Eredeti beállítások visszaállítása
`.VINFO`	- vDrive firmware információk
`.UPDATE`	- vDrive firmware frissítése
`.C n`	- CAT rövidítés
`.L n "name"`	- LOAD rövidítés
`.S n "name"`	- SAVE rövidítés
`.V n "name"`	- VERIFY rövidítés
`.F n "name"`	- FORMAT rövidítés
`.E n "name"`	- ERASE rövidítés
`.M n "name"`	- MERGE rövidítés
`.HELP`	- Toolkit help

(A help nagybetűkkel írja ki a Toolkit utasításokat, de kisbetűkkel is ugyanúgy működnek.)

Kovács Péter (kpbendi)

JÁTÉKÚJDONSÁGOK

ALL HALLOWS (RISE OF THE PUMPKIN)

Kiadó: **Rucksack Games**
Szerzők: **John Blythe (programozás, grafika), Andy Johns, Allan Turvey, David Saphier (egyéb programozás), Loner, Sam Styler, Alex (zene)**
HW igény: **48K/128K**
Stílus: **arcade/platform**
Vezérlés: **Billentyűzet (definiálható, alapértelmezésben Z, X - balra, jobbra) és Kempston, Sinclair Joystick**
Megjelenés: **2018. szeptember, ingyenes**

John Blythe 2017 óta ontja a szebbnél szebb AGD-vel készült játékait, melyek játékmenetben talán nem hoznak sok újat, de minden más szempontból nagyon hagulatosak.
Ezer évnyi börtönbüntetésed véget ért! Nyújtóztasd ki elgémberedett tagjaidat… vagyis ne, mert nincs olyanod, egy végtagoktól mentes, klasszikus tök vagy, már amennyiben egy átlagos tök ezer évig életben marad egy börtönben... Egyáltalán milyen tököt börtönöznek be? Szóval nem lehetsz igazán átlagos, és még ugrálni is tudsz, ami némi rugalmasságot feltételez, szemben a vacak, középszerű tökökkel. A feladatod miatt pedig azt kell mondjam, hogy igazán tökös tök vagy, ugyanis nem kisebb ügyet vállaltál fel, mint hogy szembe szállsz a rémséges Erdei Úrral (béna név, nem is félelmetes, pedig nagyon gonosz egy pali). Öt holdkövet kell a sikerhez bezsebelned. Ne barátkozz senkivel, aki mozog, az nem szimpatikus, ráadásul már anyukád is megmondta több mint ezer éve, hogy ne állj szóba idegenekkel!

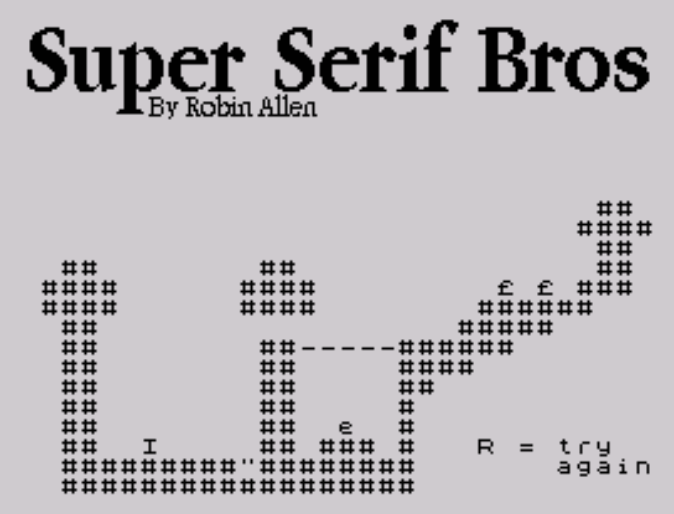
GANDALF DELUXE

Szerzők: **Cristian M. Gonzalez, Alvin Albretch, Einar Saukas (programozás), S9, Beykersoft (zene), Fede "Abu Simbel" (tesztelés)**
HW igény: **128K**
Stílus: **arcade/platform (Super Mario Bros-klón)**
Vezérlés: **Billentyűzet (definiálható, alapértelmezésben Q, A, O, P, M - fel, le, balra, jobbra, tűz) és Kempston, Sinclair, Cursor Joystick**
Megjelenés: **2018. október, ingyenesen letölthető, és megvásárolható kazettán a normál változata 8.75, illetve az "enhanced" változata 25 euroért a Matranettől**

A *Gandalf Deluxe* nem egy teljesen új játék, hanem a ZX-DEV Conversions'2017 versenyre megjelent Gandalf (9. lett a 18 fős mezőnyben) teljesítményben és játék alatti zenében tuningolt változata. A megjelenés rendkívül színes, amit Einar Saukas Nirvana Plus grafikai motorjának használata eredményez.

A történet röviddel a gyűrű megsemmisítése után játszódik, mikor Gandalf visszatér Középföldére bosszút állni. Meg kell semmisítenie Középfölde négy utolsó démonját, ez természetesen csak akkor sikerülhet, ha segítesz neki. A játék négy körből áll, minden kör 16 képernyőn játszódik, a körök végén meg kell küzdened egy-egy démonnal, hogy tovább léphess.

SUPER SERIF BROS

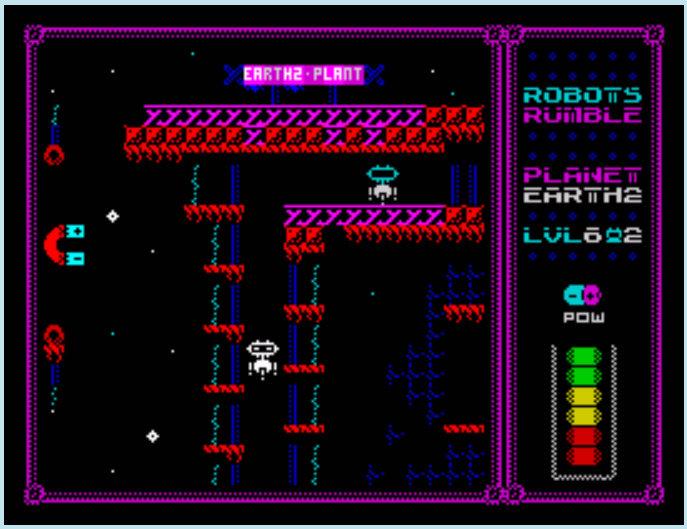


Szerzők: **Robin Allen**
HW igény: **48K**
Stílus: **arcade/puzzle**
Vezérlés: **Q, A, O, P - fel, le, balra, jobbra**
Megjelenés: **2018. szeptember, ingyenesen**

Egy Java klasszikus Spectrumon fekete-fehér ASCII grafikával. A kuszának látszó karaktertömeg valójában egy bonyolult és jól megtervezett kirakós játékot rejt, ahol a cél nagyon egyszerű: embereddel (I) szedd össze a pénzt (£), majd érd el a kijáratot (E). Ahhoz, hogy ez sikerüljön, használd a lifteket("), vízszintesen mozgó platformokat (T), aktiváló kapcsolókat (~) és még sok egyebet.

ROBOTS RUMBLE

Cím: **Robots Rumble**
Szerzők: **Miguetelo**
HW igény: **48K/128K**
Stílus: **arcade/akció**
Vezérlés: **Billentyűzet (Q, A - bal oldali mágnes mozgatása fel, le, P, L - jobb oldali mágnes mozgatása fel, le)**
Megjelenés: **2018. szeptember, ingyenesen**



Miguel Ángel Tejedor, vagyis Miguetelo negyedik játéka, mindegyik AGD-vel készült, de egyik sem egy tipikus sétálgatós platformjáték.

Neved Slade, 45 éves vagy, robotzúzóként dolgozol. Különböző bolygókra küldenek el a robotok elpusztítására, ahol különbözőek a körülmények, de a küldetésed ugyanaz: két hatalmas régi mágnes segítségével a lávába navigálni a robotokat. El kell kerülnöd a robotokkal a zöld kryptonit köveket, ugyanis ha a robotok hozzáérnek, akkor felrobbannak, és a detonáció az egész bolygót beszennyezi. Kerülnöd kell még a növények őrzőit is. A mágneseknek korlátozott a töltöttségük. Fel tudod őket tölteni, ha felveszed az elemeket, de ha elfogy

az energia, akkor meghalsz. Ha mindkét mágnes próbálja mozdítani a robotot, akkor az mozdulatlanul fog állni. Használhatod a mozgó platformokat a mélyedések áthidalására, a fénylifteket fel és lefelé mozgatásra, stb.

POOPER SCOOPER

Kiadó: **The Death Squad**
Szerzők: **Davey Sludge (programozás), Yerzmyey (zene)**
HW igény: **48K+AY**
Stílus: **arcade**
Vezérlés: **Billentyűzet (Q, A, O, P, Space - fel, le, balra, jobbra, tűz) és Sinclair Joystick**
Megjelenés: **2018. szeptember, ingyenes**

Ahogy Davey Sludge eddigi játékainak többsége, így ez is székletközpontú.
A játék folyamán négy különféle, különböző képességekkel bíró szereplővel kell megtisztítanod a khm, szóval hát a szaros képernyőt, miközben elkerülöd az ellenségeket. Az említett főszereplők a képernyő aljának bal és jobb szélén várakoznak, aktiválni úgy tudod őket, hogy megérinted őket. Bal oldalon látható Scott, a baseballsapkás, bunyós srác, aki az öklével próbál tárgyakat mozgatni, alatta a szarügyi ügyintéző, Mark Barton Dung, jobb oldalon Willy, a darázs és Chav Doley, a laza kukac. A játék során kell felfedezned, hogy ki mit tud, mihez ért, és sokszor csapatként kell őket irányítanod a pályák teljesítéséhez.

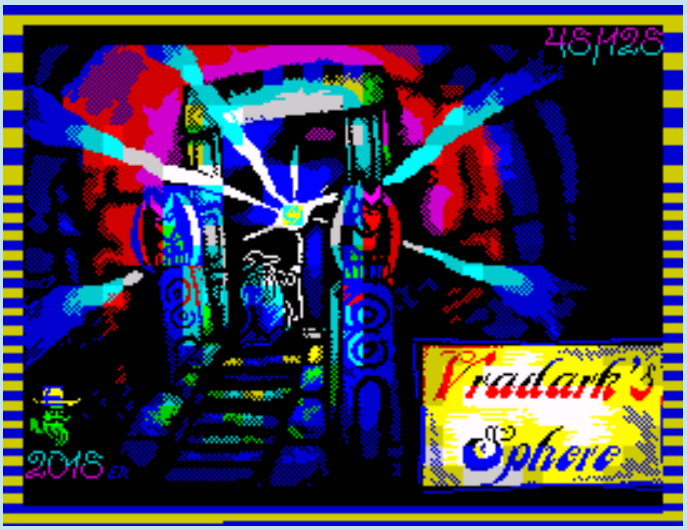
NIGHT STALKER ZX

Kiadó: **AMCgames**
Szerzők: **Aleisha Cuff**
HW igény: **48K**
Stílus: **arcade/labirintus**
Vezérlés: **Billentyűzet (Q, A, O, P, Space vagy M - fel, le, balra, jobbra, tűz) és Kempston, Sinclair Joystick**
Megjelenés: **2018. szeptember, ingyenes**

1982-ben jelent meg Intellivision-re Steve Montero nagy sikerű játéka, a Night Stalker. A kanadai illetőségű AMCgames 26 évvel később készítette el a játék ZX Spectrum átiratát Jonathan Cauldwell Arcade Games Designere segítségével.

Menekülsz. A támadóid hajthatatlan robotok. Mikor megsemmisítesz egyet közülük, felváltja egy még okosabb, gyorsabb robot. Ez egy rémálom. A labirintusban találhatsz fegyvereket, melyeket használhatsz a robotok és állatok ellen. Mikor egy fegyver kiürül, kerüld el az ellenségeket, míg nem találsz egy másik fegyvert. A pókok és denevérek nem veszik el ugyan az életedet, viszont megbénítanak rövid időre, ha hozzájuk érsz.

VRADARK'S SPHERE



Kiadó: **Sanchez crew**
Szerzők: **Sanchez (programozás), ER (grafika), Nik-O (zene)**
HW igény: **48K/128K**
Stílus: **roguelike**
Vezérlés: **Billentyűzet (definiálható, alapértelmezésben Q, A, O, P, Space, I - fel, le, balra, jobbra, tűz, info) és Kempston Joystick**
Megjelenés: **2018. szeptember, 1 EUR-ért letölthető, vagy a demo változata ingyenesen letölthető**

A Vradark's Sphere sokkal egyszerűbb játék, mint Sanchez (Aleksander Udotov) korábbi munkái (Survivisection, Castlevania, Mighty Final Fight). Az egyszerűség azonban csak látszólagos, ami a játék műfaja miatt van, azonban amit egy roguelike (egyfajta RPG, amit az 1980-as Rogue játékról neveztek el) játékból ki lehet hozni, azt Sanchez és csapata ki is hozta.

Az ősi vár romja a legnagyobb gonosz otthona lett, Vradark ül a trónon. Küldetésed mi más is lehetne, mint megszüntetni ezt a baljós állapotot. Vradark ördögi diktatúrájának megdöntéséhez használd fel egyre gyarapodó élet- és tűzerődet, ami először talán inkább egyre fogyatkozónak tűnik majd, de hamar belejössz, ám mindenek előtt ne feledkezz meg a logikáról, türelemről és stratégiáról sem, szükséged lesz rájuk! Varázslattal véletlenszerűre generált labirintusokon kell átverekedned magad a végső siker eléréséhez, a mágikus gömb visszaszerzéséhez.

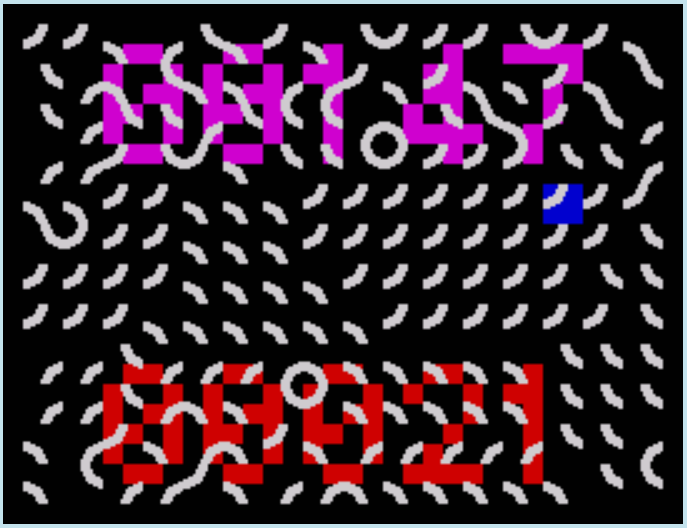
Öld meg a szörnyeket, kutasd át a ládákat, sírokat, némelyikben hasznos zsákmányokat találhatsz, melyekkel feltöltheted apadó élet- és tűzerődet (piros lombik - élet, kék lombik - tűz). Ne kapkodj, inkább gondolkozz (ezért jó az RPG, nem támadhatnak meg hirtelen, csak ha te is lépsz, akkor lép az ellenség is), figyeld meg ellenlábasaidat (az info gomb segítségével láthatod, hogy mennyi az életerejük, és hogy mennyire pusztítóak), ha lehetséges, inkább kerüld el őket. Kilenc szintet kell végigjárnod, minden szinten van egy kijárat, és egy azt nyitó, véletlenszerű helyen tartózkodó kulcs. Ha megtalálod a kulcsot, már mehetsz is a kapuhoz, és irány a következő szint.

GRID I

Szerzők: **g0blinish**
HW igény: **48K+AY**
Stílus: **arcade/logikai**
Vezérlés: **Billentyűzet (Q, A, O, P, M és Space - fel, le, balra, jobbra, tűz), Cursor Joystick**
Megjelenés: **2018. szeptember, ingyenes**

Dmitry Krapivin egy ismeretlen szerző flash-alapú játékát ültette át Spectrumra.

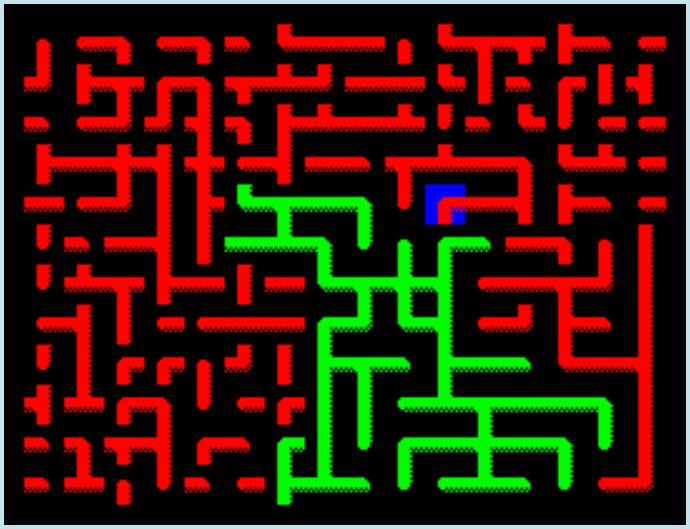
A képernyőn csődarabkákat látsz, ha valamelyiket megmozdítod a tűz gomb lenyomásával és a forduló csődarab valamelyik vége egy másik csődarab valamely végéhez ér, akkor a megérintett csődarab is fordul egyet, és ha szintén hozzáér egy vége egy



másik végéhez, akkor az is fordul, és így tovább... A célod az, hogy a lehetséges 31 fordításod egyikével a lehető legmagasabb számú ívdarabot forgasd el az előbbi elv szem előtt tartásával.

PIPES!

Szerzők: **g0blinish (programozás), Cosmix (zene)**
HW igény: **48K+AY**
Stílus: **arcade/logikai**
Vezérlés: **Billentyűzet (Q, A, O, P, M és Space - fel, le, balra, jobbra, tűz), Cursor Joystick és Kempston egér**
Megjelenés: **2018. szeptember, ingyenes**



Dmitry Krapivin, alias g0blinish 2018 második félévében számolatlanul ontja a logikai játékokat. A Pipes! egy konverzió, az eredeti szerzője, Michael Kerley és mobileszközökre készítette el a játékot. A játék egyik fontos paramétere a 'seed' (ez egy véletlenszerű érték, melyet a szintek generálásához használ a szerző). A szintek létrehozása sok időt igényel, közben a játék mutatja a 'seed' értéket, és hogy éppen nagyon csinál valamit.
A feladatod az, hogy a képernyőn lévő csőhálózat összes szegmensét addig kell fordítgatni, míg azok összeállnak egy összefüggő nagy és zöld hálózattá. A csődarabkákat az óramutató járásával megegyező irányban lehet forgatni, amint egy forgatott csődarab összeköttetésbe kerül a képernyő közepén lévő zöld csővel, az is zölddé válik. A csövekben nem áramlik az égvilágon semmi, nem fog kifolyni, nem fog gázosítani, így nem is kell sietned, gondolkozhatsz, ameddig rá nem jössz a megoldásra.

MAGIC

Szerzők: **g0blinish (programozás), jimbo_77 (grafika)**
HW igény: **48K**
Stílus: **arcade/logikai**
Vezérlés: **Billentyűzet (Q, A, O, P, M és Space - fel, le, balra, jobbra, tűz), Cursor Joystick**
Megjelenés: **2018. augusztus, ingyenes**

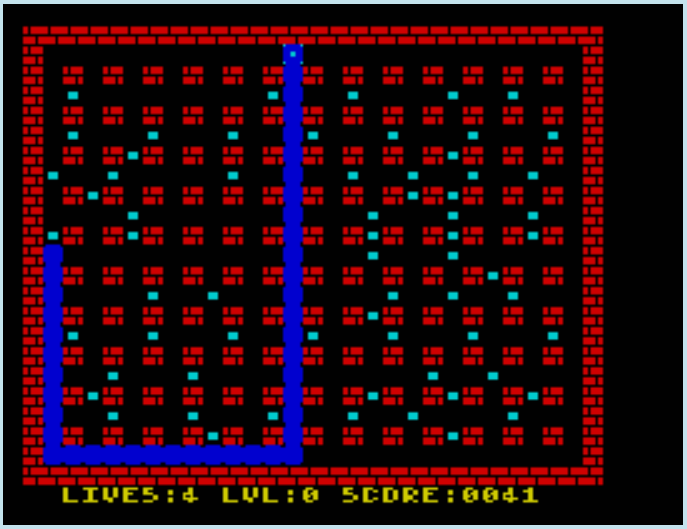
Dmitry Krapivin az 1990-ben Atarira megjelent Tensoft játékot ültette át nekünk tetsző platformra.

Mágiára is szükséged lesz a feladatod teljesítéséhez. Egy varázsló vagy, és rendetlen raktárakban kell összeszedned a csillogó, kék gyémántokat. Nem kimondottan varázslóknak, inkább földi halandóknak való feladat, azonban felhasználhatod a tudományod, mikor túl sűrűnek találod a dobozok helyzetét eltűntethetsz egy-egy dobozt (ha a magic számlálód ezt megengedi). Ugyanis nem vagy elég kigyúrt több láda egyidejű mozgatásához, csak egy dobozt tudsz egyszerre tologatni. Csak

tologatni, lökdösni és taszigálni, de húzogatni, vonogatni nem!

SLITHER

Szerzők: **g0blinish**
HW igény: **48K**
Stílus: **arcade/akció**
Vezérlés: **Billentyűzet (Q, A, O, P - fel, le, balra, jobbra), Cursor Joystick**
Megjelenés: **2018. augusztus, ingyenes**



Dmitry Krapivin egy 1985-ös, Atari 8 bitesre kiadott Snake-klónt írt át Spectrumra. Érdekes, hogy a WOS-on van fent egy hasonló nevű és stílusú, ám pillanatnyilag elérhetetlen játék, ami 1983-ban lett kiadva. Lehet, hogy az a játék volt az Atari változat eredetije?

A Slither koncepciója nagyon egyszerű, vezesd a kis hernyót, melynek az összes kék pontot el kell fogyasztania. Minden alkalommal, amikor egy pontot megeszik, mérete növekszik, ami nehezebbé teszi a manőverezését. Az akció a második szinttől kezd eldurvulni, mivel onanntól egy valódi labirintusban kell az egyre hosszabb, önveszélyes kígyót navigálni.

Mezei Róbert (m/zx)

Az alábbi magyar fejlesztésű ZX Spectrum programokat keressük z80/tap/tzx formátumban vagy akár kazettán, természetesen keressük a hozzávaló kazettaborítót, leírást magyar vagy akár angol nyelven is.

Játékprogramok:

- 3D Sakk (Novotrade)
- Alquerque (InterBit)
- BASIC compiler (InterBit)
- Betűpóker (Novotrade)
- Betűrömi (Novotrade)
- Erdélyi fejedelmek, Habsburg uralkodók (Sági György, 1986, KLTE OK.)
- Gazdasági játékok (16K) (InterBit)
- HOT-HIT (Interbit)
- Memória (Király Péterné; Király-Török SW)
- Színkereszt (16K) (InterBit)
- Tervezhető CHAOS (Héra Tibor)
- Tologató játék (INTEGRÁL)
- Trilog kazettaborító (Novotrade)
- Űrjátékok (16K) (InterBit)

Oktatóprogramok:

- Hangbűvölő (oktató, InterBit)
- Teszt-Mester (oktató, InterBit)

Segédprogramok:

- BASIC compiler (InterBit)
- Felhasználói cartrige (InterBit)
- EPROM égető program (DEMAK)
- HUN-CHAR (Tasword) (INTEGRÁL)
- KATALÓGUS (KRONOSZ)
- Monitor-Disassembler (InterBit)
- Spectmusic (Rozsnyai György)

Ügyviteli programok:

- Adófejtő (INTEGRÁL)
- Címletezés (LSI ATSZ)
- Bruttósító (???)
- GMK jövedelemadózás (LSI ATSZ)
- Havidíjasok bérelszámolása (LSI ATSZ)
- Jövedelemadó-számítás és -nyilvántartás (INTEGRÁL)
- Keresetszint szerinti adózás (LSI ATSZ)
- SZJA'88
- Üszőnyilvántartás (INTEGRÁL)

A speccyalista@sinlair.hu email címen értesíthesz bennünket, ha neked megvan valamelyik. Természetesen, ha olyan magyar fejlesztésű program van a birtokodban, ami ebben a listában nem található, akkor se késlekedj:

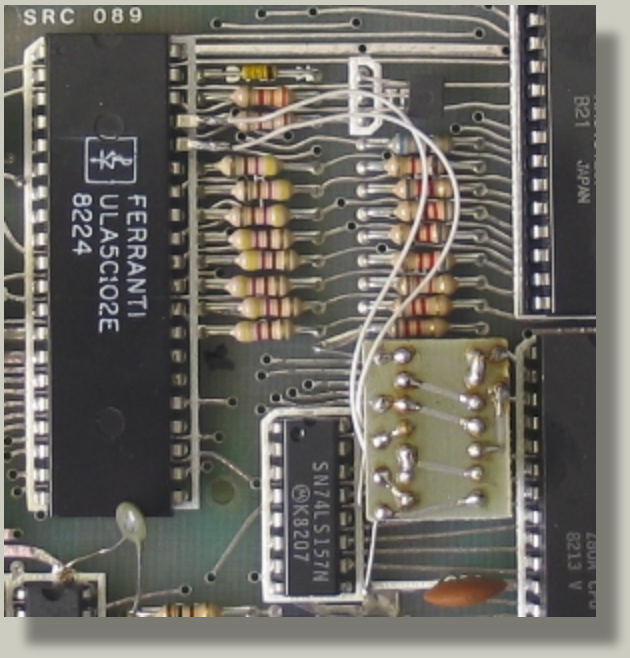
KERESSÜK

SPECTRUMOLÓGIA

ZX Spectrum ULA típusok

Az ULA a ZX Spectrum mikroszámítógépek „lelke”. Az ULA-t (Uncommitted Logic Array-Többcélú Logikai Áramkör) a Ferranti nevű cég fejlesztette és a Sinclair elsők között kezdte használni. Első változata a ZX81-ben használt 2C158E ULA, amely a ZX80 diszkrét elektronikai alkatrészeit egy NMI (nem maszkolható megszakítás) áramkörrel egyesítette.
A 16/48k-s ZX Spectrumokban az alábbi ULA verziókkal lehet találkozni:

5C102E ULA
Az első Spectrumos ULA. Az összes Issue 1-es és a korai Issue 2-es modellekbe került.
Ez a sorozat sajnos hibásan készült el. Az ULA beépített megszakítás kezelője (NMI) bizonyos feladatoknál le kell tiltsa a CPU hozzáférését az alsó memóriához, hogy ott az ULA I/O (írás/olvasás) műveletet végezhessen. Az Issue 1-nél azonban akkor is megszakította a CPU működését, ha az pl. a billentyűzetet olvasta le (Inkey$). Emiatt a billentyűzet figyelés kb. 50%-osan működött. A hiba javításához egy 74LS00 áramkört forrasztottak a NYÁK-ra. Később saját kis segéd NYÁK-ot kapott *(Cockroach mod)*.



5C112E, 5C112E-2, 5C112E-3 ULA
Az ULA második, javított verziói. Az előző ULA hibáját javították, de ettől újabb hiba keletkezett. Erre kétféle javítás létezik. Az első egy diódával és egy ellenállással (korai Issue 2 NYÁK lapok), a második pedig egy ZTX313 tranzisztorral oldja meg a működést.

Mivel ezt a hibát már nem tudták az ULA-n belül megoldani, ezért az Issue 3-as NYÁK verziótól kezdődően, a ZTX313 tranzisztor TR6 néven kapott helyet *(Spider mod)*.

Jelenleg nem tisztázott, hogy milyen különbségek vannak az eredeti 5C112E és a -2 és -3 változatok között.

6C001E-5 ULA
Az első 6C sorozatú ULA, újdonsága a csökkentett áramfelvétel, illetve kompatibilis lett olyan TV készülékekkel, amikkel a korábbiak nem működtek, igaz ennek következtében a képet egy karakterrel balra tolva jeleníti meg (ez legjobban fekete BORDER színnél látszik).
Egy belső változtatásnak köszönhetően az EAR bit értéke az ULA bemelegedéséig 0 és 1 között lebeg, ellentétben az 5C szériájúakkal, ahol fixen 1 értékű. Emiatt az olyan programok, amelyek a billentyűzetet úgy olvassák ki, hogy közben nem maszkolják a többi adatbitet (EAR bit), nem működnek megfelelően.
Az ilyen ULA-val szerelt gépekben kis kondenzátort szereltek az alsó RAM /RAS (címsor)-vonalára, feltehetően azért, hogy felerősítse az ULA által generált gyenge jelet.
Aránylag ritkán lehet ilyen ULA-val találkozni. Csak a kései, 1983-as év 20. és 24. hete között gyártott Issue 2-es alaplapokba került ilyen.

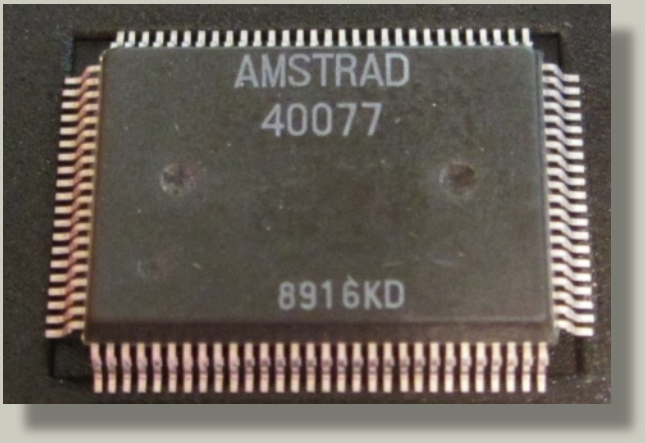
6C001E-6 ULA
A 6C ULA-k első javított verziója. Kései Issue 2-es lapokra, illetve az Issue 3/3B lapok túlnyomó többségére ez került. Feltehetően javították benne a /RAS-vonal gyenge jelszintjét.

6C001E-7 ULA
A 48K gépek utolsó ULA változata. Az Issue 4A nyákon megjelenő változtatásokkal együtt javítja a memória időzítő jelet. Ez az egyetlen ULA változat, ami az összes 48k-s ZX Spectrummal működik.
Néhány, az utolsók között gyártott ilyen szériás ULA-n ’PS’ jelzés, vagy a Plessey nevű cég hullámos logója van, mert 1988-ban megvásárolták a Ferranti

félvezető részlegét. Ezek az ULA-k feltehetően már az Amstrad-dal szerződött szervizeknek készültek.

7K010E-5 ULA / Amstrad 40056
Ezek az ULA-k a 128k-s Spectrumhoz, illetve a szürke +2-eshez készültek. Bár különböző az elnevezésük, de működésük azonos. A korábbi modellektől eltérően RGB kimenetük van YUV helyett, illetve szét van bennük választva az EAR és a MIC tüskéje (bár a nyáklapon újra összefutnak, a korábbi működés emulálása miatt).

+2A/+3 kapu áramkör (Amstrad 40077)
Ez már nem ULA, de az Amstrad kapu áramköre egyesíti az ULA a PCF és a HAL chipek működését egyetlen tokban. A korábbi DIP (átmenő furatos) kialakítás helyett ez már QFP (négy oldalú, felületre forrasztott) kivitelű.



Melyik ULA melyik géppel működik?
A 6C001E-7 ULA az összes 48k-s géppel (Issue 1-Issue 6A-ig) működik. A többi az alábbiak szerint használható:

- 5C102E - Issue 1 és 2-es alaplapok (a javító áramkörnek rajta kell lenni)
- 5C112E - Issue 1 alaplapok (javító áramkör nélkül) és Issue 2-esek
- 6C001E-5 és 6-os – Issue 1,2,3 alaplapok

Más párosítások is működhetnek, de a fentiek azok, amik hivatalosan dokumentáltak és a szervizkönyv is megerősít.
A 128k-s és a szürke +2-es gépekben használt 7K010E-5 ULA és az Amstrad 40056 IC-k szabadon, változtatás nélkül felcserélhetők.

ULA helyettesítők
Mivel az ULA-k gyártási folyamata rég meghaladott és több évtizede megszűnt, így hosszú évek óta nem lehet újakhoz hozzájutni. Bebizonyosodott, hogy bonyolult a működést lemásolni, amiatt, hogy a Sinclair teljesen az ULA segítségével biztosította az olyan analóg funkciókat, mint a magnó interfész, vagy a YUV kép kimenet.
Mindezek ellenére a közelmúltban kétféle ULA pótlási megoldás is megjelent.

NebULA
A 6C001E-7 és a korábbi ULA-k foglalatába helyezhető és annak működését tökéletesen emulálja. Phil Ruston és Alessandro Dorigatti készítették.

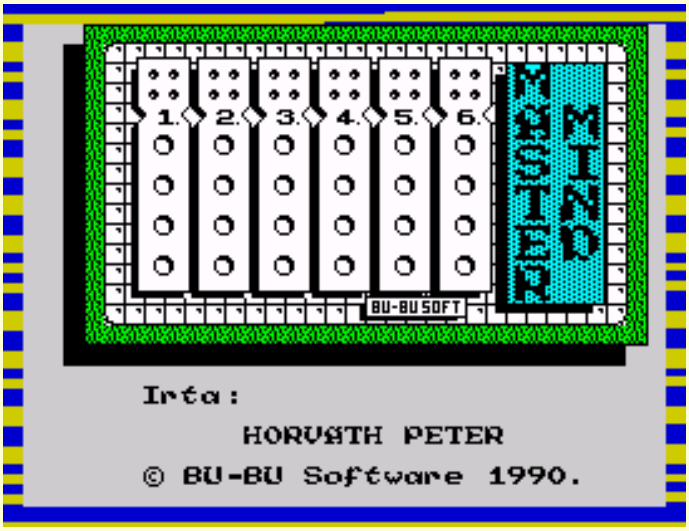
SLAM 128
A 7K010E-5 és az Amstrad 40056 ULA foglalatába egyaránt behelyezhető ez a 128-as és a szürke +2-es gépekhez készült helyettesítés. Mark Smith tervezte és Pjotr Bugaj gyártotta le.

A +2A/B és +3 gépek 40077-es kapuáramköréhez eddig nem készült helyettesítő, de szerencsére ezek az áramkörök nagyon megbízhatóak.

Szilágyi György (Pupos)

A LÁNGÉSZ

Kezdő Spectrum programozók és fantáziájukat vesztett programozás tanárok kedvence a MASTERMIND elnevezésre hallgató játék. Az alap játékban 6 színből négyet kell kitalálni. Hat-nyolc próbálkozásunk van, amelyeket a számítógép fekete és fehér pöttyökkel jutalmaz. Talán nincs olyan értelmes létforma a galaxisban, aki ne játszotta volna még ezt a játékot.
E program egyik megvalósítása a **Bu-Bu Szoftver** terméke. Jobb híján a program készítőjét, azaz saját magamat kérdezem a programról:



- Nagy fantáziára vall, hogy a legalább 100 féle Mastermind verzió mellé sikerült egy 101.-et készítened.
- Köszönöm dicsérő szavaidat. Valóban, keresni kellett egy feladatot, mely megvalósítása előre láthatóan nincs kudarca ítélve. Amint látod – saját magam legnagyobb megdöbbentésére – sikerült egy működő verziót összehozni.

- Mennyi időt áldoztál életedből e rettentő probléma megoldására?
- Nos, a programot 1986-ban kezdtem el készíteni és a jelenlegi formáját 1990-ben érte el. Mielőtt megkérdeznéd, a fejlesztési idő nem négy év volt, mondhatnám, hogy összességében csak kb. 1-2 hetet vett igénybe a programozás. A tervezési idő a feladat kigondolásáig tartott, ezt követte a megvalósítás. Előbb a Beta Basic kiegészítővel készült a program, majd rájöttem, hogy felesleges 3 utasítás miatt betölteni az egész kiegészítést. Így a 3 utasítást helyettesítettem alap BASIC, valamint gépi kódú programrészekkel.

- Elég izgalmas a képernyő betöltés, honnan sikerült a rutint koppintani?
- Látom tisztában vagy Assembly képességeimmel, a rutin a Spectrum Világ egyik számában volt, de ott a képernyőt felülről lefelé töltötte be a program. A betöltő rutint nem, de a képernyő kimentést kellett átírni úgy, hogy a kívánt sorrendbe rakja ki a karaktereket. A képernyő az ART Studio programmal készült. Ha kipróbáltad a játékot, láthattad, hogy maga a játék is a címképernyőn zajlik, így rengeteg PRINT és DRAW utasítást sikerült megspórolni.

- Láttam valami vonatot is a programban, hogy került az oda?
- Egy-két programomba beleépítettem, mert jól néz ki, más funkciója nincs. Egyébként a Laser Basic kiegészítésből „ollóztam” ki. A toplista alatt hallható zene sem az én alkotásom.

- Te olyan gyűjtögető típus vagy!? Még egy szerencsétlen vonatot se tudtál rajzolni?
- ???

- Visszatérve a programra: van toplista is.
- Igen, a toplista automatikusan betöltődik a program indításakor. Ha a játékos neve helyére a „save” szót írod, akkor kimentheted, ha „load”-ot akkor betöltheted a listát. Az „uj” szóra pedig alaphelyzetbe áll a lista, azaz nullázódik. Bár ez még nem jelenti azt, hogy te is képes lennél felkerülni a listára.

- Mit jelentett számodra a program elkészítése?
- Jól elszórakoztam vele, amíg elkészült, majd a családommal játszottunk vele néhány órát. Örültem neki, hogy tudtam csinálni egy működő programot.

- Az imént utaltál rá, hogy létezik több programod is, nem mintha engem nagyon érdekelne, de talán az olvasót…
- Igen, készítettem még egy-kettőt, de neked egyet se fogok megmutatni, mert mindig leszólod a munkámat.

- Végezetül, mondhatnál valamit az irányításról. Mivel lehet tüzelni?
- Grrrrrrr !/=!”+!!%”!%+!%”!

Horváth Péter (Hpeter)

Az 1984 egy menedzserjáték, amiben az Egyesült Királyság 1984-ben megválasztott miniszterelnökeként kell tevékenykednünk mindenki megelégedésére. A játék a brit gazdaság 1983-as valódi adataival kezdődik, innen kell 1999-ig eljutnunk, túlélve három parlamenti ciklust és két választást. Határoznunk kell adókról, segélyekről, fizetésekről, költségvetésről, hitelekről, banki alapkamatról és még rengeteg olyan dologról, amelyek megingathatják az addigi egyensúlyt. Minden év forgatókönyve ugyanaz, ugyanazokat a kérdéseket, táblázatokat kapjuk egymás után. Minden döntés, és nem csak az egyértelműen rossz döntés után megkaphatjuk a be nem adott lemondásunk elfogadását.

Az éves forgatókönyv állandó elemei:

1. **Kormányzati mérleg**

Megtudhatjuk, hogy mennyi a …

- bevételünk a különféle adónemekből
- kiadásunk a támogatásokra, fizetésekre, segélyekre
- a bevételek és kiadások közötti különbség

2. **Főbb gazdasági jellemzők megtekintése:**

- infláció
- munkanélküliség
- GDP
- aranytartalék
- ipari termelés
- kereskedelmi mérleg
- átlagbér
- dollár/font árfolyam

3. **Banki alapkamat megadása**

Az aktuális évre érvényes.

4. **Grafikon**

Kapunk egy grafikont arról, hogy az öt fő terület, amire hatással vagyunk, mennyire van kiegyensúlyozva. Figyelve ezeket az értékeket, megtudjuk, hogy mely területek szenvednek hiányt és mely területek vannak túlfinanszírozva.

A területek:

- a kormányzat
- az ipar, kereskedelem, mezőgazdaság
- lakosság
- bankok, biztosítók
- külföld

5. **Bértárgyalások**

Az ország három fő munkavállalói csoportjának (köztisztviselők, közalkalmazottak és a többi munkavállaló) képviselői járulnak elénk béremelési kéréssel, amire tennünk kell egy béremelési javaslatot. Óvatosak legyünk, mert csak egy ajánlatot tehetünk mindhárom csoportnak, ha az ajánlatunkat nem fogadják el, akkor az általuk igényelt, akár irreális bérkövetelés lesz jóváhagyva. Meg kell próbálnunk a 2. pontnál megadott inflációnak megfelelő szint környékére korlátozni a béremeléseket, különben túl nagy lesz a bérkiáramlás és nő az infláció.

6. **Banki befektetéseink**

Csak annyit fektessünk be, amennyit nagyon muszáj.

7. **Minisztériumok költségvetése**

Minden terület költségvetésénél vegyük figyelembe, hogy az ott dolgozók bérét és a tőkeberuházásokat is fedezni kell. Például ha a bértárgyalásoknál sokat engedtünk a közalkalmazottaknak, akkor biztos sokat kell emelnünk az egészségügyi kereten is. Miután elfogadjuk a költségvetést, a

minisztériumoknak még van lehetőségük reklamációra, amit figyelmen kívül is hagyhatunk, de módosíthatjuk is a költségvetéseket.

A következő „minisztériumok” vannak:

- Társadalombiztosítási
- Hadügyi
- Egészségügyi
- Oktatásügyi
- Kereskedelmi és Ipari
- Szállítási
- Igazságügyi
- Környezetvédelmi
- Lakásügyi
- Egyéb ügyek

8. **Adók, járadékok, segélyek**
Itt beállíthatjuk, hogy mennyi legyen az aktuális adók és járadékok mértéke (egyszerre csak 10 százaléknyi változtatás lehetséges), ezzel pl. jelentősen emelhetjük az állam bevételeit, vállalva az emiatti elégedetlenséget.

9. **Külföld támogatása**
Harmadik világ támogatása, próbáljunk se túl keveset, se túl sokat adni.

10. **Ipari engedmények**
Az ipar és mezőgazdaság támogatására szánt éves összeg.

11. **Évértékelés**
Ha idáig eljutunk, akkor kapunk egy **évértékelést** a gazdasági mutatóink alapján, ami a választások idején válik nagyon fontossá, mert ha mutatók szerint szegényes a teljesítményünk, akkor nem kapunk a kormányzásra több bizalmat.

Év elején még belefuthatunk kabinetgyűlésekbe, ahol általában egyszerű kérdésekre igen-nem válaszokat adhatunk. Pl. a védelmi tanács jelzi, hogy az öregedő eszközöket le szeretné cserélni és szeretne 10% költségvetés-növelést vagy ki kell mentenünk a gazdasági válság miatt csődközelbe jutó bankokat a csávából.

Ha nem jutnánk el az első újraválasztásig sem, nyugodjunk meg, a miniszterelnökség nehéz pálya, ráadásul a szimulált 80-as évekbeli brit közeg sem olyan toleráns, mint a Magyarországon megszokott. Sok esemény, helyzet nagyon aktuálisnak hathat, ami nem a véletlen műve, a programozó, Rob M.H. Carter BASIC-ben megírt munkáját dicséri. A kivitelezés egyszerű, nélkülözi a grafikai finomságokat, de a játékélményt nem rontva, ilyen szempontból talán csak a szándékos lassúság kifogásolható. Zene nincs, hangok is csak elvétve, de nem is hiányoznak, kivéve talán az elejére egy himnusz brit hazafiúi érzelmeinket fokozandó, szimulálva az eskütételünket.

Végezetül ajánlom az általam „fordított” magyar változatot (a pénzügyekhez nem értést és angolul nem tudást lelkesedéssel tudtam csak pótolni).

Mezei Róbert (m/zx)

TUDTAD?

A **Speccyalista** elnevezést az **Úr 1998. évének május havában, annak 5. napján** adta nekünk Pgyuri, amit azóta is lelkesen licenszelünk. Noha a portál indításának gondolata 1997-re datálódik, születésnapunknak mégis ezt a napot tekintjük.

Idén már fennállásunk huszadik évfordulóját ünnepeljük.

Logónkat 2004. szeptemberében Mópi alkotta meg számunkra egy pályázat keretében, amin **a** tagság egyhangúlag az Ő művét választotta a Speccyalista Baráti Kör **jelképének.**

HARDVER SIMOGATÓ

KLAVIA-TÚRA: STONECHIP

44 billentyű és plussz funkciók

Sikerült hozzájutnom egy ilyen elég ritka darabhoz. Az első, amelyiket ebayen láttam, „aranyárban” kelt el. A másodikat sikerült elfogadható áron megcsípnem, így kerülhetett sor a kipróbálására.

Külcsín

A billentyűzeten van pár új billentyű is: egy „Extended” és egy „Delete”, valamint két „Reset” gomb. Az első kettő értelemszerűen a Spectrum két gomb egyidejű megnyomásához kötött funkcióját látja el egyetlen gombbal. Resetből azért van kettő, mert a véletlen reset kiküszöbölésére a bal középen és jobb felül lévő gombok egyidejű megnyomása kell a gép reseteléséhez, ami egyedülálló megoldás az eddig látott Spectrumhoz készített billentyűzetek között, és emellett szerintem praktikus biztonsági megoldás.
Érdekes dolgok találhatók a felső részen, középen egy piros led jelzi a gép bekapcsolt állapotát „Power” felirattal ellátva. Bal oldalon egy háromállású kapcsoló található „Load-Beep-Save” felirattal, értelemszerűen a LOAD/SAVE funkciók leválasztására illetve BEEP állásban a jobb oldalon beépített saját hangszórón szólal meg a zene, mellette egy szintén háromállású kapcsoló „High-Med-Low” felirattal, mely a hangszín kiválasztására szolgál.

Belbecs

Nem csak a ritkasága miatt érdekes ez a klaviatúra, hanem az általam eddig látott Spectrum billentyűzetek között egyedülálló abban is, hogy „szőröstől-bőröstől” (azaz dobozostól) az egész Spectrumot elnyeli, a billentyűzet dobozában kialakított csatlakozók illeszkednek a Speccy minden kimenetére. A beszereléshez tényleg csak a Stonechip csavarjait kell kitekerni, és a gép behelyezése után visszacsavarni és a szerelés ezzel kész is.

A próbadarabba egy fóliahibás gépet tettem, természetesen ez is rendben működik, mert a csatlakozósávon keresztül kommunikál a géppel, nem a fóliacsatlakozót használja.

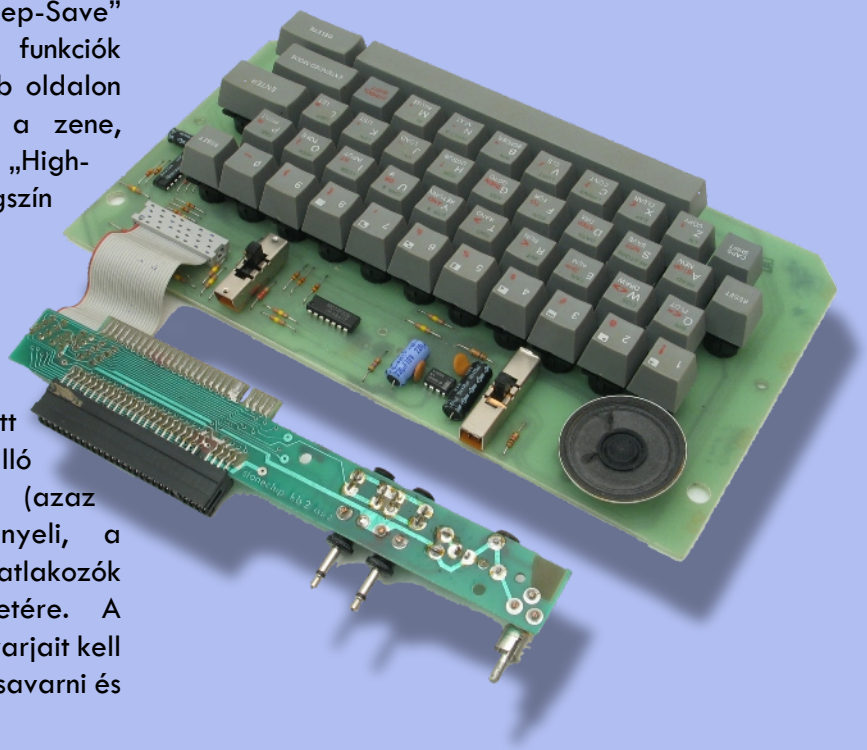
Jó vétel?

A Spectrumhoz készített billentyűzetek közül egyértelműen nem a legjobb vétel, már ha valaki használni is szeretné ezzel kedvenc gépét, mert nem kényelmes rajta gépelni, játszani, bár kétségtelenül praktikus megoldások találhatóak rajta.
Gyűjtői szemmel már sokkal inkább jobb vételnek tűnik, mivel ritka és érdekes darab, ára is természetesen ehhez igazodik.

Summázás

Egyszóval gyűjtőknek elengedhetetlen darab, amúgy csak egy érdekes szerzet.

Makranc



HOGYAN ÍRJUNK JÁTÉKOT ZX SPECTRUMRA - 8. RÉSZ

Sprite-ok

Pixel pozíciók konvertálása képernyő-címekké

Az UDG-k és a karaktergrafika nagyon finom és dandy, de a jobb játékok általában sprite-okat használnak, azonban nincsenek ilyen hasznos ROM-rutinok, amelyek segítenénk nekünk a sprite kezelésben, mert Sir Clive a Spectrumot nem játékgépnek tervezte. Előbb-utóbb egy játék programozónak szembe kell néznie a Spectrum kellemetlen képernyőelrendezésével. Ez egy trükkös vállalkozás, hogy az x és y pixelkoordinátákat egy képernyőcímmé alakítsuk át, de van néhány módszer, amit alkalmazhatunk.

Egy képernyőcím feloldó („look-up") táblázat használata

Az első módszer egy előre kiszámított címtáblázat használata, amely a képernyő címét tartalmazza a Spectrum mind a 192 sorához, mint ez a példa, vagy valami ehhez hasonló változat:

```
xor   a           ; carry flag és az
                    akkumulátor törlése
ld    d, a        ; DE magas bájt törlése
ld    a, (xcoord) ; függőleges pozíció
rla               ; shift balra a kétszeres
                    szorzáshoz
ld    e, a        ; elhelyezése a DE alacsony
                    bájtjában
rl    d           ; felső bit shift a DE magas
                    bájtjába
ld    hl, addtab  ; a képernyőcímek táblája
add   hl, de      ; mutató beállítása a tábla
                    bejegyzésre
ld    e, (hl)     ; a képernyőcím alacsony
                    bájtja
inc   hl          ; mutató beállítása a magas
                    bájtra
ld    d, (hl)     ; a képernyőcím magas bájtja
ld    a, (ycoord) ; vízszintes pozíció
rra               ; osztás kettővel
rra               ; még1x a néggyel való
                    osztáshoz
rra               ; és még1x a nyolccal való
                    osztáshoz
and   31          ; elmaszkoljuk a szemetet
add   a, e        ; hozzáadjuk a sor
                    kezdőcíméhez
ld    e, a        ; új érték betöltése E-be
ret               ; visszatérés a címmel a
                    DE-ben
.
Addtab:
  defw  16384
  defw  16640
  defw  16896
.
```

Ez nagyon gyors megoldás, de azt is jelenti, hogy mind a 192 sor címét tárolni kell egy táblázatban, ami 384 bájtot foglal, ami esetleg máshol hiányozhat.

Képernyőcímek számítása

A második módszer magában foglalja a cím számítását, így nem igényel címfeloldó táblázatot. Ennek során három dolgot kell megvizsgálnunk: a képernyő melyik harmadába esik a pont, a karaktersor, amelyhez legközelebb van, és a pixel vonal, amelyen az adott cellába esik. Az AND operandus megfelelő használata segít eldönteni mindhármat. Ez azonban bonyolult vállalkozás, ezért légy türelemmel, és megpróbálom elmagyarázni, hogyan működik.

Megállapítjuk, hogy a három képernyő szegmens közül melyikben található pont, a függőleges koordináta legkevésbé fontos hat bitjét elmaszkova, hogy 0, 64 vagy 128 értéket maradjon a 64 pixel szegmensek mindegyikénél. Mivel a 3 képernyő szegmenscímek magas bájtjai 64, 72 és 80 (8-as különbség a szegmesek között), ezért a maszkolt értéket vesszük és elosztjuk 8-cal, hogy 0, 8 vagy 16 értéket adjon nekünk. Ezután hozzáadjuk a 64-et, hogy megkapjuk a képernyő szegmensének magas bájtját.

Minden szegmens 8 karakteres cellás pozíciókra oszlik, amelyek 32 bájtra vannak egymástól, így a címünk szempontjából az a lényeg, hogy vegyük a függőleges koordinátát, és maszkoljuk el a két legjelentősebb bitet, - amelyet a szegmens meghatározásához használtunk - a három legkevésbé lényeges bittel együtt. amik a pixel helyzetét határozták meg. Az AND 56 utasítás szépen megcsinálja. Ez megadja a karaktercella pozíciót 8 többszöröseként, és mivel a karaktersorok 32 bájtnyira vannak, ezt 4-gyel meg kell szorozni, majd helyezzük el a kapott számunkat a képernyőcím alacsony bájtjában. Végül a karaktercellák tovább osztódnak 256 bájtos pixel vonalakra, ezért ismét vesszük a függőleges koordinátát, maszkoljuk mindent, kivéve azokat a biteket, amik meghatározzák a sort az AND 7 használatával, az eredményt pedig adjuk a képernyőcím magas bájtjához. Ez megadja a függőleges képernyőcímet.

Vegyük a vízszintes koordinátankat, osszuk el 8-cal és adjuk hozzá a címünkhöz.
Itt ez a rutin, ami visszatér a képernyőcímmel az x, y koordinátákhoz a DE regiszterpárban. Könnyen módosítható, ha a címet inkább a HL vagy BC regiszterekben szeretnénk visszakapni.

```
scadd:
  ld    a, (xcoord) ; függőleges koordináta
                      beolvasása
  ld    e, a        ; eltárolása E-ben

; Megkeressük a cellán belüli sort.

  and   7           ; sor 0-7 a karakteren belül
  add   a, 64       ; 64 * 256 = 16384 = kezdő
                      képernyőcím
  ld    d, a        ; sor * 256

; Megkeressük melyik képernyőharmadban vagyunk.

  ld    a, e        ; függőleges koordináta
                      visszatöltése
  and   192         ; szegmens 0, 1 vagy 2
                      szorzása 64-gyel
  rrca              ; osszuk el ezt 8-cal
  rrca
  rrca              ; végül a szegmens 0-2
                      nyolccal megszorozva
  add   a, d        ; adjuk hozzá D-hez, szegmens
                      kezdőcíme
  ld    d, a

; Megkeressük a karaktercellát a szegmensben.

  ld    a, e        ; 8 karakter négyzet per
                      szegmens
  rlca              ; x osztása 8-cal és szorzása
                      32-vel,
  rlca              ; nettó számítás: szorozzuk
                      meg 4-gyel
  and   224         ; szügségtelen bitek
                      elmaszkolása
  ld    e, a        ; függőleges koordináta
                      számítása kész

; Hozzáadjuk a vízszintes elemet.

  ld    a, (ycoord) ; y koordináta
  rrca              ; csak nyolccal kell osztani
  rrca
  rrca
  and   31          ; 0 - 31 négyzetek a képernyőn
  add   a, e        ; adjuk hozzá eddig
  ld    e, a        ; DE = képernyő címe
  ret
```

Eltolás

Most, hogy kiötlöttük a címet, már csak azon kell elmélkednünk, hogy miként kerül oda a grafikánk. A vízszintes koordináták három legalacsonyabb bitje jelzi, hogy hány képpontos eltolásra van szükségünk. Egy lassabb módja a pixel kirajzolásának, ha felhasználjuk a fenti scadd rutint, AND 7 végrehajtása a vízszintes koordinátán, majd jobbra eltolni egy pixelt zérótól hétszer - az eredménytől függően -, mielőtt a képernyőre kerülne. Az eltoló sprite rutin ugyanígy működik. Egyszerre a grafika egy sorát kiveszi a memóriából, áthelyezi a megfelelő pozícióra, majd mielőtt a következő sorra ugrana, elhelyezi a képernyőn, és ez a folyamat ismétlődik. Inkább írhatnánk egy sprite-rutint, amely kiszámolja a képernyő címét minden vonalra, valójában az első sprite-rutin, amit valaha írtam, ilyen módon működött. Szerencsére sokkal egyszerűbb annak a meghatározása pár utasítással, hogy egy karaktercella belsejében mozgunk-e, karakterkészlet-határokat lépünk át, vagy egy szegmenshatárt lépünk át, és ennek megfelelően növeljük vagy csökkentjük a képernyő címet. Egyszerűen mondjuk ki, hogy az AND 63 nulla értéket ad akkor, amikor az új függőleges pozíció egy szegmenst keresztez, és az AND 7 nulla értéket ad vissza, ha áthalad egy karaktercella határán, és bármi más azt jelenti, hogy az új vonal ugyanazon karaktercellán belül van, mint az előző sor. Ez egy eltoló sprite rutin, amely felhasználja a korábbi *expedd* rutint. Használatához egyszerűen be kell állítani a koordinátákat a dispx és a dispy változókban, mutasson a BC regiszterpárra a sprite grafika, és hívjuk meg a sprite rutint.

```
sprit7:
  xor   7           ; kiegészítjük az utolsó 3
                      bitet
  inc   a           ; adjunk hozzá eyget
sprit3:
  rl    d           ; forgatás balra...
  rl    c           ; ...a középső bájtba...
  rl    e           ; ...és végül a bal karakter
                      cellába
  dec   a           ; eltolások számolása
  jr    nz, sprit3  ; vissza amíg van még eltolás

; Egy sor sprite: E + C + D, ebben a formában
van rá szükségünk C + D + E.

  ld    a, e        ; a kép bal széle most az
                      E-ben van
  ld    e, d        ; tegyük ide a jobb szélét
                      inkább
  ld    d, c        ; a középső bit megy a D-be
  ld    c, a        ; és a bal széle vissza a C-be
  jr    sprit0      ; végeztünk, mehet a
                      képernyőre


sprite:
  ld    a, (dispx)  ; sprite kirajzolása (HL)
  ld    (tmp1), a   ; y mentése
  call  scadd       ; képernyőcím számítása
```

```
  ld    a, 16       ; a sprite magassága pixelben
sprit1:
  ex    af, af'     ; loop számláló mentése
  push  de          ; képernyőcím mentése
  ld    c, (hl)     ; első sprite grafika
  inc   hl          ; sprite mutató növelése
  ld    d, (hl)     ; a sprite következő bitje
  inc   hl          ; lássuk a sprite következő
sorát
  ld    (tmp0), hl  ; mentsük el tmp0-ba
  ld    e, 0        ; jobb bájt törlése
  ld    a, b        ; B-ben van y
  and   7           ; hogyan keresztezzük a
                      karakter cellákat?
  jr    z, sprit0   ; nincs keresztezés, nem
                      zavarja az eltolást
  cp    5           ; 5 vagy több jobb eltolás
                      kell?
  jr    nc, sprit7  ; igen, eltolás balról, mert
                      gyorsabb
  and   a           ; hoppá, carry flag van,
                      töröljük
sprit2:
  rr    c           ; bal bájt forgatása
                      jobbra...
  rr    d           ; ...a középső bájton
                      keresztül...
  rr    e           ; ...a jobb bájtba
  dec   a           ; eggyel kevesebb eltolás van
                      hátra
  jr    nz, sprit2  ; vissza, ha végeztünk

sprit0:
  pop   hl          ; képernyőcím visszatöltése
                      a stack-ből
  ld    a, (hl)     ; ami már itt van
  xor   c           ; képadatok egyesítése
  ld    (hl), a     ; irány a képernyő
  inc   l           ; következő karaktercella
                      jobbra
  ld    a, (hl)     ; ami már itt van
  xor   d           ; egyesítsük a kép középső
                      bitjével
  ld    (hl), a     ; tegyük vissza a képernyőre
  inc   hl          ; a képernyő következő bitje
  ld    a, (hl)     ; ami már itt van
  xor   e           ; a sprite jobb széle
  ld    (hl), a     ; irány a képernyő
  ld    a, (tmp1)   ; y koordináta vissza
  inc   a           ; következő sor
  ld    (tmp1), a   ; új y mentése
  and   63          ; átmentünk a képernyő
                      következő harmadába?
  jr    z, sprit4   ; igen, jöhet a következő
                      szegmenst
  and   7           ; átmentünk az alatta lévő
                      karakter cellába?
  jr    z, sprit5   ; igen, jöhet a következő sor
  dec   hl          ; 2 bájt maradt
  dec   l           ; nem folytonos 256-bájt
                      határ itt
  inc   h           ; következő sor a
                      karaktercellában
sprit6:
  ex    de, hl      ; képernyőcím DE-ben
  ld    hl, (tmp0)  ; sprite aktuális sor címének
                      visszatöltése
  ex    af, af'     ; loop számláló visszatöltése
  dec   a           ; csökkentsük
  jp    nz, sprit1  ; nem értük még el a sprite
                      alját, ismétlés
  ret               ; elértük, végeztünk
sprit4:
  ld    de, 30      ; a következő szegmens 30 bájt
  add   hl, de      ; adjuk hozzá a
                      képernyőcímhez
  jp    sprit6      ; ismételjük
sprit5:
  ld    de, 63774   ; vonjunk le 1762-őt
  add   hl, de      ; 1762 levonása a fizikai
                      képernyőcímből
  jp    sprit6      ; loop
```

Amint látható, ez a rutin a XOR utasítást használja arra, hogy egyesítse a sprite-ot a képernyőre, amely ugyanúgy működik, mint a PRINT OVER 1 a Sinclair BASIC-ban. A sprite összeolvad minden olyan grafikával, amely már jelen van a képernyőn, ami nem feltétlen szép látvány. Egy sprite törléséhez elég, ha csak újból megjelenítjük, és a kép varázslatosan megjavul.

Ha egy sprite-ot akarunk rárajzolni valamire, ami már a képernyőn van, akkor szükségünk van néhány extra rutinra, hasonlóan a fentihez. Az egyikre ahhoz lenne szükség, hogy a grafikát a képernyőn eltárolja egy pufferben, hogy a képernyő egy része visszaállítható legyen, amikor a sprite törlődik. A következő rutin egy sprite maszkot alkalmazna a sprite körüli és mögötti képpontok eltávolítására AND vagy OR-ok segítségével, így a sprite végül megjeleníthető mindennek a tetejére. Egy másik rutinra lesz szükség ahhoz, hogy visszaállítsa a képernyő megfelelő részét a korábbi állapotára, ha a sprite-ot törölni kellene. Azonban ez sok CPU időt vesz igénybe, így az én tanácsom az, hogy a játékod használjon valami kettős pufferelést, más szóval háttér képernyő technikát vagy használj előre-eltolt sprite-okat, amit hamarosan megvitatunk.
Egy másik módszer, amelyet fontolóra vehetsz, amikor a sprite megjelenik a háttértárgyak mögött, egy trükk, amelyet a Haunted House vagy az Egghead in Space játékaimban láthattál. Bár ez a módszer praktikus a 'colour clash' csökkentésére, de jelentős memóriát igényel. Mindkét játékban a 24576-os címen található egy 6k bájtos maszk-képernyő, és a sprite-adatok minden egyes bájta a dummy-képernyő adataival ÉS-előtt, mielőtt a 16384-es címen található fizikai képernyőre kerülne. Mivel a fizikai képernyő és a maszk-képernyő

pontosan 8k távolság volt, a H regiszter 5-ös bitjének átkapcsolásával lehetett átállítani őket.
Ehhez a sprite rutinhoz képest a *sprit0* rutinunk így néz ki:

```
sprit0:
  pop   hl          ; képernyőcím visszatöltése
                      stack-ből
  set   5, h        ; dummy képernyő címe
  ld    a, (hl)     ; ami már itt van
  and   c           ; maszkoljuk el az objektum
                      mögötti részeket
  res   5, h        ; fizikai képernyőcím
  xor   (hl)        ; képadatok egyesítése
  ld    (hl), a     ; tegyük ki a képernyőre
  inc   l           ; következő karaktercella
                      jobbra
  set   5, h        ; dummy képernyő címe
  ld    a, (hl)     ; ami már itt van
  and   d           ; maszkoljunk a kép középső
                      bitjével
  res   5, h        ; fizikai képernyőcím
  xor   (hl)        ; képadatok egyesítése
  ld    (hl), a     ; tegyük vissza a képernyőre
  inc   hl          ; a képernyő következő bitje
  set   5, h        ; dummy képernyő címe
  ld    a, (hl)     ; ami már itt van
  and   e           ; maszkoljuk a sprite jobb
                      szélét
  res   5, h        ; fizikai képernyőcím
  xor   (hl)        ; képadatok egyesítése
  ld    (hl), a     ; irány a képernyő
  ld    a, (tmp1)   ; ideiglenes y koordináta
```

Előre-eltolt sprite-ok

Az eltoló sprite rutin egy nagy hátránya a sebesség. Az összes grafikus adat pozícióba való áthelyezése időt vesz igénybe, és ha a játéknak szüksége van sok mozgó sprite-ra a képernyőn, akkor érdemes inkább az előre-eltolt sprite-okat használni. Ehhez nyolc szeparált másolat szükséges a sprite-ból, minden eltolt pixelpozícióra egy. Ezután egyszerűen csak azt kell meghatározni, hogy mely sprite-ot használjuk a vízszintes elhelyezkedése alapján, és mi a képernyő címe, és már mehet a sprite a képernyőre. Bár ez a módszer sokkal gyorsabb, a memória felhasználás szempontjából sokkal költségesebb. A sima eltoló sprite rutin 32 bájtot igényel egy 16 × 16 képpontos maszkolás nélküli sprite esetén, míg az előre-eltolt sprite rutin 256 bájtot igényel ugyanarra a sprite-ra. A Spectrum játékok írásakor kompromisszumra kell jutnunk a sebesség és a rendelkezésre álló memória között. Általánosságban én azt preferálom, hogy a sprite-jaimat 2 pixel per képkockával mozgatom, ami azt jelenti, hogy a páratlan képpontokhoz nem szükségesek előre eltárolni. Az előre-eltolt sprite-ok még így is 128 bájtnyi értékes RAM-ot igényelnek.

Előfordulhat, hogy nem feltétlenül ugyanazt a sprite képet szeretnénk használni minden előre-eltolt helyzetben. Például, ha egy sprite lábának helyzetét megváltoztatjuk az előre eltolt pozíciók mindegyikében, akkor egy sprite animálható úgy, mintha balról jobbra haladva mozogna a képernyőn. Ne feledjük egyeztetni a karakter lábait az egyes képkockákon mozgó pixelek számával. Ha egy sprite-ot mozgatsz 2 pixelenként képkockánként, akkor fontos, hogy a lábak is 2 képpontot mozogjanak a képkockák között. Ha ennél kevesebb, akkor úgy néz majd ki, mintha korcsolyázna, és az életéért küzdene. :)

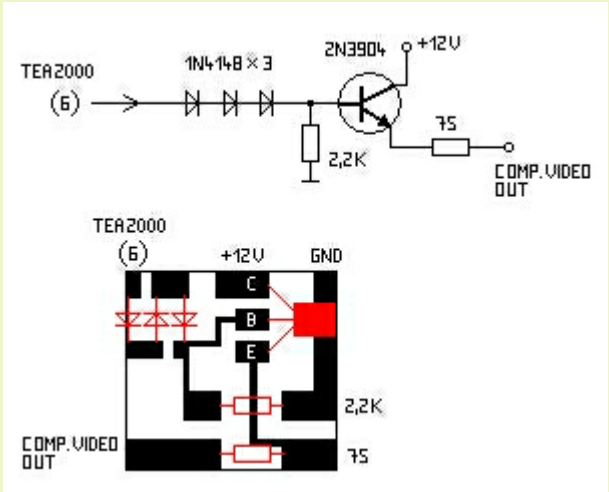
Beavatlak egy kis titkokba: hiszed, vagy sem, de a megfelelő animáció valóban hatással van a játékérzetre.

Fordította:
Kardos Balázs (Balee)

Folytatjuk…

KOMPOZIT VIDEÓ KIMENET ZX SPECTRUM +2A/B/+3-ON

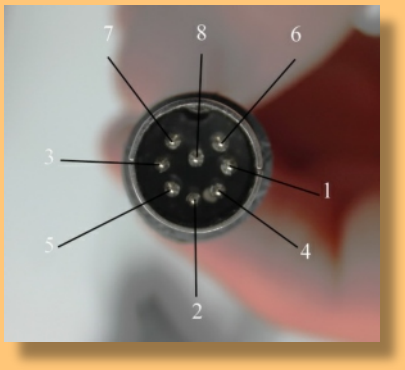
A kapcsolás szinte egy az egyben ugyan az mint a sima 128k-ás Speccyben lévő, csak egy kicsit le egyszerűsítettem. Készítettem hozzá nyákot, ami a TEA2000 tetején helyezkedik el, de annyira kevés alkatrész kell hozzá, hogy anélkül is megépíthető.



Náray József (Njozsef)

ZX SPECTRUM +2 RGB-SCART KÁBEL KÉSZÍTÉSE LCD-HEZ

Mikor megvettem Bali-tól a szürke +2-esét, a képet csak antennakimeneten tudtuk megnézni, mert a géppel hozott kábele nem adott képet LCD monitoron. Megbeszéltük, hogy CRT monitoron volt kép, ezért biztos voltam benne, hogy a kimenet rendben van. Gondoltam megnézem a felépítését a neten és építek egyet, nem lehet nehéz. ☺
A probléma az volt, hogy a netes leírások, mind 8-10 évesek, ezért CRT-kre, vagy korai plazma kijelzőkhöz jók. A különbség amúgy nem vészes, de egyéb kisebb tévedésekkel is találkoztam. Remélem, ez az útmutató segít másoknak is működő kábelt csinálni!



1. ábra: A kábelvégen jelöltem azokat a kimeneteket, amikre a következő táblázatban hivatkozok.

1	Kompozit jel		
2	0V	0V	4,5,9,13,17
3	Bright jel (LK7), Audio jel (LK8)	Audió jel	2,6
4	Kompozit szinkronjel	Horizontális és Vertikális szinkronjel együtt	20
5	Függőleges szinkron jel (LK1), +12V (LK3)	AV átkapcsoláshoz szükséges feszültség	8
6	Zöld	Zöld (150 ohm)	11
7	Piros	Piros (150 ohm)	15
8	Kék	Kék (150 ohm)	7
USB	+5V	Blanking jel	16
USB	-5V	Blanking jel földelése	18

2. ábra: A táblázatban zölddel jelölve, amiket érdemes bekötni.

Magyarázat:

1. Kompozit jelre nincs szükségünk. Ezt több helyen is rosszul jelölik.
2. A 0V-ot az egyik lábbal össze kell kötni, majd az összes többit is egymáshoz.
3. Az alaplapon át kell forrasztani egy jumpert: LK7 legyen nyitva, LK8 legyen zárva.
   A kábelt vagy a 2-es, vagy a 6-os lábra kötni, majd az előzőhöz hasonlóan átkötni a másikra.
4. Erre a videójel átviteléhez van szükségünk! Ha a sima kompozitra kötjük, futni fog a kép.
5. Az alaplapon át kell forrasztani egy másik jumpert: LK1 legyen nyitva, LK3 legyen zárva. Ha át van kötve, akkor az 5-ös lábon megjelenő +12V vezérelheti az automatikus átkapcsolást AV-ra.
6. 150 ohm-os ellenállás kell.
7. 150 ohm-os ellenállás kell.
8. 150 ohm-os ellenállás kell.

USB: A Blanking jelhez 5V-ra van szükség. Ezt én USB kábellel oldottam meg. Az áramot kaphatja egy számítógép USB kimenetéről, de akár egy mezei 5V-os fali töltőről. Az USB kábel levágott végénél mindig a piros a +5V, a fekete a -5V.
FONTOS: Ne felejtsük el a kábel árnyékolását a SCART csatlakozó fém részének oldalsó kivágásához rögzíteni!

3. ábra: A kábel csatlakozói

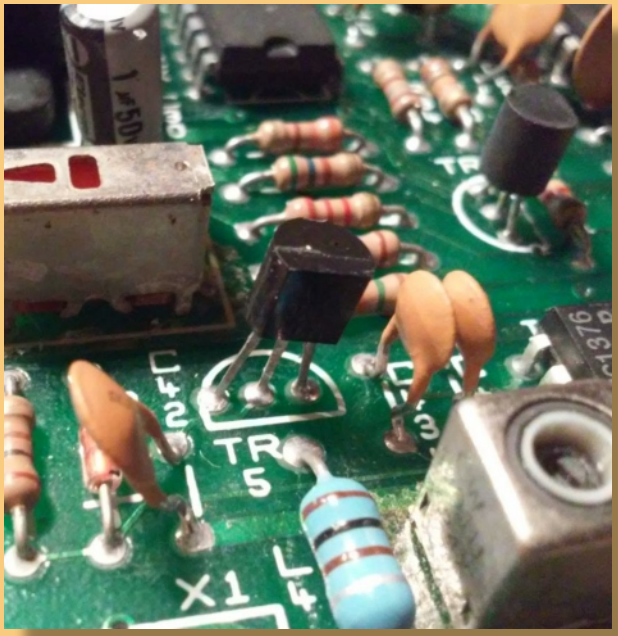
A szükséges alkatrészek:

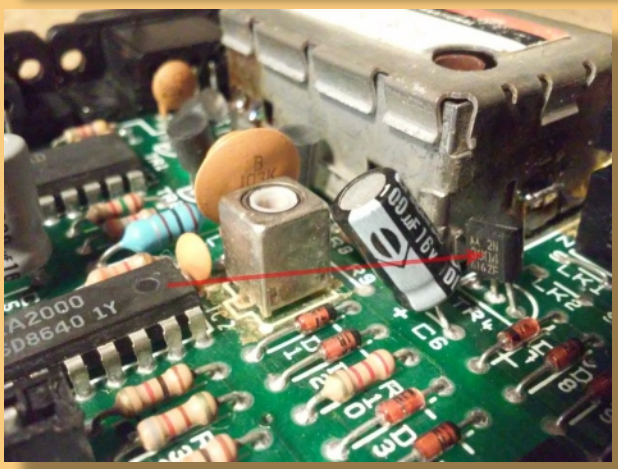
- szerelhető SCART kábel
- 8 pólusú mini DIN csatlakozó
- „USB A" kábeldarab (jó, ha legalább 150 cm hosszú)
- 8 eres kábel (lehet ennél több eres is ☺), inkább hosszú legyen, mint rövid

Sajnos van még egy probléma, amit meg kell oldani. Ha már úgyis szétszedtük a gépet, ellenőrizzük, hogy az alaplapunk milyen sorozatú (azaz Issue számú). Ha 0500 ISS3 akkor valószínűleg két tranzisztort is meg kell fordítanunk.

A 4. ábrán *(lásd a cikk végén)* pirossal bekarikáztam a megcserélendő jumpereket, kékkel pedig a két tranzisztort, amiket meg kell fordítani. A jumpereket mindenképpen cserélni kell, viszont a tranzisztorokat csak akkor fordítsuk meg, ha az alaplapunk 0500 ISS3 **ÉS** a két tranzisztor **2N3904** típusú. Ezek gyárilag fordított polaritással lettek beépítve, mert a

helyükre ZTX tranzisztorok lettek betervezve, de a gyártáskor ezek valamiért nem voltak elérhetők.





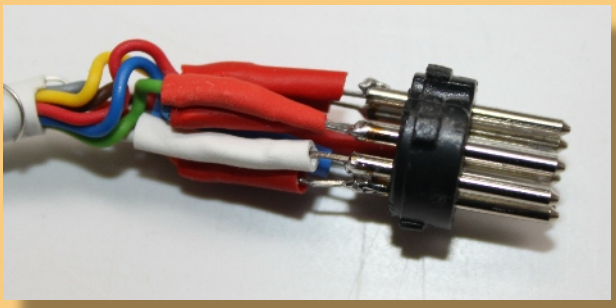
5. ábra: A két tranzisztor, fordított beépítéssel

A szerelésről

A kábel szerelése ügyességet igényel, főleg, mivel a mini DIN csatlakozóba problémás beforrasztani a kábel vezetékeit. Én úgy oldottam meg, hogy ISA csatlakozó tüskéit szedtem ki.

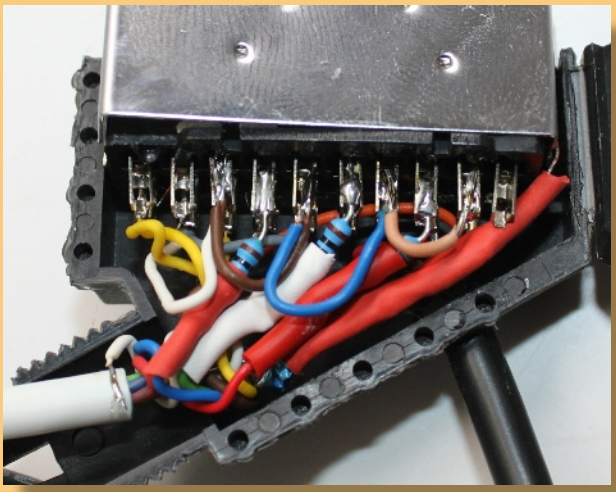
6. ábra: ISA csati tüskéje

A szélesebbik felét óvatosan szétfeszítettem, belefűztem a drótot, amire szigetelődarabot tettem és ráhajlítottam a fémet. Ezután a szúrós felét már könnyű volt beleforrasztani az aprócska mini DIN foglalatba.



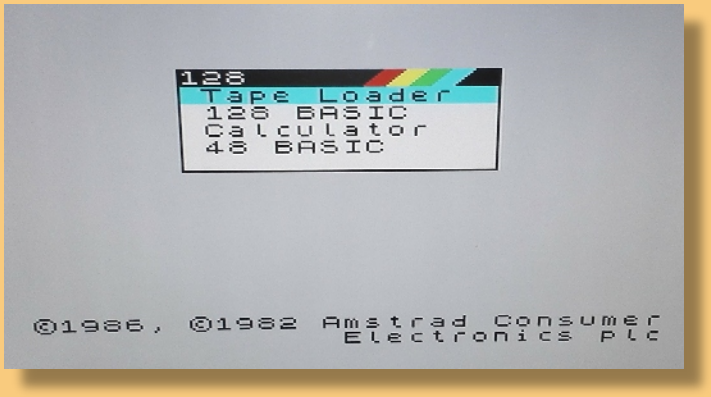
7. ábra: Beforrasztva

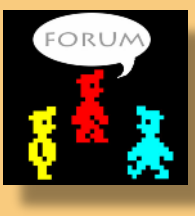
Könnyebb feladat a SCART csatlakozó forrasztása



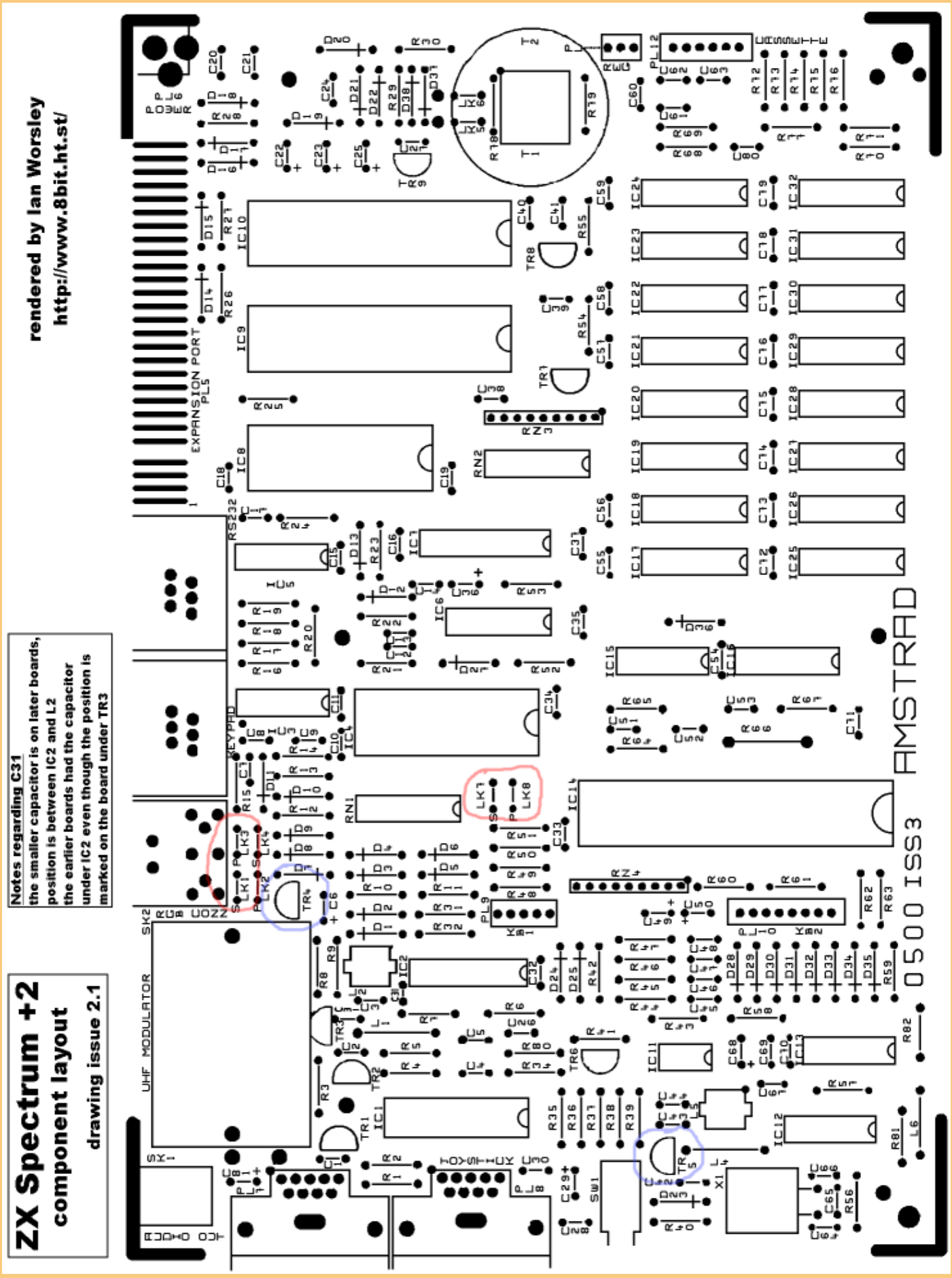
8. ábra: A SCART oldal (avagy a Laokoon csoport ☺)

A képen látható, hogyan oldottam meg az ellenállásokat (csíkos izék), az árnyékolást (jobb oldali piros kábel), illetve a földelést (egymásba, hurokszerűen csatlakozó vezetékek).
Végül lássuk a végeredményt!





Szilágyi György (Pupos)



4. ábra: Szükséges változtatások az alaplapon