

TVC ROM



VIDEOTON

TV-Computer

Ludányi László

A TV-Computer ROM programja

NOVOTRADE RT.
1988

Lektorálta: Beszeda Tamás
Szerkesztette: Stankovics János

A kiadásért felel RENYI GÁBOR,
a Novotrade Rt. vezérigazgatója

Budapest, 1988

ISBN 963 02 5895 1

Copyright (c) Ludányi László, 1988

A következő pár(száz) oldalon megpróbálom Önnel, kedves Olvasó, megosztani azokat a régi élményeimet, amiket az első személyi számítógép megismerésekor éltem át. Kezdve azzal, amikor gyerekes áhítattal még csak rácsodálkozni tudtam a burkolatától megfosztott gép elektronikus paneljére és eljutva addig, amíg megfejtettem a ROM-jába írt programot. Az egymáshoz kapcsolódó fizikai és logikai szintek bejárása közben későbbi világnézetemet is befolyásoló felismerésekre döbentem rá. Először éltem át komolyan, hogy egyszerű építőelemekből hogyan épülhet fel egy magasabb rendű működést megvalósító összetett rendszer.

Sokáig foglalkoztatott az az átmenet, amire akár a legegyszerűbb példa, a felállított dominókból álló kígyósor világít rá. Az elbillenés szituációja és a gravitációs mező összekapcsolja ezeket, s a sok kis dominó egy új rendszerben sajátos funkciójú működésbe kezd.

Vagy másik példa: több ezer gyufaszál és némi ragasztó, meg sok-sok emberi munka, miként alkot egyetlen monumentális palotát. A ragasztó teszi? Vagy a tűzgyújtáshoz való nagy halom gyufaszál?

Természetesen, mindezekben az ember a lényeg. Agyunk legmagasabb rendű tevékenysége, amely képes elvonatkoztatni, túllépni azon, ami van, elképzelni azt, ami lehet, s megalkotni azt, ami lesz!

Ez a gondolat éppen amiatt válhatott a világnézetemet befolyásoló tényezővé, mert szinte mindenütt találkozhatunk vele. S feltétlenül azért, mert éppen a számítógép egyike azoknak a példákknak, ahol ez a gondolat a legnagyobbban és többszörösen megtestesül. A rész és egész, az elem és rendszer kapcsolatának lenyűgöző példáit épp a mikroelektronika és a számítástechnika szolgáltatja.

Ám mindezekben az emberi elme a legcsodálatosabb.

Az alkotás csodája persze eléggé mindennapi csoda. Mégis, talán annak az első számítógépnek tudható be, hogy azóta mindenben keresnem kell a benne átélt csodákat, s nem tudom eléggé értékelni azokat a dolgokat, amelyekből az hiányzik.

Pedig sajnos -- a kedves olvasó is tudja - hogy mennyi logikátlanság, zavarosság van a környezetünkben, ahol az egyes részletek nem úgy kapcsolódnak egymáshoz, ahogy kellene.

Egy számítógépben szigorúak a feltételek. Sok lazaság nem engedhető meg, mert az egész eléggé bonyolult ahhoz, hogy a lazaságok lehetetlenné tegyék a működést. Itt az alapelemek és a végső funkció között igen nagy a logikai szintkülönbség, így aztán minden, látszólag apró részletben tett engedmény magasabb szinten végzetes lehet.

Éppen ez a körülmény, valamint éppen ezeknek a logikai szinteknek a felismerése, egymásra épülésük meglátása jelenti a számítógép tanulmányozásának nagyszerűségét. Azzal együtt, hogy az embert egyszersmind jelentős tanulságok levonására készíti. A számítógépben esetleg többszörösen átélt csoda hatással lesz minden tevékenységünkre, vélemény nyilvánításunkra. És ebben, emögött már ott van a világnézet!

Ebben a könyvben egy kalandra hívom az Olvasót. Végigjárjuk majd azokat a szinteket, amelyek a számítógép működésében egymásra épülve a végsőként tapasztalható működést eredményezik. Egy-egy szinten hosszasan elidőzünk, a számítógép belső világának más tájain pedig megelégszünk egy távlatibb panorámával.

Vannak persze, akik már jól ismerik ezt a világot. Akik nem a megismerés izgalmaira, a megismerhetőség nagyszerűségére vágynak, akiket már csak speciális részletek érdekelnek. Ha Ön ezek közé tartozik, természetesen lapozzon a megfelelő helyre és kívánságom az, hogy mielőbb kapjon választ az Önt érdeklő kérdésre. Elnézését kérem, ha egy-egy esetben ez közvetlenül mégsem sikerül. Ilyenkor arra kérem, hogy a közölt részletekben önállóan elmélyülve, játsszon el kicsit a biztosan adódó lehetőségekkel.

Ön nyilván eldöntötte már, hogy így vagy úgy velem tart, nyissuk fel tehát, képletesen szólva, a TV-Computert és azután: indulás!

A könyv a 64k típusú, 1.2 BASIC verziójú géppel foglalkozik. Más változatú számítógépekben (32k, 64k+, későbbi fejlesztésű BASIC) ettől kisebb-nagyobb eltérések lehetnek. Az ilyen gépek tulajdonosai a könyv konkrét felhasználásakor kellő óvatossággal járjanak el.

1. AZ ISMERKEDÉS ALAPSZINTJE: AMIT A HARDVERRŐL TUDNI KELL

Ön nyilvánvalóan tudja, hogy nem valamiféle programozási alaptankönyvet tart a kezében. Nem akarunk a fizikai működés apró részleteiben sem elmélyedni, így tehát az alapszint valami egészen mást jelent.

A hardverrel foglalkozni sokféleképpen lehet, ám nekünk, felhasználóknak csak annyit kell róla tudni, ami a számítógép működésének megértéséhez és használatához szükséges.

Vizsgálódásunk a hardverről szól, de ezen a szinten teljesen funkcionális szemléletű. Megkérem Önt, kedves olvasó, hogy a következőkben szinte csak vázlatosan közölt fizikai sajátosságokat, az egyes hardver elemek tulajdonságait, a TV-Computerben alkalmazott konkrét megoldásokat, újra és alaposan gondolja át, mert ezekre később sokszor hivatkozunk. Ebben semmi különös nincs, hiszen végső soron az egész működés ezen a szinten realizálódik. Később azonban, már nem szabad, hogy ezek a részletek kössék le a figyelmünket.

A számítógéppel együtt vásárolt "kezelési útmutató" és a "BASIC programozási segédlet" mellett a TV-Computerről szóló sorozatban megjelent "Operációs rendszer" című könyvből szerezhethetünk részletes, a későbbiekben jól hasznosítható ismereteket.

1.1 A mikroprocesszor

A TV-Computer egy Z80-as típusú mikroprocesszorral működik (CPU = Central Processor Unit, központi vezérlő egység).

A számítógép többi elemével való kapcsolatát 13 vezérlőjel, 16 bites cím- és 8 bites adatvezeték biztosítja.

Az alkalmazott órajel-frekvencia 3.125 MHz.

A 16 bites címzés 64 kbyte (65536 byte) memória közvetlen elérését teszi lehetővé.

A számítógép önálló funkciójú eszközeivel és a külvilággal a CPU ún. portok segítségével tartja a kapcsolatot.

Ezek pontos ismerete a gép működésének megértéséhez nélkülözhetetlen.

A Z 80-as CPU 18 db 8 bites és 4 db 16 bites regiszterrel rendelkezik. Ezek a mikroprocesszor saját, belső tárolóegységei. Segítségükkel oldja meg a CPU a memóriában elhelyezett program, valamint a saját aritmetikai és logikai egysége közötti adatmozgatást, ezek tartalmával zajlanak a számítások, logikai és egyéb műveletek.

A regiszterek felsorolását és a velük kapcsolatos néhány fontos tudnivalót a Függelékben adjuk meg.

A CPU teljes utasításkészletének ismertetésétől ugyancsak eltekintünk, viszont a Függelékben összefoglaljuk az utasítások leírására használt mnemonikokat (a műveletekre emlékeztető rövid szimbólumokat) és azok jelentését.

1.2 A memória

A Z 80-as CPU a 16 címvezetékével 65536 cím közül választhat. Ennek megfelelően, főleg a korábban gyártott gépekben igen gyakori volt az ennek pontosan megfelelő, 64 kbyte-os memória. Egyszerű lehetőség van persze arra, hogy a mikroprocesszor 64 kbyte-os horizontján szükség szerint cseréljük magukat a memóriákat.

Így elvileg egy számítógépben akármennyi tárolóelem elhelyezhető, csak arról kell gondoskodni, hogy a CPU mindig az aktuális tartalmú, 64 kbyte-nyi memóriarészt lássa. A közvetlenül címezhető memóriaszeletek cseréjét igen kifejező szóval, memórialapozásnak nevezzük.

Gyakori és a TV-Computerben is alkalmazott megoldás, hogy a teljes memóriát 16 kbyte-os szegmensekre, lapokra osztják, így a memórialapozás a CPU 64 kbyte-os címtartományán belül a 4 elérhető lap cseréjét jelenti.

Idézzük fel a konkrét lehetőségeket!

A TV-Computer 64k jelű gépekben a CPU-val 128 kbyte memóriát kezelhetünk, ami éppen duplája a mikroprocesszor címtartományának.

A címek és adatok írásánál ezután -- az egyszerűbb kezelhetőség és a számábrázolás számítógépben realizálódó formája miatt -- a 16-os számrendszert használjuk.

A CPU 64 kbyte-os címtartományát 4 db 16 kbyte-os lapra osztjuk fel:

- 0. lap: 0000 - 3FFF címek,
- 1. lap: 4000 - 7FFF címek,
- 2. lap: 8000 - BFFF címek,
- 3. lap: C000 - FFFF címek.

A TVC-be (TV-Computer-be) beépített memóriákat is ilyen szegmensekre osztjuk és a következő megkülönböztető nevekkkel látjuk el:

SYS (SYSTEM ROM): a rendszer ROM (csak olvasható memória) fő része.

EXT (extension, kiterjesztés, meghosszabbítási): ebben a 16 kbyte-os szegmensben a felső 8 kbyte a rendszer ROM kiegészítése, alsó 8 kbyte-ján pedig a kiválasztott csatolóártya IOMEM-jét láthatja a CPU.

VID a képmegjelenítésre használt 16 kbyte-os RAM (írható, olvasható memória).

CART a számítógép bal oldalán kialakított csatlakozóba dugaszolható 16 kbyte-os programmodul ROM vagy RAM.

U0...U3 4 db, tetszőleges tartalommal feltölthető, 16 kbyte méretű RAM.

Egy adott pillanatban tehát a mikroprocesszor 64 kbyte-os címtartományába négyet helyezhetünk az alábbi 16 kbyte-os szegmensek közül:

U0 U1 U2 U3 VID CART SYS EXT

A konkrét kiosztást illetően sajnos csak korlátozott lehetőségeink vannak: az U0...U3 RAM-ok csak a sorszámukkal azonos lapon lehetnek, a VID csak a 2. lapon, az EXT csak a 3. lapon, a CART és a SYS pedig csak a 0. vagy a 3. lapon helyezhető el.

A kívánt lapozási konfiguráció a 02-es port írásával állítható be.

Az EXT szegmensben a rendszer ROM folytatását tartalmazó terület mellett, valamelyik csatolóártyán elhelyezett memóriát éri el a mikroprocesszor. Hogy a 4 közül melyiket, azt a 03-as port b7-b6 bitjeinek írásával lehet kiválasztani.

A TV-Computer portjainak funkció szerinti kiosztását a könyv Függelékében megtalálja az Olvasó.

1.3 A képmegjelenítő hardver

A tv kép megjelenítését egy speciális integrált áramkör, a 6845 típusú CRT kontroller (vezérlő) végzi. Az áramkör programozható, 18 belső regisztere van, amelyekbe beírt adatokkal a képmegjelenítés jellemzői tág határok között változtathatók.

A CRT kontroller egy regiszterét a 70H-s port írásával választhatjuk ki. Tartalmának olvasása, vagy írása pedig ezt követően, a 71H-s porton bonyolítható le.

A 2 és 4 színű üzemmódban a képernyőn megjelenő pontok színét a palettaregiszterek határozzák meg. Ezek tartalmát a 60H...63H című portok írásával állíthatjuk be.

A képernyő keretszínét meghatározó ún. borderregiszter írási port címes 00.

A képmegjelenítés eszközeihez tartozik még a TVC hátoldalán található színcapcsoló, amelynek pillanatnyi állapota az 59H-s port olvasása után, a b6 biten érzékelhető.

1.4 A billentyűzet

A TV-Computer billentyűzetének összes nyomógombja, a beépített, valamint a két külső botkormány, egy 10x8-as mátrixba van szervezve. Ez 10 sor- és 8 oszlopvezetékét jelent. Valamelyik gomb lenyomása, vagy egy botkormány működtetése a hozzá tartozó sorvezetékét a neki megfelelő oszlopvezetékekkel köti össze.

A 10 sorvezeték egyikét a mikroprocesszor 03-as port alsó 4 bitjébe írt adattal választja ki és egyben logikai 0 állapotba hozza. Ezután a 8 oszlopvezeték állapota egyszerre olvasható be az 58H-s portról. A beolvasott adatbitek 0 vagy 1 értékéből pedig a kiválasztott sorhoz tartozó 8 billentyű lenyomott vagy felengedett állapotára lehet egyértelműen következtetni.

1.5 A magnetofon kezelése

A TV-Computer és a magnetofonok kapcsolatát egyszerű, kétállapotú áramkörökkel (flip-flop) valósítjuk meg.

Az egyik flip-flop a számítógép magnetofon kimenetére csatlakozva, felvétel esetén magas vagy alacsony szintű feszültséget küld a magnetofon felé. Ellenkező állapotba billentése az 50H-s port írásáival történik.

Tehát csak egyszerű feszültségváltásról van szó, amelyet a port aktiválása vált ki, a portra küldött adat közömbös. Ezeket a feszültségváltozásokat a ROM-ba írt program irányítja úgy, hogy a ráháruló, nem kis bonyolultságú feladatnak megfeleljen.

A memóriatartalom szalagról való visszaolvasása esetén a magnetofonról egymás után érkező feszültségváltozásokat hasonló áramkör dolgozza fel, s a pillanatnyi jelszint most az 59H-s port b5 bitjén jelenik meg. Ezután már a ROM-ba írt program oldja meg a magnetofonról jövőt jelváltások értelmezését.

A megfelelő ROM-lista rész tanulmányozásánál kellő részletességgel megvizsgáljuk majd e rendkívül szellemes és sok szempontból tanulságos mechanizmust.

A TV-Computer két magnetofon távvezérlését is megoldja úgy, hogy a 05H-s port b6-b7. bitjére küldött adat motorvezérlő jeleket juttat a két magnetofoncsatlakozó felé: minden adatmozgatás előtt elindítja, a végén leállítja a motorokat.

1.6 A nyomtató vezérlése

A kiírandó karakter kódját a 01H-s port írásával küldhetjük a CENTRONICS típusú csatlakozó adatvonalaira.

Az adatérvényesítő STROBE jelet a 06H-s port b6 bitjébe 0, majd 1 írásával lehet előállítani, a nyomtató 0 szintű ACK (nyugtázó) jelét pedig az 59H-s port b7 bitjén olvashatjuk be. A két vezérlőjel kölcsönösen alaphelyzetbe állítja egymást, így az adatátvitel során többszörös vagy téves vezérlési helyzet nem állhat elő.

1.7 A soros vonali hardver

A TV-Computer soros vonalának lelke egy 8251A típusú USART (Universal Synchronous-Asynchronous Receiver-Transmitter, általános szinkron-aszinkron üzemmódú adó-vevő) integrált áramkör.

Az USART a CPU oldaláról parancskiküldés vagy állapotolvasás esetén a 11H, 21H, 31H és 41H portcímeiken, míg adatkiküldés vagy beolvasás esetén a 10H, 20H, 30H és 40H címeiken érhető el aszerint, hogy a soros vonali kártya a 0...3. csatlakozók közül melyikbe van dugva.

Az USART programozásával és a soros vonal működésével kapcsolatos részletekről az Olvasó bővebb információkat talál az "Operációs rendszer" c. kézikönyvben.

1.8 A hanggenerátor

A hanggenerátor lényegi része egy 12 bites programozható osztó, amelynek bemenetét 3.125 MHz-es órajel hajtja meg. A mindenkori osztásarány határozza meg a kilépő jel frekvenciáját. A felső 4 bit a 05-ös port b3...b0 bitjeibe, az alsó 8 bit pedig a 04-es portra írva állítható be. A kiküldött 12 bites adat megfelel a BASIC-ből ismert PITCH értéknek (PITCH = osztásarány, de pl. hangmagasság is). Ezt a szimbólumot használva, a 12 bites osztó kimenetén megjelenő jel frekvenciája: $3125/(4096-PITCH)$ kHz (decimális értékek).

Ezt a jelet először felezve kapjuk a soros vonalhoz használt órajelet: $1562500/(4096-PITCH)$ Hz.

Tényleges hangképzés, valamint megszakításkérés céljaira ezt a jelet még egy 8-as osztóra vezetjük, így itt már $195312.5/(4096-PITCH)$ a kilépő jelek frekvenciája. A különbséggel való osztás azt eredményezi, hogy a beírt PITCH értékkel képezett hang frekvenciája nem lineáris: kis értékekkel (mély hangok) egészen finom szabályozás érhető el, ugyanakkor a magas hangok tartománya (nagy PITCH értékek) eléggé durván szabályozható.

Példaként a 0, 1, 1, ill. a 4000, 4001, 4002 PITCH értékekhez tartozó hangfrekvenciák rendre: 47.68 Hz, 47.70 Hz, 47.71 Hz, ill. 2034.51 Hz, 2055.92 Hz és 2077.79 Hz. Látható, hogy a magas hangok csak jelentős kompromisszumokkal állíthatók be.

A 8-as osztó után a hangjel kétfelé ágazik: egy-egy kapcsolóra jut, amelyek egy-egy port egy-egy bitjén érhetőek el.

A hangjel a 05-ös port b4 bitjével letiltható (0) vagy engedélyezhető (1). Az engedélyezett jel egy 4 bites amplitudó- (hangerő-) szabályozó áramkörre jut, amelynek osztása a 06-os port b6...b2 bitjeinek írásával állítható be.

A 05-ös port b5 bitje a hang IT-t érvényesíti vagy tiltja. Amikor azonban a hangjeleket megszakításkérésre használjuk, előtte célszerű -- a pontot időzítés miatt -- a hanggenerátort alaphelyzetbe állítani. Ezt a funkciót az 5BH-s port olvasásával érhetjük el. Ezután az első hangjel (megszakításkérő jel) pontosan a beírt PITCH értékkel meghatározott idő múlva jelenik meg a generátor kimenetén.

Még egyszer hangsúlyozzuk, hogy a számítógép működésének megértéséhez és a ROM-ba írt program tanulmányozásához az előbbiekben összefoglalt ismeretek nélkülözhetetlenek. Ha a kedves Olvasó úgy érzi, hogy részletesebb magyarázatokra is szüksége van, azokat az "Operációs rendszer" c. könyvben találja meg.

2. AZ ISMERKEDÉS KÖVETKEZŐ SZINTJE: ELŐKÉSZÜLETEK

Mostantól elmélyültebb munkába kezdünk.

A későbbi biztonságos eligazodás megkívánja, hogy precíz leltár készüljön a részletekről. Mindenekelőtt alapos előkészületeket teszünk a továbbiakra: megkezdjük a rendszer által használt memóriaterületek benépesítését, megismerjük a fontos munkaterületeket, megkezdjük a fizikai feltételek és a magasabb logikai szinthez tartozó funkciók egymáshoz rendelését; figyelmünket ezután kizárólag a mikroprocesszorra, a memóriára és a portokra irányítjuk.

2.1 A memória eleje és néhány Z 80-as sajátosság

A Z 80-as mikroprocesszor 16 cím-, 8 adat- és 13 vezérlővonalával, az öt kiszolgáló áramkörökkel és tekintélyes utasításkészletével egyszerre el tudja érni a 64 kbyte-os címtartományába helyezett memóriarekeszek bármelyikét, az utasításkészletében levő IN és OUT utasítások révén pedig mindazokat a portokat, amelyek közvetlenül a különböző funkciójú hardver elemekkel létesítenek kapcsolatot.

A CPU az INT (Interrupt, maszkolható megszakításkérés) vezetékére juttatott aktiváló jel hatására -- az ún. 1-es megszakítási módban (IM1) -- befejezi az éppen végrehajtás alatt álló műveletet, majd direkt ugrással a 0038H címen kezdődő utasítássorozatot hajtja végre.

Az egybyte-os szubrutinhívó (RST) utasítások ugyancsak a memória első címeire (00,08,10,18,20,28,30,38 hexadecimális címek) történő ugrást eredményeznek. Ezek használatához is az szükséges, hogy a meghatározott memóriacímeken megfelelő -- többnyire megintcsak speciális -- funkciókat megvalósító utasítássorozatok álljanak.

Mindez azt sugallja, hogy célszerű a memóriának ezt a részét -- tehát a 0000H kezdőcímtől egy kellően nagyméretű területet -- kizárólag ilyen speciális funkciók ellátására biztosítani.

A Z 80-ra épített számítógépek esetében gyakori, hogy éppen az előbbiek miatt, erre a területre eleve ROM-ot helyeznek, így a CPU által is megkülönböztetetten használt fix címeken mindenkor változatlan utasítássorozat áll. (Ami persze nem zárja ki a rugalmas működést, mert ezekre a címekre a RAM-ba mutató ugrási utasításokat írhatunk, az ott elhelyezett rutinok pedig már szabadon átírhatók.)

A TV-Computer esetében ezzel szemben a memória elejére eleve RAM-ot helyezünk. Az előírt címekre így a gép minden bekapcsolásakor be kell írni a szükséges utasítássorozatokat. Az a tény, hogy itt RAM-ot használunk, az egész rendszer működésének rendkívül nagyfokú rugalmasságot biztosít, mivel a kitüntetett címek tartalmát elvileg tetszőleges programokkal írhatjuk át.

A lehetséges memórialapozási konfigurációk ismeretében látható, hogy a TV-Computer működésében az U0 RAM-nak eleve kitüntetett szerepe van.

2.2 Az U0 RAM funkciói

Láttuk, hogy a CPU helyes működéséhez a megszakításkezelő és az RST rutinok belépési címlein a megfelelő rutinoknak kell állniuk. A következőkben látni fogjuk, hogy az U0 RAM-ban a működés alatt levő rendszer számtalan, fontos paraméterét is tárolni kell.

Ezek egy része valóban csak az aktuális rendszerállapotot tükröző értékek tárolására szolgál: ezek a rendszerváltozók.

Más RAM-területeken ugyanakkor komoly munka folyik: adatmozgatások, átrendezések, számolások, az egyes funkciók ellátása közben adódó részeredmények adminisztrálása. Ezek a munkaterületek.

A mikroprocesszor a memóriacímek és a regisztertartalmak átmeneti tárolására és azok visszaállítására ugyancsak megfelelő méretű RAM-területet használ (stack).

Mindezeknek természetesen célszerű ugyancsak az U0-ban helyet adni. Az eddig leírtak teljes részletezésére csak később kerül sor, annyi azonban máris nyilvánvaló, hogy ez a memóriaszegmens az egész rendszer működésében meghatározó jelentőségű információk hordozója lesz.

A ROM-ba írt program -- mint majd tapasztaljuk -- igen nagy gondossággal adminisztrálja ezt a területet, ezért felülírása a felhasználó részéről is csak a legnagyobb körültekintéssel történhet.

A következőkben részletesen leírjuk az U0 ide vonatkozó részének szerkezetét.

Az Olvasó részletes ismertetést talál majd az egyes byte-ok funkciójáról, amelyek minél alaposabb ismerete (de legalábbis megértése) nagyon fontos: sok esetben egyenesen feltétele a ROM-leírás egy-egy fejezete eredményes tanulmányozásának.

A legtöbb fontos címet névvel is ellátjuk, ami lehetővé teszi az ezekre való rövid és egyértelmű hivatkozást.

Viszonylagos bonyolultsága ellenére munkánk mostantól kezdve válik igazán izgalmassá. Olyan terepszemlét tartunk, amelyben az egyes objektumok már közvetlenül a számítógép intelligens működésének alapelemei, bennük a magasan szervezett rendszer mély és csodálatos értelmének nagyon is praktikus nyomaira lelhetünk. A gép logikai értelemben vett agyától még nagyon messze vagyunk, de közvetlenül annak hatókörében mozgunk.

Érdemes -- éppen ezért -- már most odafigyelni arra, hogy az U0 RAM egyes byte-jainak tudatos átírásával, a rendszer egészének működését hogyan mozdíthatjuk el a számunkra célszerűbb funkciók irányába. RAM-ról, átírható memóriáról van szó, tehát elvileg bármilyen eredeti ötlet reálisan keresztülvihető.

2.3 Az U0 RAM eleje

00000-0002 Felhasználatlan. Ez egyúttal azt is jelenti, hogy a TV-Computer nem használja az RST 00 egybyte-os szubrutinhívást.

0003 P-SAVE (paging-save, lapsorszámozás-mentés)

A kívánt memórialapozási konfigurációt - mint tudjuk -- a 02-es portra írt adattal lehet beállítani. Tekintettel azonban arra, hogy meghatározott típusú feladatokat a rendszer is csak megfelelő memórialapozással tud elvégezni -- pl. írás a képernyőre --, a rendszer kénytelen a korábban beállított állapotot időnként elrontani. A feladat elvégzése után a P-SAVE-ben tárolt érték alapján megtörténhet az elrontott lapozási konfiguráció visszaállítása.

0004-0007 Felhasználatlan.

0008 BASIC hibakezelő rutinok belépési pontja. Amikor a ROM-ba írt program fut, a rendszer kiterjedt felügyeletet gyakorol a felhasználó minden tevékenysége fölött.

A tervezett magasabb szintű logikai működést természetesen nem képes felülbírálni, de észreveheti a nyilvánvaló tévedéseket (gépelési hibák, ismeretlen szavak, formai, logikai ellentmondások stb.). Az észlelt hibákat a rendszer sorszámokkal, hibakódokkal különbözteti meg, s a legtöbb esetben azok eredetére utaló szöveg kiírásával tájékoztatja a felhasználót. Néha csak a hibakódot írja ki.

Az RST 08 utáni byte-on meg kell adni a hiba kódját.

A szubrutin:

0008	E1	POP	HL	Elsőként kiemeli a verem tetején levő visszatérési címet, és innen
0009	7E	LD	A, (HL)	beolvassa a hibakódot
000A	B7	OR	A	Beállítja a Z jelzőbitet
000B	E5	PUSH	HL	és visszateszi a címet,
000C	00	NOP		
000D	00	NOP		majd üres utasításokkal
000E	00	NOP		ráfut a másik hibakezelő
000F	00	NOP		ágra (átírható!)

0010 Ez is a hibakezelő belépési pontja, az itt kezdődő rutin az RST 10 utasítással hívható. Olyan esetben használjuk, amikor valami oknál fogva az A regiszterben már eleve a hibakód van. Az előző programrész idáig éppen ezt oldotta meg.

0010	C8	RET	Z	Ha a hibakód 0: visszatér, egyébként pedig
0011	E1	POP	HL	eldobja a címet, majd
0012	C35CFD	JP	FD5C	ugrás a ROM tényleges hibakezelő rutinjára

0015 Ugrási cím az RST 18-cal vezérelt rutinokhoz.

0015	C30000	JP	0000	A 00 byte-ok aktiválás előtt természetesen megfelelő tartalommal töltődnek fel
------	--------	----	------	--

0018 Az erre a címre épített RST hívások logikailag sajátos közbülső nyelvet alkotnak a tiszta gépi kódú vezérlés és a magasabb szintű nyelvek (pl. BASIC) között. Ugyanilyenek az RST 30-ra épített ún. funkcióhívások is. Az RST 18 utasítást követő byte-okon egyszerűen csak bizonyos rutinsorszámokat kell megadni, s az eredmény az adott számhoz tartozó, sokszor meglehetősen bonyolult utasítássorozatból álló feladat elvégzése. Ez az eljárás közel áll az ún. makroassembler logikához, ott is egy rövid elnevezéssel hivatkozunk speciális feladatokat ellátó

szubrutinokra. Az RST 18 után több szimbolikus kódot (rutinsorszámot) is felsorolhatunk, a végét az utolsó byte-ban b7=1 beállítása jelzi. (Tehát pl. a 05 sorszám helyett 85H-t kell írni.) Maga a szubrutin egyetlen ugrási utasítást tartalmaz az U0 RAM-ban:

```
0018 C382FB    JP    FB82          Ezen a ROM-címen kezdődő
                                program értelmezi és hajtja
                                végre a szimbolikus kóddal
                                kijelölt feladatot
```

001B Szubrutinként hívhatjuk ezt a programrészt. Egy funkcióhívást bonyolít le, amelynek kódját az A regiszter tartalmazza. (A funkcióhívásokról később részletesebben szó lesz.) Itt -- és a továbbiakban mindig -- az RST 30-hoz kapcsolódó funkciókódot a mnemonikját (utasítás szimbólum) követő kettőspont után írjuk.

```
001B 321F00    LD    (001F),A    A funkciókódot a 001FH címre,
001E F7nn     RST  30: nn      tehát az RST 30 kódját (F7)
0020 C9       RET                                követő byte-ra teszi és kész
```

A következő byte-okon az EXT0...EXT6 BASIC utasításokhoz tartozó gépi kódú szubrutinok kezdőcímeit tároljuk. Az EXTn kényelmes lehetőséget kínál a felhasználó gépi kódú rutinjainak BASIC-ből való indítására.

```
0021-002E    USR-TAB        (felhasználói szubrutin címek táblája)
0021-0022    EXT0
0023-0024    EXT1
0025-0026    EXT2
0027-0028    EXT3
0029-002A    EXT4
002B-002C    EXT5
002D-002E    EXT6
```

002F üres, a felhasználó rendelkezésére áll.

0030 A funkcióhívások belépési pontja.

Még mindig nem mondunk róla semmit -- a megfelelő helyen úgyis kimerítő részletességgel kell majd foglalkozni vele -- maga a belépési pont amúgyis csak egyetlen ugrási utasítást tartalmaz:

```
0030 C3230B    JP    0B23          Ugrás a 0B23 RAM-címre
```

0033-0037 Felhasználatlan 5 byte

0038 A rendszer megszakításkezelő alprogramjának belépési pontja. Valahányszor a CPU az INT vezetéken vezérlőimpulzust kap, erre a címre ugrik.

A belépési pont most is csak előkészíti a tényleges működést:

0038 F5	PUSH	AF	A két regiszter aktuális
0039 3E70			értékét a verembe menti,
003B D302	LD	A,70	U0-U1-U2-SYS memórialapozást
003D C312C4	OUT	(02),A	állít be, majd ugrás a ROM
	JP	C412	megszakításkezelő
			alprogramjára

0040-006F A bővítő csatolókarták (interface-ek) azonosítói
0070-009F és input-output (bemeneti-kimeneti) pufferei
00A0-00CF (átmeneti tárolói) számára fenntartott 4 x 30H
00D0-00FF méretű terület (0...3. csatlakozó sorrendben).

A rendszer inicializálása (a bekapcsolást követően a működési jellemzők első beállítása, s egy sor, a rendszer beindításával kapcsolatos, később részletezett tennivaló) során a rendszer megvizsgálja a 4 csatlakozó állapotát. Ez az 5AH-s port beolvasásával történik, ahol b7-b6 a 3., b5-b4 a 2., b3-b2 az 1. és b1-b0 a 0. csatlakozó állapotát jellemzi. A bitpárok értéke alapján azonosítható a csatolókartya típusa: 00 -- soros vonal (RS232); 01 -- videojáték-modul azonosítására tervezték; 10 -- floppycsatoló; 11 -- üres vagy azonosítatlan.

Azonosított kartya esetén kerül sor a megfelelő címtartomány feltöltésére, a következők szerint.

Az első byte-ra az azonosító név hossza kerül.

A következő -- legfeljebb 6 byte-on -- a név karakterei találhatóak, ASCII kódban.

A 8. byte az ún. egységszám: több, azonos típusú kartya esetén 00...03 lehet, egy-egy kartya esetén: 00.

A többi - csatolónként 28H db - byte pufferként (átmeneti tárolóhely) szolgál az input-output műveletekhez.

0100-06FF ASCII-kódú képernyő-másolat az Editor számára.

Később részletesen megismerkedünk a TV-Computer full screen editorának (teljes képernyős szövegszerkesztő) működésével. Ez nem más, mint egy összetett programcsomag, amely a tv képernyőjén megjelenő információk, elsősorban szövegek kezelésének, javításának hatékony eszköze. Közvetlenül elsősorban a BASIC programok gyors javítására, fejlesztésére használható. Az Editor a képernyőn egyidőben látható maximális terjedelmű szöveget ezen az ASCII kódú másolaton tartja nyilván, változásait itt adminisztrálja.

Mérete ennek megfelelően -- a 2-es üzemmód legnagyobb területigényét is kielégítve -- 24 x 64 = 1536 byte.

0700-073F Szabad terület. Mérete figyelemreméltó

0740-0AFF A definiálható karakterek mátrixa.

A képernyőn megjelenő karakterek képe 10 sorban és soronként 8 pontból áll össze. A 8 pont egyetlen byte 8 bitjén, tehát egy karakter 10 egymást követő byte-on ábrázolható. Célszerű az alapszínű (PAPER) pontokat 0 értékű bittel, az írandó pontokat (INK) 1 értékkel reprezentálni. Az így kapott byte-ok értelmezését a ROM karaktermegjelenítő rutinjai megfelelően feldolgozzák -- a 4 és 16 színű üzemmódban pedig a szélesebb karakterek megjelenítéséhez a szükséges átalakításokat is elvégzik.

Két példán mutatjuk meg a karakterek fent leírt definícióját: az egyik a normál á, a másik a billentyűzet állapotát jelző inverz C. A tárolt byte-ok és a bitekkel ábrázolt karakterkép a következő:

```

08 00001000 . . . . # . . .   FF 11111111 # # # # # # # #
08 00001000 . . . . # . . .   C3 11000011 # # . . . . # #
08 00001000 . . . . # . . .   99 10011001 # . . # # . . #
3C 00111100 . . # # # # . .   9F 10011111 # . . # # # # #
06 00000110 . . . . . # # .   9F 10011111 # . . # # # # #
3E 00111110 . . # # # # # .   9F 10011111 # . . # # # # #
66 01100110 . # # . . # # .   99 10011001 # . . # # . . #
3E 00111110 . . # # # # # .   C3 11000011 # # . . . . # #
00 00000000 . . . . . . . .   FF 11111111 # # # # # # # #
00 00000000 . . . . . . . .   FF 11111111 # # # # # # # #

```

A 0740-0AFF memóriaterületet lefoglaló 960 byte -- mivel RAM-ról van szó -- az átírható karakterek pontmátrixait tartalmazza. Ezek a TV-Computer esetében a 80H-DFH ASCII kódú (decimálisan 128-223) karakterek. Közölük az első 32 (decimális) az inicializálás során az ékezetes nagy- és kisbetűk, valamint az ún. félgrafikus karakterek mátrixával töltődik fel. Természetesen ezek is átírhatók, de nem célszerű.

A következőkben ezen karakterek 10 byte-os definícióját és ASCII kódját adjuk meg.

ASCII kód	Cím	Karakter	Definíció
80H (128)	0740	À	08 08 3E 6B 63 7F 63 63 00 00
81H (129)	074A	É	08 08 7E 68 60 7C 60 7E 00 00
82H (130)	0754	Ï	08 08 3C 18 18 18 18 3C 00 00
83H (131)	075E	Ù	08 08 3E 6B 63 63 63 3E 00 00
84H (132)	0768	Ö	14 00 3E 63 63 63 63 3E 00 00
85H (133)	0772	ó	14 14 3E 63 63 63 63 3E 00 00
86H (134)	077C	Û	08 08 6B 63 63 63 63 3E 00 00

ASCII kód	Cím	Karakter	Definíció
87H (135)	0786	ü	14 00 63 63 63 63 63 3E 00 00
88H (136)	0790	ű	14 14 77 63 63 63 63 3E 00 00
89H (137)	079A	graf	00 00 00 00 1F 1F 18 18 18 18
8AH (138)	07A4	graf	18 18 18 18 1F 1F 00 00 00 00
8BH (139)	07AE	graf	18 18 18 18 18 18 18 18 18 18
8CH (140)	07B8	graf	18 18 18 18 1F 1F 18 18 18 18
8DH (141)	07C2	graf	00 00 00 00 FF FF 18 18 18 18
8EH (142)	07CC	graf	18 18 18 18 FF FF 18 18 18 18
8FH (143)	07D6	inv. A	FF E7 C3 99 99 81 99 99 FF FF
90H (144)	07E0	á	08 08 08 3C 06 3E 66 3E 00 00
91H (145)	07EA	é	08 08 08 3C 66 7E 60 3C 00 00
92H (146)	07F4	í	08 08 00 38 18 18 18 3C 00 00
93H (147)	07FE	ó	08 08 08 3C 66 66 66 3C 00 00
94H (148)	0808	ö	00 24 00 3C 66 66 66 3C 00 00
95H (149)	0812	ő	24 24 00 3C 66 66 66 3C 00 00
96H (150)	081C	ú	08 08 08 66 66 66 66 3E 00 00
97H (151)	0826	ű	00 24 00 66 66 66 66 3E 00 00
98H (152)	0830	û	24 24 00 66 66 66 66 3E 00 00
99H (153)	083A	graf	00 00 00 00 F8 F8 18 18 18 18
9AH (154)	0844	graf	18 18 18 18 F8 F8 00 00 00 00
9BH (155)	084E	graf	00 00 00 00 FF FF 00 00 00 00
9CH (156)	0858	graf	18 18 18 18 F8 F8 18 18 18 18
9DH (157)	0862	graf	18 18 18 18 FF FF 00 00 00 00
9EH (158)	086C	inv. C	FF C3 99 9F 9F 9F 99 C3 FF FF
9FH (159)	0876	inv. S	FF C3 99 9F C3 F9 99 C3 FF FF

Ezután -- a 0880-0AFF területen -- 64 db 00 áll, ezek tetszőleges tartalommal feltölthetők, s a megfelelő karakterek az A0H-DFH (160-223) ASCII kódokkal megjeleníthetők. Csak a kódok és a megfelelő 10 byte-os mátrixok kezdőcímét soroljuk fel:

ASCII	+0	+1	+2	+3	+4	+5	+6	+7
A0	0880	088A	0894	089E	08A8	08B2	08BC	08C6
A8	08D0	08DA	08E4	08EE	08F8	0902	090C	0916
B0	0920	092A	0934	093E	0948	0952	095C	0966
B8	0970	097A	0984	098E	0998	09A2	09AC	09B6
C0	09C0	09CA	09D4	09DE	09E8	09F2	09FC	0A06
C8	0A10	0A1A	0A24	0AZE	0A38	0A42	0A4C	0A56
D0	0A60	0A6A	0A74	0A7E	0A88	0A92	0A9C	0AA6
D8	0AB0	0ABA	0AC4	0ACE	0AD8	0AE2	0AEC	0AF6

Az U0 RAM további -- a rendszer által lekötött része -- alapvető fontosságú táblázatokat, változóparamétereket, munkaterületeket tartalmaz. Ezek mindenkori értékei és állapota döntően befolyásolja a számítógép egész működését. Értékei átírásánál rendkívüli figyelemmel kell eljárni, mert sokszor a mégoly ártatlannak tűnő változtatások is, amennyiben következményeit nem gondoljuk végig, kritikus

esetben a rendszer működésének teljes összeomlásához vezethetnek. Új, még nem kitapasztalt változtatások előtt feltétlenül javasolt a gépben levő felhasználói programok biztonságos tárolása. Ez persze az U0 RAM eddig tárgyalt részeire is -- kevés kivétellel -- igaz.

2.4 A funkcióhívásokról

A hardverről szóló fejezetben a funkciók szempontjából csoportosítottuk azokat a fizikai eszközöket, amelyek a számítógép működése szempontjából a legfontosabbak. Ott minden esetben beszéltünk arról is, hogy ezek az eszközök a CPU oldaláról hogyan érhetők el. Megadtuk azokat a port címeket, amelyek írásával - olvasásával a mikroprocesszor a szóban forgó eszközöket kezelni tudja.

Ez az első lépés a jól kézben tartható, magasabb szintű működéshez: a számítógépet együttesen megvalósító, legkülönbözőbb funkciójú és fizikai tulajdonságú eszközöket egyetlen irányító szerv, a mikroprocesszor fennhatósága alá vonjuk.

Az egyes eszközökkel kapcsolatos feladatok azonban sohasem egészen elemi formában jelentkeznek. Ezenkívül jórészt függetlenek is a magasabb szinten megfogalmazódó feladatoktól. Ha pl. egyetlen betűt kell a képernyő adott helyére írni, ez a feladat a video-RAM néhány byte-jának megfelelő tartalommal való feltöltését jelenti. Ezt a CPU valamekkora utasítássorozat végrehajtásával el tudja intézni. Maga a feladat azonban teljesen független attól, hogy a kérdéses betűt miért kell egyáltalán a képernyőre írni. Minden kiírási feladatnál ugyanazt az eljárást kell megismételni.

Ezeket a feladatokat, ill. a nekik megfelel utasítássorozatot egyszer kell csak összeállítani. Ha az összes működtetett fizikai eszköz funkcióit a tervezett felhasználás minden igényét kielégítő részfeladatokra osztjuk és ezek megvalósítását egyszer aprólékosan kidolgozzuk, akkor ezzel olyan -- alacsony szintű -- rutincsomaghoz jutunk, amely a későbbi felhasználást nagymértékben leegyszerűsíti.

Ezek a rutinok újabb szintet képviselnek a gép fizikai működésétől az emberméretű alkalmazási feladatok megvalósításához vezető úton.

Érdemes a továbbiakban az eszköz eddigi értelmezését is kiterjeszteni. Tulajdonképpen közömbös, hogy egy alacsony szintű rutin ténylegesen egy fizikai eszközt kezel-e. Felhasználói szempontból inkább az elhatárolt funkción, ill. ilyen funkciók logikailag összetartozó csoportján van a hangsúly. Így jutunk a logikai eszköz fogalmához. Ez azt jelenti, hogy hozzá nem tartozik

okvetlenül valamilyen konkrét hardver, de a vele kapcsolatos funkció mégis bizonyos számú alacsony szintű rutin közreműködésével valósul meg. A TV-Computerben ilyen logikai eszköz pl. az Editor (képernyőszerkesztő). Ezentúl az eszköz szót ilyen, általánosított értelemben használjuk.

Aszerint, hogy egy parancs a megnevezett eszköz számára küld-e feldolgozandó információt vagy fordítva: a parancs az eszköztől várja az adatot -- output (kiviteli) és input (bemeneti) műveletekről beszélünk. Némely eszköz mindkét irányú, műveletre képes (pl. magnetofon, editor) mások csak az inputra (billentyűzet), vagy az outputra (nyomtató).

A TV-Computerben minden eszközhöz hozzá van rendelve bizonyos számú olyan szubrutin, amely a gyakran előforduló feladatokat önállóan megoldja. Ha sorra vesszük a lehetséges eszközöket: képmegjelenítő -- billentyűzet - editor -- hang - nyomtató -- magnetofon -- csatolókárttyák (külső eszközökhöz vezető interface áramkörök), akkor láthatjuk, hogy az egy-egy adat vagy egész adatblokkok feldolgozására vonatkozó, az egyik eszközre kiadott parancs a másik eszköz számára is végrehajthatóan értelmes lehet.

Ez az eszközöket illetően további általánosításra, kezelésüket tekintve pedig szokatlanul gazdag és praktikus megoldásokra ad lehetőséget.

Az eszköz fogalmán túllépve, ún. funkcióosztályokról beszélünk. Azt mondjuk, hogy az egy-egy feladat elvégzésére kiadott parancsainkat ezekhez a funkcióosztályokhoz -- és nem az eszközökhöz -- intézzük. A funkcióosztályokat ugyanazokkal a nevekkel látjuk el, mint az eszközöket, azonban ezeket elvileg szabadon egymáshoz rendelhetjük. Így egy adott funkcióosztálynak küldött parancsot egy másik eszköz alacsony szintű rutinjai is végrehajthatják -- természetesen az általuk kiszolgált eszközön. Mindehhez, mint látni fogjuk, elég egyetlen byte tartalmának átírása a megfelelő hozzárendelési táblázatban.

Másképpen kifejezve arról van tehát szó, hogy az eszközök és a felhasználó közé egy újabb láncszemet iktatunk be. Ez egyben újabb logikai szintet is jelent, amelyek egymáshoz -- a magasabbról az alacsonyabb szintek felé -- a következőképpen kapcsolódnak: feladat -- funkcióosztály -- a feladat megvalósítását a hozzárendelt eszközön megvalósító szubrutin -- mikroprocesszor -- eszköz. Talán mondani sem kell, hogy ez a struktúra a felhasználó számára mennyire dinamikus lehetőségeket biztosít a számítógép input-output eszközeinek kezelésére.

Az eszközök és a funkcióosztályok egymáshoz rendelését minden pillanatban a következő táblázat határozza meg.

0B00-0B0F Input-output hozzárendelés tábla

Funkció- osztály	I n p u t			O u t p u t		
	Cím	Kiszolga- ló eszköz		Cím	Kiszolga- ló eszköz	
0. Video	0B00	FF -		0B08	00 video	
1. Billentyűzet	0B01	01 bill.		0B09	FF -	
2. Editor	0B02	02 edit.		0B0A	02 edit.	
3. Hang	0B03	FF -		0B0B	FF -	
4. Nyomtató	0B04	FF -		0B0C	04 nyom.	
5. Magnetofon	0B05	05 magn.		0B0D	05 magn.	
6. Csatolóártya	0B06	06 cstl.		0B0E	06 cstl.	
7. Csatlakozó kiválasztás	0B07	FF v. ssz.		0B0F	FF v. ssz.	

A táblázat adatainak átírásával az egyes funkcióosztályokhoz más eszközöket rendelhetünk hozzá. Ha pl. a 0B08H címre 04-et írunk, akkor a videoosztályhoz küldött parancsokat a nyomtató fogja végrehajtani.

A funkcióhívásokról további részleteket talál az Olvasó az "Operációs rendszer" c. kézikönyvben.

A funkcióhívás azonosítása az RST 30, egybyte-os szubrutinhívó utasítással történik. Az RST 30-at követő egyetlen byte-on kell megadnunk a funkciókódot, mely meghatározza az előírt művelet irányát, a funkcióosztályt és a kért funkciót végrehajtó szubrutin egyszerű sorszámát, a következők szerint:

b7	b6 --- b5 --- b4	b3 --- b2 --- b1 --- b0
Átvitel	Funkcióosztály	Meghívott, végrehajtan-
iránya	sorszama	dó szubrutin sorszama

(Listáinkban a nagyfokú összetartozás hangsúlyozására az utasítás mnemonikja után, kettősponttal elválasztva adjuk meg a funkciókódot, pl.: RST 30: 22)

A funkcióosztály -- eszköz hozzárendelések megváltoztatásának alapja természetesen az, hogy az adatátvitellel kapcsolatos parancsok mindegyik eszköz számára egyformán végrehajthatóak. Az első 3 funkció azonos valamennyi eszköz esetén; a 0. sorszámú rutinokat a megszakításkezelő alprogram hívja, az 1. számú szubrutinok az egy karakter feldolgozásával kapcsolatos parancsokat dolgozzák fel, míg a 2. számú rutinok több karakterből álló blokkok átvitelét bonyolítják le.

2.5 A rendszerváltozók

Folytatjuk az U0 további részeinek vizsgálatát. Az input-output hozzárendelési táblázat után alapvető fontosságú rendszerváltozók következnek.

0B10 INT-DES, a kurzor- vagy hang-IT által kiszolgálható eszközök. A CPU felé megszakításkérő jeleket a képmegjelenítést vezérlő áramkör (CRT kontroller) kurzor-kimenete, valamint a hanggenerátor küldhet. A megszakításkérést követően lefutó alprogram annak megfelelően hajtja végre az egyes eszközök erre a célra szolgáló rutinjait, hogy az INT-DES rendszerváltozóban a hozzá tartozó bit értéke 0 vagy 1 (0 az aktív érték). A bitkiosztás a következő:

b7	b6	b5	b4	b3	b2	b1	b0
3.	2.	1.	0.	hang	edi-	bil-	vi-
cstl.	cstl.	cstl.	cstl.		tor	lentyű	deo

0B11 PORT03, a 03-as output port kópiája.

Azt az értéket tárolja, amelyet utoljára erre a portra küldtünk. Több ilyen rendszerváltozó is van, a legelső példát a 0003 cím P-SAVE nevű változója adta, ahol az aktuális lapozási konfigurációt tároljuk.

Azonkívül, hogy esetenként jó tudni az egyes portokra kiküldött aktuális értékeket, s e változókból bármikor kényelmesen visszaolvasható, az ilyen rendszerváltozóknak más, komolyabb szerepe is van.

A 03-as portra küldött adat bitjeinek jelentése:

b7 - b6	b5-b4	b3 - b2 - b1 - b0
EXT belapozása esetén		a billentyűzet beolvasásához a 0...9 sor
a 0...3. IOMEM	- -	egyikének kiválasztása
egyikének kiválasztása		egyikének kiválasztása

Látható, hogy a b7-b6, ill. b3...b0 bitek teljesen más funkciókhoz tartoznak. Emiatt pl. a billentyűzet sorkiválasztása esetén csakis a b3...b0 biteket szabad átírni, a b7-b6 beállított értékét változatlanul kell hagyni, és fordítva. Ez pedig csak úgy hajtható végre, ha valahol rendelkezésre áll a portra korábban beállított érték teljes másolata.

Amennyiben pl. a billentyűzet 5. sorát kell kiválasztani -- az IOMEM kiválasztás beállított értékének megváltoztatása nélkül -- ez a PORT03 felhasználásával a következőképpen hajtható végre:

```

LD C,05      Betöltjük -- pl. a C regiszterbe a
              beállításra kerülő biteknek megfelelő
              adatot,
LD A,(0B11)  az A regiszterbe pedig a 03-as port
              másolatát, amely az IOMEM kiválasztó biteket
              helyesen tartalmazza.
AND F0       Az 11110000 bitekkel való AND (logikai és)
              művelet a felső 4 bitet változatlanul
              hagyja, tehát az IOMEM kiválasztó bitek
              megmaradnak, az alsó 4 bit értéke pedig 0
              lesz.
OR C         Az OR (logikai vagy) hatására a C-beli 1
              bitek az A-ba másolódnak, így ezután az A
              felső bitjei
              ugyanazok lesznek, mint korábban, míg az
              alsó bitek a megváltoztatott állapotot
              mutatják.
LD (0B11),A  Ez lesz a 03-as port új kópiája,
OUT (03),A   s ez az adat ki is küldhető

```

Ezzel a technikával -- vagy más, vele egyenértékű megoldással -- a ROM-lista sok helyén találkozhatunk.

0B12 PORT05, a 05-ös port kópiája.

Ennek a portnak a bitjei négyféle funkcióhoz tartoznak, így az egyes bitek változtatása esetén gondoskodni kell a többi bit korábbi értékének megőrzéséről. A 05-ös output port bitjei:

b7	b6	b5	b4	b3 - b2 - b1 - b0
Magnetofon moto-		hang	hang	hangfrekvencia, PITCH
rok vezérlése		IT	jel	felső 4 bit
jobb	bal			
0: ki		0:tilt.	0:tilt.	
1: be		1:eng.	1:eng.	

0B13 PORT06, a 06-os port kópiája.

Ez a port 3 különböző funkcióhoz tartozó biteket tartalmaz. A port írása ennek figyelembevételével történhet. A 06-os port bitjei:

b7	b6	b5 - b4 - b3 - b2	b1 - b0
0 szintű	-	a hangjel	üzemmód:
adatérvénye-		amplitudója	00: 2 szín
sító impulzus			01: 4 szín
a nyomtatónak			10:16 szín
(STROBE)			11:16 szín

0B14 Ez a rendszerváltozó egyszerű - jelzőfunkciót tölt be:
hang van folyamatban. A 0FFH érték azt jelenti, hogy a
korábban kért hang képzése még nem fejeződött be.

0B15 Egy másik hangképzéssel kapcsolatos paraméter.
Tulajdonképpen egy parancsot képvisel. Ha értéke 0FFH,
akkor az új hangnak nem kell megvárnia, hogy az előző befejeződjön,
az új hang megszakítja a régít.

0B16 STOP, a billentyűzeten CTRL+ESC lenyomása.
Nagyon fontos rendszerváltozó. Ha egyidejűleg lenyomjuk
a CTRL és az ESC billentyűt, akkor ide FF érték íródik. Ennek a
memóriarekesznek a tartalmát a BASIC rendszeresen megvizsgálja, s ha
FF jelzést talál, megfelelő üzenet kiírása után megállítja a
programok futását és a parancsok végrehajtását is megszakítja.

0B17-0B18 A processzor-stack alsó határa a zárt alakzatok
festésénél. A CPU -- mint arról már beszéltünk --
működése közben megfelelő RAM-területet igényel a szubrutinok
visszatérési címeinek és belső regisztereinek tárolására. Ennek
neve: stack (veremmemória). E memóriaterület aktuális címekre mutat
a CPU 16 bites SP regisztere (stack pointer, veremmemória címmutató).
A stack egy felülről definiált kezdőcímtől, a tárolt adatok számától
függően, az alsó memóriacímek felé mélyül.

Erre a célra az U0 RAM-ban meglehetősen nagyméretű terület áll
rendelkezésre, azonban bonyolult alakzatok kifestése esetén, az ezt
a feladatot megoldó szubrutin különösen veszélyes mértékben
mélyítheti a vermet. Annak érdekében, hogy az alsóbb címeken
elhelyezett egyéb adatok nehegy megsérüljenek, az alakzatfestő rutin
folyamatosan ellenőrzi az SP értékét.

A 0B17-0B18H címek a veremmutató megengedhető legkisebb
értékét tárolják.

A beírt érték: 0F10H.

Valójában a mikroprocesszor számára fenntartott terület már a
0EACH címtől kezdődik, tehát a festésnél megengedett határérték
elérése esetén még mindig 64H (100 dec.) byte marad, amit a CPU
verem-műveletekhez használhat.

0B19-0B1A HI-MEM, a használható legnagyobb RAM-cím

A TVC 64k jelű gépeknél -- hibátlan esetben -- a
legnagyobb memóriacím: 0BFFFH.

A ROM számos művelet elvégzése előtt ellenőrzi a felhasználó
memória hivatkozásainak korrektségét, s a legnagyobb lehetséges RAM-
cím túllépése esetén hibát jelez.

0B1B U3-STAT, az U3 RAM állapota.

A rendszer a gép bekapcsolását követő inicializálás során valamennyi RAM-ot teszteli, az összes memóriarekesz ismételt írásával-olvasásával. Az U3 teszt eredményét erre a címre teszi a rendszer:

00: az U3 RAM jó
FF: az U3 hibás.

0B1C Bővítőkérdő alaphozzárendelés.

A rendszer a 06. funkcióosztályhoz a soros vonalat rendeli hozzá, így ha van a rendszerhez RS232-es soros vonali kártya csatlakoztatva: 00, egyébként FF a byte tartalma.

0B1D-0B1E TIME, a rendszer belső, kétbyte-os órája.

A megszakításokat kezelő alprogram mindig megnöveli eggyel (a kurzor-IT esetén pl. 20 ms-onként).

0B1F IRQ-STAT, az IT források engedélyezése.

A TV-Computerben megszakításkérő jelek forrása a képmegjelenítést vezérlő áramkör kurzorkimenete, a hangkeltésre szolgáló frekvenciaosztó és egy külső eszköz valamelyik bővítőcsatlakozóba helyezett interface-kártyája lehet. Az IRQ-STAT változó bitjeinek 1 értéke ezeket engedélyezi, a következők szerint:

b7	b6	b5	b4	b3	b2	b1	b0
-	-	3.	2.	1.	0.	hang kurzor	
		cstl.	cstl.	cstl.	cstl.		

0B20 INT-FLAG, egyszerű állapotjelző.

Amikor a megszakításkezelő rutint hajtja végre a rendszer, értéke 0FFH, egyébként 00.

0B21 WARM-FLAG, meleg RESET végrehajtását jelzi.

Eléggé elterjedt szóhasználat "meleg RESET"-nek nevezzük a számítógép olyan állapotváltoztatását, aminek eredményeként minden fontos paraméter az alapértékre áll vissza, ám a memóriába írt programok nem vesznek el. Ez tehát egyfajta inicializálási folyamat. Ha a rendszer éppen ezt hajtja végre, akkor a WARM-FLAG-be átmenetileg 0FFH-t ír.

0B22 COLD-FLAG, hideg RESET szükségességét jelzi.

Amennyiben a rendszer annyira megsérült, hogy a normális állapot visszaállítása már lehetetlen, vagy ha a felhasználó kifejezetten így intézkedik, akkor a rendszer teljes inicializálást hajt végre, a számítógép a bekapcsolási alapállapotba kerül.

Ha a COLD-FLAG értéke 0FFH, akkor hideg RESET-et kell végrehajtani.

2.6 A nagy gondolat: ROM program a RAM-ban

0B23 A funkcióhívások bevezető része.

Az RST 30 szubrutinhívás kezdőcímén csak a JP 0B23 ugrási utasítás van, tehát a tényleges végrehajtás itt kezdődik.

0B23	E3	EX	(SP),HL	A visszatérési címet a veremből HL-be tölti, egyidejűleg a HL korábbi értékét tárolja
0B24	7E	LD	A,(HL)	Az RST 30 utáni byte-ról beolvassa a funkciókódot
0B25	23	INC	HL	Az ezt követő címre kell majd visszatérni
0B26	E3	EX	(SP),HL	Ezt a címet teszi vissza a verembe, egyidejűleg HL is az eredeti lesz.
0B27	08	EX	AF,AF	A funkciókódot a másodlagos készletbe teszi,
0B28	F5	PUSH	AF	az eredeti AF'-t menti,
0B29	3A0300	LD	A,(0003)	majd az aktuális lapozási
0B2C	F5	PUSH	AF	(paging) byte-ot is.
0B2D	3E70	LD	A,70	A végrehajtáshoz beállítja
0B2F	320300	LD	(0003),A	az U0-U1-U2-SYS
0B32	D302	OUT	(02),A	lapozási konfigurációt,
0B34	C363C3	JP	C363	majd ugrás a ROM megfelelő címére.

Így indul tehát valamennyi funkcióhívás kezelése.

A feladat lebonyolítása után pedig a program ugyan itt, az U0 RAM következő címén folytatódik. Most az A regiszterben fontos információ van: a hibakód (00, ha "minden rendben"), ezt meg kell őrizni a felhasználó számára.

Íme a funkcióhívások befejező része:

0B37	08	EX	AF,AF	A másodlagos AF-fel felcserélve, menti a hibakódot,
0B38	F1	POP	AF	A veremből kiveszi az eredeti paging-byte-ot és
0B39	320300	LD	(0003),A	visszaállítja a korábbi
0B3C	D302	OUT	(02),A	memorialapozást
0B3E	F1	POP	AF	Visszaáll az eredeti AF'
0B3F	08	EX	AF,AF	és a hibakód, tehát
0B40	C9	RET		kész.

Csak még -- sajnos -- a közepe hiányzik!

Érdemes észrevennünk a nagyszerű lehetőséget. Mivel a funkcióhívások végrehajtásának be- és kifutó része a RAM-ban van, tehát átírható, lehetőségünk van a funkcióhívások bővítésére. Ha ezek működési sebessége egyébként elfogadható, akkor a rendszer számára definiálatlan funkciókódoznak is értelmet adhatunk, önállóan megírva a szükséges végrehajtó rutinokat. Ehhez az előbbi programrészek jelentéktelen változtatására van csak szükség. (Ekkor a külön megírt rutinokat minden alkalommal be kell tölteni a memóriába, de az egységes hívhatóság miatt megéri.)

0B41 A megszakításokat kiszolgáló, 0038H címen kezdődő szubrutin ottani része csak bizonyos előkészületeket végez el. A rutin fő része a ROM 0C412H címén indul. A befejező rész pedig ebben az esetben is az U0-ban kapott helyet. Ezért az ide vonatkozó RAM-címek átírásával most is lehetőségünk van az operációs rendszer IT-rutinjának megváltoztatására, bővítésére vagy akár teljes lecserélésére. Erre bizonyos felhasználói programoknak szüksége lehet, azonban nagyon figyelmesen kell eljárni.

Az IT-alprogram befejező része:

0B41	320300	LD	(0003),A	Visszaállítja az eredeti
0B44	D302	OUT	(02),A	memórialapozást, majd az
0B46	F1	POP	AF	AF eredeti értékét
0B47	FB	EI		Engedélyezi a megszakításokat
0B48	C9	RET		és kész.

2.7 A video rendszerváltozók

0B49-0B4A A CPU veremmutatójának ideiglenes tárolására szolgál.

0B4B L-MODE, a vonalkeresztezési mód.

Ha a képernyő olyan helyére kell egy pontot elhelyezni, amely már előzőleg sem volt üres (vonalak metszéspontja, karakterek felülírásai, akkor az ilyen esetekre a TVC négyféle lehetőséget kínál. Ezek közül választunk az L-MODE változóba beírt értékekkel, a következők szerint:

- 00: teljes felülírás, az új bitek betöltése a video-memóriába
- 01: OR művelet a videomemória régi és új ponthoz tartozó bitjei között
- 02: bitenkénti AND művelet
- 03: XOR művelet

0B4C L-STYLE, vonaltípus.

A két pontot összekötő folyamatos és szaggatott vonalak 14 féle lehetséges típusából választhatunk az L-STYLE változó írásával. Nem soroljuk itt fel az összetartozó értékeket és vonalakat, a ROM ide vonatkozó részeinél a meghatározó bitmintákat úgyis megbeszéljük.

0B4D INK, az aktuális tintaszín.

A változó tartalmát 2-es és 4-es üzemmódban egy palettaregiszter sorszámának, 16-os üzemmódban pedig egy szín sorszámának értelmezi a rendszer. A tintaszín hivatkozás azt jelenti, hogy a képernyőn -- annak alapszínétől függetlenül -- a betűk, írásjelek, pontok és vonalak ezzel a színnel jelennek meg: tehát mintha egy alapszínű papírra ilyen színű tollal írnánk.

0B4E PAPER, az aktuális papírszín.

Tartalmát a rendszer ugyanúgy értelmezi, mint az INK változóét. A képernyő alapszínét határozza meg.

0B4F BORDER, az aktuális képkeretszín.

Az ebben tárolt értéket mindig a következő bitkiosztás szerint értelmezi a rendszer:

b7	b6	b5	b4	b3	b2	b1	b0
I	-	G	-	R	-	B	-

általános
intenzitás zöld vörös kék

0B50 V-FLAG, karakterfelülírás módja.

A képernyőn egy egész karakter helyén határozza meg a megjelenítés körülményeit olyan esetben, amikor az újonnan kiírandó karakter helyén már egy másik karakter található. Ilyenkor négyféle dolog történhet, a V-FLAG-ba írt értéktől függően:

00: a karaktermező az új PAPER és INK színnel teljesen átíródik

01: az új INK helyein a karaktermező régi tartalma látszik, máshol az új PAPER eltakarja a régit

02: éppen fordítva, most az új INK-pontok takarják el a régit, az új karakter PAPER része nem érvényesül

03: sem az új INK, sem az új PAPER nem érvényesül, tehát a kiírási kép a karaktermező pontjaiban változatlan

E lehetőségek kihasználásával változatos és szokatlanul érdekes képeffektusok érhetők el.

2.8 Még egyszer a billentyűzetről

0B51-0B5A PICTURE, a billentyűzet aktuális állapota.

A TV-Computer megszakításkezelő programja minden lefutáskor megvizsgálja a billentyűzet állapotát és az eredményt ezen a területen tárolja. Az itt található 10 byte tartalma tehát pontosan tükrözi a billentyűzet állapotát az utolsó vizsgálat idején (erre a vizsgálatra, mint tudjuk, átlagosan másodpercenként 50-szer kerül sor).

A következőkben megadjuk az egyes byte-ok által hordozott információkat. Emlékezzünk arra, hogy a billentyűk 10 sorba vannak szervezve, egy-egy sorban 8 billentyű található. A billentyűk lenyomott állapotának itt az 1 érték felel meg. A következő byte-ok mindegyike tehát bitenként egy-egy, összesen egy sor 8 billentyűjének állapotát mutatja. (Kivételt képeznek az ALT, LOCK, SHIFT és CTRL nyomógombok, ezek itt nem állítanak be bitet.)

Cím:	Sor:	b7	b6	b5	b4	b3	b2	b1	b0
0B51	0	4	1	Í	6	0	2	3	5
0B52	1	7	Ö	Ó	*	Ü	9	8	^
0B53	2	R	Q	@	Z	;	W	E	T
0B54	3	U	P	Ú	[Ő	O	I]
0B55	4	F	A	<	H	\	S	D	G
0B56	5	J	É	Ü	RET	Á	L	K	DEL
0B57	6	V	Y	LOCK	N	SHIFT	X	C	B
0B58	7	M	-	SPACE	CTRL	ESC	.	,	ALT
0B59	8		LEFT	RIGHT	ACC	FIRE	DOWN	UP	INS
0B5A	9		LEFT	RIGHT	ACC	FIRE	DOWN	UP	

A felsorolásban mindenütt csak a billentyűkre írt alsó karaktereket adtuk meg. A billentyűkön nem -- vagy nem így -- szereplő rövidítések és szavak jelentése:

RET	=	RETURN	RIGHT	=	jobbra	FIRE	=	tűz
SPACE	=	betűköz	DOWN	=	le	ACC	=	gyorsítás
LEFT	=	balra	UP	=	fel			(accelerator)

A billentyűzetkezelés teljes mechanizmusát a ROM lista tanulmányozása során ismerjük meg.

0B5B-0B64 OLD-PICT, a billentyűzet előző állapota.

Fontosan megfelel a PICTURE-vel kapcsolatban leírtaknak, de mindig a megelőző vizsgálat szerinti állapotot tárolja. A két 10 byte-os mátrix összehasonlításával nyilvánvalóan eldönthető, hogy mikor és milyen változás történik a billentyűzet állapotában.

0B65 DELAY-KEY, billentyűkésleltetés.

A tartósan lenyomott billentyű automatikus funkcióismétlésének késleltetését állítjuk be ebben a változóban. Az azonnali ismétlés nyilvánvalóan nem célszerű. A beírt érték 20 ms egységekben adja meg a késleltetést. Az alapérték 1EH, tehát egy billentyű tartós lenyomása esetén közelítőleg 0.6 s idő után kezdődik a funkcióismétlés.

0B66 LOCK-KEY, a billentyűzet tartós LOCK állapota.

Az Olvasó számára feltehetően ismert, hogy a billentyűzet egyes nyomógombjainak hatása az ALT, a CTRL vagy a SHIFT gombok együttes lenyomásával alkalmilag, az előbbiek és a LOCK billentyű lenyomása után pedig tartósan megváltoztatható. Ez utóbbi állapotokban a képernyőn látható kurzor képe is jelzi az alapműködéstől való eltérést. A következőkben megadjuk az egyes állapotok jellemzőit.

SHIFT+LOCK a kurzor képe inverz S.

Ebben az állapotban nagybetűket írhatunk, ill. a többi billentyű esetében a rájuk írt felső szimbólumok jelennek meg. SHIFT-tel együtt lenyomva az alapállapotnak megfelelő funkciókat kapjuk.

CTRL+LOCK a kurzor képe inverz C.

Ebben az állapotban nagybetűket írhatunk, de a többi billentyű hatása nem változik meg. A SHIFT alkalmi lenyomása most is visszaváltó hatású.

ALT+LOCK a kurzor képe inverz A.

Az egyes nyomógombok lenyomása a definiálható karaktereket jeleníti meg, a következők szerint (a táblázatban természetesen csak az egyes ASCII kódokat generáló eredeti billentyűszimbólumokat tudjuk megadni:

ASC	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A	0	1	2	3	4	5	6	7	8	9	*	;	<	-	.	,
B	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
C	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	
D	À	É	Ï	Ó	Ö	Ő	Û	Ü	Û	Ü						

A táblázatból hiányzó 0CFH és D9...DFH kódok nem az ALT állapotból, hanem a CTRL billentyű és az Á, ill. a CTRL és az É, Í, Ó, Ö, Ő, Ú, Ü egyidejű lenyomásával érhető el. Most az ALT billentyű alkalmi lenyomása lesz visszaváltó hatású.

A LOCK nyomógomb önmagában való lenyomása mindig az alapértelmezéshez juttatja a billentyűzetet.

A LOCK-KEY változóba írt érték a billentyűzet állapotát a következők szerint határozza meg (00: alapállapot):

b7	b5	b4	b3	b2	b1	b0
-	-	-	-	ALT	SHIFT	CTRL
				(04)	(02)	(01)

0B67 RATE-KEY, a billentyűismétlés üteme.

A tartósan lenyomott billentyű funkcióismétlésének szaporaságát állíthatjuk be, 20 ms egységekben. Már tudjuk, hogy ez az érték nem azonnal érvényesül a billentyű lenyomása után, hanem csak egy meghatározott késleltetés -- DELAY-KEY változóban megadott idő -- letelte után.

Az alapérték: 03, tehát az első késleltetés után a tartósan lenyomott billentyű hatása 60 ms-onként ismétlődik.

0B68 HOLD-DIS, a CTRL+P hatásának tiltása.

Az operációs rendszer megszakításokat kiszolgáló alprogramja a billentyűzet minden leolvasásakor ellenőrzi e két nyomógomb együttes lenyomását. Ha a CTRL és a P billentyűk le vannak nyomva, akkor a rendszer működését teljesen beszünteti.

Ez a funkció nagyon hasznos minden olyan esetben, amikor egy gyorsan pergő esemény közben akarunk bizonyos vizuális elemzéseket tenni, vagy éppenséggel kifejezetten időt akarunk nyerni egy program futása közben. Így megállíthatjuk egy hosszú program kilistázását a képernyőn vagy a képernyőn rövid időre megjelenő és változó ábrákat hosszabb időre kimerevíthetjük stb.

Kívánatos lehet ugyanakkor ennek a funkciónak a tiltása is. Erre szolgál a HOLD-DIS (hold disable, a beragadás megakadályozása) változó.

Ha a HOLD-DIS változóba FF-et írunk, akkor a rendszer nem foglalkozik a CTRL+P billentyűk együttes lenyomásával.

2.9 A soros vonal két byte-ja

0B69 BAUD, a soros vonal sebessége.

Egyetlen byte-ot használunk fel, tehát a beírt érték nyilvánvalóan szimbolikus, a hozzá tartozó átviteli sebességek baud egységekben (bit/s) a következők:

BAUD:	00	01	02	03	04	05	06	07	08
bit/s:	110	150	300	600	1200	2400	4800	9600	19200

Emlékeztetem az Olvasót arra, hogy a közölt baud értékek a soros vonalat vezérlő USART beállított állapota szerint, a hanggenerátor frekvenciáját is szigorúan meghatározzák. A BAUD-ban inicializált érték 04, tehát az átviteli sebesség 1200 baud bit/s). Ugyanakkor az USART üzemmódbéállító utasításában 16-szoros órajel-frekvenciát írtunk elő, amiből a hanggenerátor soros vonalhoz szinkronozott frekvenciája: $16 \cdot 1200 = 19200$ Hz. A megfelelő programozandó PITCH érték pedig: 4015 (0FAFH). Ha a PITCH érték ettől eltér, azt egy külön változóban jelezni fogjuk.

0B6A FORMAT, a soros vonali USART üzemmódja.

Nem bocsátkozunk ismét az USART parancsaival kapcsolatos részletekbe, csak annyi jegyzünk meg, hogy az USART alapállapotában, a 0...3 csatlakozóhelyeknek megfelelően a 11H, 21H, 31H vagy 41H portokra kiküldött parancsbyte üzemmódkiválasztó utasításnak számít és a FORMAT-ban inicializált 0EEH alapérték a következőt jelenti:

- 2 STOP bit,
- paritáskiküldés és -ellenőrzés nincs,
- karakterhossz 8 bit,
- 16-szoros órajel.

2.10 A magnetofonkezelés rendszerváltozói

A soros vonalhoz tartozó két változó után a magnetofonnal kapcsolatos információkkal folytatódik az U0 RAM.

0B6B BUFFER, file-típus (pufferelt vagy sem).
00: nem pufferelt file,
FF: pufferelt file.

Megmagyarázzuk e fogalmakat.

Amikor a memória meghatározott részének tartalmát magnetofon segítségével tároljuk, ez a szalagon egy logikailag és fizikailag is összefüggő adatállományt: file létrehozását eredményezi.

A file-okhoz szervesen hozzátartoznak az azokat azonosító, jellemzőiket megadó és bizonyos ellenőrző funkciókat betöltő byte-ok is. Ennek részleteiről a ROM magnetofonkezelő

rutinjainak tanulmányozásakor még szó esik. Most csak annyit kell tudnunk, hogy minden file-hoz hozzá tartozik egy ún. fejtömb és egy vagy több adattömb. Utóbbi esetleg el is maradhat.

A nem pufferelt file-ok a fejtömb után egyetlen adattömbben tartalmazzák a teljes tárolt memóriaterületet.

Pufferelt file-ok esetén az adatok 256 byte-os tömbökben úgy helyezkednek el a szalagon, hogy az egyes tömbök fizikailag is elkülönülő blokkokat alkotnak és részben különállóan kezelhetők.

A pufferelt -- nem pufferelt elnevezést az indokolja, hogy a kétféle típusú file kezelése különböző: a pufferelt file-ok esetén a számítógép és a magnetofon közötti adatforgalom egy meghatározott című, állandó memóriaterület (puffer) közvetítésével bonyolódik le, míg nem pufferelt file-ok esetében a kimentendő vagy beolvasandó adatok közvetlenül a magnetofon és a megcímezett memóriaterületek között mozognak.

A felhasználótól függ, hogy a memória tárolásakor milyen típusú file-t hoz létre, ez nyilvánvalóan a tervezett későbbi felhasználástól függ, azonban a rendszernek az eltérő kezelés miatt tudnia kell, hogy melyik típusról van szó. Ennek jelzésére való a BUFFER rendszerváltozó.

0B6C REMRED, a magnetofonmotorok vezérlése.

A magnetofonkezelő hardverről szóló fejezetben megbeszéltük, hogy a TVC a magnetofonnal lebonyolított adatforgalom során két készülék táv szabályozását képes megoldani (feltéve, hogy a magnetofonok is fel vannak szerelve a motor-távkapcsoló áramkörökkel).

A 05-ös port b7-b6 bitjének írásával küldhetünk a gép magnetofoncsatlakozóira motorvezérlő jeleket.

A két magnetofont külön-külön felvételre vagy lejátszásra állíthatjuk, s ezután a REMRED változóba írt adattal egyszerű munkamegosztást alakíthatunk ki, a következő módon:

Érték	Bal	Jobb
00	olvasás	írás
40	-	olvasás-írás
80	olvasás-írás	-
C0	írás	olvasás

Ha a magnetofon nincs felszerelve a távkapcsoláshoz szükséges áramkörökkel, akkor az előbbi vezérléseket természetesen kézzel kell megoldani. A számítógép két magnetofonkimenetének jelvezetéke egyébként közös, tehát ilyen magnetofonok esetén az is közömbös, hogy az összekapcsoló vezeték melyik csatlakozóba dugjuk.

0B6D PROTECT, a file írásvédelme.

Ha egy file szalagra való felírása előtt ebbe a változóba nem nullát írunk, akkor a file írásvédett lesz. Ez azt eredményezni, hogy a rendszer bizonyos fokú védelmet nyújt az ilyen file-ok tartalmának átírásával szemben. A részletes mechanizmussal a megfelelő helyen ismerkedünk meg.

0B6E EOF, file vége.

Egyszerű jelzőbyte, amelyet a rendszer nem nulla értékre állít be, ha a szalagról a file utolsó karakterét is beolvasta.

0B6F-0B70 MUDDLE, az ellenőrzőbyte-ok megváltoztatása.

A muddle szó összezavarást, rendetlenséget jelent. Értéke eredetileg 0.

A file-ok felírásánál az adatok fizikailag 258 byte-os szektorokba vannak szervezve. Minden szektorhoz hozzá tartozik néhány olyan byte -- az adatokon kívül - amely ezt a szektort azonosítja, jellemzi. Ezek között 2 byte a szektorban található adatok ellenőrzésére szolgáló CRC (Cyclic Redundancy Check, az egész blokkra kiterjedő ciklikus paritásellenőrzés). Ennek értékét a rendszer a file szektorainak kiküldésekor folyamatosan kiszámítja és eredményét a szalagra írja. Visszaolvasáskor a szalagról beolvasott byte-okkal végzi el ugyanezt a számítást, s amennyiben a kapott érték eltér a szalagon rögzített értéktől, az nyilvánvalóan rögzítési vagy olvasási hibát jelent. Ilyenkor a rendszer le is állítja a további beolvasást.

A CRC számítás kezdőértéke mind a kimentésnél, mind a visszaolvasás kezdetén a MUDDLE-ben tárolt érték. Ezt a rendszer 0-ra inicializálja, ha azonban egy file tárolása előtt átírjuk, akkor a dolog természetéből adódóan az egyes szektorokhoz felírt CRC érték szinte bizonyosan mindig eltér attól, ami a MUDDLE 0 értékével a szalagra kerülne.

Ebből pedig már következik, hogy a visszaolvasás csak úgy lehetséges, ha előtte a MUDDLE-t ugyanarra a kezdőértékre állítjuk be, ami a tárolás előtt volt, másként az egyes szektorok beolvasásnál számított és szalagon rögzített CRC értéke állandóan eltér.

Így elérhető, hogy a tárolt file-okhoz csak az férhessen hozzá, aki a MUDDLE változó kezdőértékét ismeri, tehát egyfajta jelszó szerepét töltheti be.

0B71 E rendszerváltozó 00 értéke azt jelzi, hogy a hanggenerátor frekvenciája a soros vonalhoz van szinkronozva.

Ez elromlik akkor, ha a rendszer a magnetofonnal kommunikál (a magnetofonkezelés, mint látni fogjuk a hanggenerátor használatára épül), vagy ha a felhasználó hanggal kapcsolatos utasításokat adott.

Ha a generátor frekvenciája megváltozott, akkor a változó értéke: 0FFH.

2.11 A munkaterületek

0B73 Az üzemmódot tárolja
00: 2 színű 01: 4 színű 02: 16 színű

0B74 Az elektronsugár ki- vagy bekapcsolt állapotát jelzi. Szemléletesen úgy fogalmazhatunk, hogy a TVC egy képzeletbeli, többszínű tollat kezel, amelynek nyoma látszik akkor, ha le van téve a papírra -- ekkor elmozgatva vonalat is rajzol -- felemelt állapotban azonban bárhová mozgatható anélkül, hogy ennek a képernyőn látható jele tenne.

Ezzel a szóhasználattal élve a 0B74-en tárolt adat jelentése:

FF: a toll le van téve

00: a toll nem ér a papírhoz

0B75 A videomemória egy byte-jába töltendő értéket tárolja

0B76-0B77 A videomemória aktuális tárrekeszének címe, amelyből az éppen zajló megjelenítési munka során adatot kell kiolvasni, vagy ahová a megjelenítendő adatot tölteni kell

0B78-0B79 Logikai x koordináta

Az Olvasó számára bizonyára ismeretes, hogy a pontok kirajzolásához használható utasításokban, annak érdekében, hogy a pontok az üzemmódtól függetlenül mindig a képernyő ugyanazon pozícióján jelenjenek meg, ún. fizikai koordinátákat használunk.

Ez azt jelenti, hogy a pontok vízszintes helyzetét minden üzemmódban a 0...1023, a függőleges helyzetet pedig a 0...959 számokkal adhatjuk meg.

Tudjuk ugyanakkor, hogy a TVC egyes üzemmódjaiban a ténylegesen megjeleníthető elemi pontméret eltér az így meghatározott mérettől, mégpedig 2-es üzemmódban 2-szer akkora, 4-es üzemmódban 4-szer akkora, 16-os üzemmódban pedig 8-szor akkora egy pont szélessége, mint ami az egy tv-sor fizikailag címzett 1024 részre való felosztásából adódna. Hiszen tudjuk, hogy 2-es üzemmódban 512 pont, 4-es módban 256, 16-os üzemmódban pedig csak 128 pont fér el egy sorban.

A függőleges helyzetet meghatározó fizikai koordinátát mindig 4-gyel kell osztani, hiszen -- bár 960 sort címezhetünk -- a képernyőn összesen csak 240 tv-sor van.

A rendszernek működése során szüksége van ezekre az átszámításokra.

A 0B78-0B79 címeken, az üzemmódot is figyelembe véve, a logikai x koordinátát tároljuk. Ez a fizikai koordinátából, az egyes üzemmódok szerint, a következő osztással adódik:

	2-es	4-es	16-os	üzemmódban
	a fizikai x koordináta			
logikai x =	: 2	: 4	: 8	

Ezek az értékek -- mint sorszámok -- az üzemmód szerinti méreteket figyelembe véve jelölik ki a pontok helyét a képernyőn.

0B7A-0B7B Logikai y koordináta

A pontok helyére vonatkozó utasításokban a sorszámot a 0...959 fizikai koordinátákkal adjuk meg. A megfelelő logikai érték minden üzemmódban ennek 4-gyel való osztásával adódik.

A 960 féle fizikai hivatkozási lehetőség a 240 db rendelkezésünkre álló tv-sor miatt sohasem használható ki.

0B7C-0B7D A fizikai x koordinátákat tárolja itt a rendszer. Ezt a 0...1023 közötti számot kell szerepeltetnünk a BASIC PLOT utasításban vagy a megfelelő funkcióhívásokban.

0B7E-0B7F A fizikai y koordináta

Ezt a 0...959 közötti számot kell szerepeltetnünk a BASIC PLOT utasításában vagy a megfelelt funkcióhívásokban.

0B83 Az L-STYLE változóban megadott vonaltípushoz tartozó, 8 pontból álló vonalelemek bitképét tartalmazza. A bit 0 értéke az aktuális PAPER, 1 értéke az aktuális INK színű pontot határoz meg.

A pontokat *-gal jelölve a 14 vonaltípus bitképből adódó kódja és illusztrációja a következő:

L-STYLE	Kód	Bitkép	Illusztráció
01	FF	11111111	*****
02	AA	10101010	* * * * *
03	CC	11001100	** ** ** **
04	EE	11101110	*** ** **
05	88	10001000	* * * *
06	DA	11011010	** ** * ** *
07	E4	11100100	*** * ** *
08	F6	11110110	**** ** ** **
09	FA	11111010	***** * ** ** *
0A	FE	11111110	***** ** ** **
0B	FC	11111100	***** ** ** **
0C	FB	11111000	***** ** ** **
0D	F0	11110000	***** ** ** **
0E	EA	11101010	*** * * ** * *

Az U0 RAM további részei a különböző funkciójú rutinok nagy fontosságú munkaváltozói, közülük csak a legfontosabbakat emeljük ki.

0BE5 A billentyűzeten egy gomb lenyomott állapotát jelzi. Ha értéke 00: nincs lenyomva billentyű, míg a 0FFH érték azt jelzi, hogy egy lenyomott billentyű kódja a megfelelő tárrekeszből (0BE9H) kiolvasható.

0BE7 00 értéke azt jelzi, hogy a LOCK nem volt lenyomva.

0BE8 Az üzemmódváltó billentyűk lenyomását jelzi:
 00: nem volt ilyen gomb lenyomva,
 02: a SHIFT le volt nyomva,
 04: a CTRL volt lenyomva,
 08: az ALT lenyomását jelzi.

0BE9 A lenyomott billentyű kódja.

0BEA A tartósan lenyomott billentyű funkcióismétlésének késleltetéséhez használt számláló.

0BEB A tartósan lenyomott billentyű funkcióismétléséhez használt számláló.

0BEC-0BED Az eltérést mutató cím az OLD-PIC táblázatban.

A rendszer az utolsó és a megelőző billentyű letapogatás eredményét visszafelé haladva hasonlítja össze. Ahol először eltérést talál, azt a címet jegyzi meg ebben a két byte-ban. Az új billentyű kódját ennek alapián képezi. Az eljárás ismerete -- amelyet a ROM megfelelő részeinél tisztázunk -- az összetett billentyűkombinációkat használó programok készítésénél hasznos.

0BEE A billentyűzet aktuális letapogatásánál az első eltérést mutató, legkisebb helyi értékű bit. Ebben a byte-ban tehát egyetlen bit értéke 1, mégpedig azé, amelyik az előző OLD-PIC táblázatbeli címen belül a legkisebb helyi értékű, az ottanitól eltérő értékű. Tehát az új, lenyomott billentyűre utal, az összehasonlítás leírt sorrendjében.

0BEF A hangidőtartam számlálásához használja a rendszer. A hangképzés definiálásakor megadott DURATION (időtartam) értéke íródik először ide, majd visszafelé számlálja a rendszer 0-ig, 20 ms egységekben.

Az ezután következő, igen nagy méretű: 0BF0--0E47H címek közötti területet a ROM magnetofonkezelő szubrutinjai használják.

0BF0 Borderszín.

A magnetofonnal kapcsolatos műveletek közben az aktuális keretszín tárolására szolgált.

0BF1 VERIFY, a szalagon tárolt file és a memóriatartalom összehasonlítása.

Ha az ezen a címen levő érték nem nulla, akkor a rendszer nem végez beolvasást, hanem csak összehasonlítja a megadott memóriaterület és a szalag adatbyte-jainak tartalmát.

0BF2 A magnetofonnal kapcsolatos műveletek során az operációs rendszer megszakításkezelő programja kikapcsolódik. A magnetofonkezelő szoftver egy C9 kódot tesz a 0038H IT-belépési címre, ami a főprogramba való azonnali visszatérést eredményez. Az IT-kezelő programra háruló feladatokat -- mint majd látni fogjuk -- ilyenkor másképpen oldja meg a rendszer, a megszakításokat hanggal vezérli és elsősorban csak pontos időzítésre használja.

A 0BF2 címen az IT alapprogram belépő byte-ját tároljuk, amelynek helyére az említett C9 kód kerül.

Ennek visszairásával történik -- egyebek mellett -- a normális működési állapot visszaállítása.

0BF3 Olvasásra megnyitott file és file-típus.
 Egy file beolvasási parancsát követően a rendszer megkeresi a szalagon a hivatkozott file-t (név megadása nélkül az elsőt), beolvassa a fejtömbjét, s az abban megadott információkat elhelyezi a munkaterületén. (Részletesebben erről még lesz szó.) Ezt a műveletet nevezzük file-megnyitásnak.
 A TVC egyetlen olvasásra megnyitott file-t enged meg.
 Ha a 0BF3 tartalma: 00, akkor még nincs nyitott file. A 01 érték puffereelt, a 11H érték nem puffereelt file beolvasását jelzi.

0BF4-0C04 A kért file neve.
 Ezen a 17 (dec.) byte-on helyezi el a rendszer az általunk beírt file-nevet. Az első byte a név tényleges hossza, utána pedig legfeljebb 16 (dec.) byte-on a név ASCII kódú karakterei állnak.

0C05-0C15 A file neve a szalagról.
 A rendszer ide tölti be a szalagról beolvasott file nevét -- ez sikeres (eltalált) beolvasás esetén megegyezik a megadott névvel. Ha a két név azonos vagy a felhasználó nem névvel adta meg a parancsot, akkor a rendszer intézkedik a szalagon talált file beolvasásáról.

0C16-0D04 Input puffer.
 Erre a területre történik a puffereelt file-ok adatainak beolvasása -- szektoronként.

0D05-0D06 A szalagról beolvasott adatbyte-ok száma.

0D07-0D08 A következő -- még fel nem dolgozott -- adatbyte címe a pufferben.

0D09-0D0A A szalagról még beolvasandó byte-ok száma.

0D0B Hibakód, a szalagról való beolvasás, ill. a feldolgozás közben esetleg észlelt hibáról tájékoztat. Hiba esetén a munka félbeszakad.

0D0C Írásvédelem.
 Ha a beolvasott file írásvédett, azt ezen a helyen is nyilvántartjuk egy nem 00 értékkel.

0D0D Szektorszám.
 A beolvasott szektor sorszámát tartalmazza.

0D0E Szektorvégjel.
 Egy-egy szektor beolvasása után a végjelet ide teszi le a rendszer. Értéke 00 a közbülső szektoroknál, 0FFH az utolsó szektor esetén.

0D0F Tömbtípus.
A beolvasott tömb típusát tartja itt nyilván a rendszer.
Értéke: FF a fejtömbök esetén, adattömböknél pedig: 00.

0D10-0D11 Az első betöltendő memóriacím.

0D13 Beolvasási fázis.
Értéke FF, ha az aktuális olvasási folyamat egy file
nyitása volt, 00: ha egy korábban megnyitott file-ból történt.

0D14 Írásra megnyitott file és file-típus.
Értéke 00, ha még nincs másik, írásra megnyitott file.
Írási parancs esetén a rendszer először ezt ellenőrzi, s ha még nincs
másik írásra megnyitott file, akkor ide bejegyzzi a file --
felhasználó által előírt -- típusát (pufferelt -- nem pufferelt),
átírva ezzel egyúttal az újabb file-ok megnyitását lehetővé tevő 00
értéket is.

0D15-0D25 Megadott file-név.
Itt helyezi el a rendszer a felhasználó által megadott
file-nevet, legfeljebb 16 (dec.) hosszúságban, az esetleges
kisbetűket nagybetűkké konvertálva.

0D26-0E25 Output puffer.
Erről a területről történik a pufferelt file-ok külső
tárolás -- szektoronként.

0E26-0E27 A tárolandó byte-ok száma.

0E28-0E29 A következő tárolandó byte címe.

0E2A Hibakód

0E2C-0E2D A tárolandó memóriaterület kezdőcíme.

0E2E A tárolandó karakter.

0E2F A file típusa: 01: pufferelt,
 11H: nem pufferelt

0E30 Írásvédelem: 00 a file nem írásvédett,
 nem 00 a kimentett file írásvédett.

0E32 Írási fázis: FF fejtömb írása,
 00 adattömb írása.

0E48 A kurzor villogtatásának időzítéséhez használt
számláló.

0E49 A kurzor y koordinátája (sorszám).

0E4A A kurzor x koordinátája, soron belüli pozíciója.

0E4B-0E4C A kurzor helyének megfelelő cím az ASCII pufferben.

0E4D A kurzorpozícionálással kapcsolatos változó.

0E4E-0E4F A kurzor pozíciójának tárolása.

0E50-0E67 Sorjellemzők.
 A képernyő 24 sorára vonatkozó információkat tárolja ezekben a memóriarekeszekben a rendszer. A részleteket a megfelelő ROM-listarészek tanulmányozásánál fogjuk megismerni.

0E68-0E6A Ugrás a végrehajtáshoz.
 Ezen a három byte-on a rendszer egy -- az üzemmódtól függő címre való -- ugrási utasítást helyez el. A karakterek megjelenítését az EDITOR nem a megfelelő VIDEO-rutinok hívásával, hanem teljesen önállóan végzi.

0E6B Aktuális sorhossz.
 A háromféle üzemmódban a képernyőn megjelenő szövegeket más-más sorbontásban kell kezelni. Ezt a byte-ot a rendszer az aktuális üzemmódhoz tartozó sorhossz értékével tölti fel, tehát:

2-es üzemmód esetén 40H (64 dec.),
 4-es üzemmódban 20H (32 dec.),
 16 színű üzemmódban 10H (16 dec.).

0E6C Karakterszélesség (byte).
 Mint tudjuk, az egyes üzemmódokban egy-egy pontot, ami a képernyőn megjelenik, a videomemóriában 1, 2 vagy 4 bittel adunk meg. Mivel egy karakter egy sorának ábrázolásához 8 pontot használtunk fel, ezért ehhez a videomemóriában a 2-es üzemmódban 1 byte, a 4-es módban 2, a 16 színű üzemmódban pedig 4 byte tartozik.
 Ezen a memóriacímen az aktuális üzemmódhoz tartozó karakterszélességet tároljuk, az előbbiek szerint:

2-es mód: 01,
 4-es mód: 02,
 16-os mód: 04.

0E6D-0E94 A kurzor helyén álló karakter tárolása.
 Ez a 40 (dec.) byte annak a video-memória területnek a másolata, amelynek megfelelő helyeken, a képernyőn a kurzor éppen villog.
 Innen történik az eredeti képtartalom visszaállítása, amikor a kurzor az adott területet nem fedi le.

0E95 INK színű vonal.
 Ebben a byte-ban azt az értéket tároljuk, amelyet a videomemóriába írva, a képernyő megfelelő helyén, az aktuális üzemmódban, az egy byte-ban tárolható hosszúságú tintaszínű vonaldarab jelenne meg.
 Tárolása a karakterek megjelenítését segíti.

0E96 PAPER színű vonal.
 Ugyanaz, mint az előző, csak a képernyő alapszínével.

0EAC-16AB A Z 80-as CPU számára biztosított veremterület. Mint már szó volt róla, a zárt alakzatok befestésénél lehet szükség ilyen nagy méretű (2 kbyte!) verem használatára.

2.12 Néhány BASIC munkaváltozó

A következő -- 16AC...19EEH közötti tekintélyes munkaterület a BASIC számára van fenntartva.

Fontos és kevésbé fontos változók (természetesen a maga nemében mindegyik az), a részeredmények és működési paraméterek adminisztrálása folyik az U0 RAM-nak ezen a részén. A megfelelő helyen foglalkozunk velük, itt csak néhányat emelünk ki közülük.

1700 Fontos jelzőbiteket tartalmaz:
 b0=1 kell TRACE funkció,
 b1=1 nem kell "ok" üzenet,
 b2=1 futó program,
 b3=1 nyitott file van.

1701 A feldolgozás előtt álló változó típusa:
 01 karakterlánc,
 03 szám.

1702 Értéke 1, ha a rendszer feltételes utasítást hajt végre.

1704 Az RST 18 utáni aktuális byte tárolása.

1705 Az aktuális funkcióosztály.

1706 A perifériahivatkozással kijelölt eszköz.

1707 AUTORUN jelzése (OFFH).

1708 TYPE, a szimbólumtábla aktuálisan kezelt elemének típusa.

1709	RND változó
170A-170B	RND változó
170C-170D	Az aktuális BASIC sor kezdőcíme.
170E-170F	A következő BASIC sor kezdőcíme.
1710-1711	A következő utasítás címe.
1712-1713	DATA sormutató.
1714-1715	Az aktuális DATA sorban a következő adat címe.
1716-1717	INPUT adatmutató.
1718-1719	Az RST 18 utáni felsorolásban következő kód címe.
171A-171B	A BASIC veremmutatója egy-egy BASIC utasítás kezdetén.
1720-1721	VLOMEM, az aktuális BASIC báziscím.
1722-1723	TEXT, az aktuális BASIC program kezdőcíme.
1724-1725	CHAIN, a szimbólumtábla utolsó elemének címe.
1726-1727	TOP, a szimbólumtábla következő szabad byte-jának címe.
172A-172B	PITCH értéke.
1732-1830	COMMAND, aktuális BASIC sorpuffer.
1831-192F	BUFFER, input puffer a BASIC sorok beírásához.
19C0-19C6	X, lebegőpontos regiszter.
19C7-19D0	Y, lebegőpontos regiszter.
19CE-19DE	File-név puffer.
19DF-19EE	Fejléc puffer.

A 19EFH cím pedig a VIDEOTON TV-Computer kitüntetett memóriacíme. Az U0 RAM ettől kezdve szabad.

2.13 Az előkészületek vége

A dolgok ezután kezdenek nagyon összefüggeni egymással. Egy-egy részprobléma kimerítő tárgyalásához, annak egészen más, szerteágazó kérdésekkel való összefüggéseit is figyelembe kell venni, csak azokkal együtt kapja meg igazi értelmét.

Meg kell elégednünk azzal, hogy amennyire lehet továbbra is igyekszünk átfogban, globálisan kezelni a problémákat, ám belenyugszunk abba, hogy adott esetben nem lehet az adott kérdéskörben felvetődő minden részletre ott és azonnal választ kapni.

Amikor a részleteken türelmesen átrágtuk magunkat módot találunk arra, hogy rövid időre megállva, visszatekintsünk az addig megtett útra, s megpróbáljuk az esetleg még mindig nyitva maradt kérdéseket megválaszolni. Az igazság azonban az, hogy a számítógép a maga legteljesebb valóságában csak minden részlet megismerése után tárul fel előttünk.

A következő fejezetekben nagyon elmélyedünk a részletekben. Egy-egy fontosabb rész előtt megbeszéljük az előttünk álló feladatokat, a fejezetek végén pedig összefoglaljuk, néhol értékeljük is a tapasztalatainkat.

Érdemes nyitott szemmel járnunk, mert alig lesz olyan része az előttünk álló munkának, amit máshol úgyesen ne használhatnánk fel. Az a gyűjtemény, amit én a ROM tanulmányozása után Önnek felajánlok, semmiképpen sem nevezhető teljesnek. Kérem, hogy közben Ön is jegyezgessen, s az egy-egy adott helyen felvillanó lehetőségeket ne hagyja veszendőbe menni.

3. A TV-COMPUTER ROM PROGRAMJA

3.1 Nyitány

3.1.1 A gép intelligenciája

Itt állunk a TV-Computer agya előtt!

Mégpedig abban az értelemben, hogy minden, ami a gép bekapcsolása pillanatától történik, az a ROM kezdeményező, szervező, felügyelő és ellenőrző munkájának köszönhető.

Természetesen, amikor ROM-ot mondunk, akkor valójában sohasem a tranzistorok százezreit és más alkatrészeket tartalmazó chip-ekre kell elsősorban gondolnunk, hanem sokkal inkább az általuk megvalósult bitekre, a belőlük szervezett és már konkrét tevékenységelemeket hordozó byte-okra, a számítógép agysejtjeire, amelyek csodálatos szövevénye végül is egy dinamikus, a külvilág felé nyitott rendszert teremt.

A ROM készséges és fáradhatatlan rabszolgája a mikroprocesszor. Minden feladatot úgy fogalmaz meg, hogy az a CPU számára végrehajtható legyen. A ROM minden intézkedése csakis a CPU közreműködésével realizálódik, ám így képes arra, hogy globálisabb problémákat is kezeljen, működése logikailag magasabb szinten is értelmes legyen.

A ROM beszélget az emberrel!

Még így is eléggé butuska jószág, azonban ez csak a TVC és más, hasonló kategóriájú számítógépek esetében van így. Ma már könnyűszerrel készíthetők olyan számítógépek, amelyek mesterséges intelligenciája nagyon magas.

Nem lehet célunk e gondolat további, részletekbe menő fejtegetése -- az eziránt érdeklődő Olvasó ehhez bőséges irodalmat talál --, de még megjegyezzük, hogy az eddig a ROM-nak tulajdonított intelligencia és az aggyal való funkcionális összehasonlítás már eleve csak részigazságokat hordoz. Még akkor is, ha csak kifejezetten funkcionális és nem képességbeli összehasonlításról van szó.

Végző soron ui. egy számítógép teljesítménye annak fizikailag meghatározott lehetőségein belül attól az utasítássorozattól (programtól) függ, amely az előbbieket a legmesszebbmenőig képes az ember szolgálatába állítani.

A hardver és az azt működtető szoftver szerencsés összhangja az adott gép képességeit messzemenőkig figyelembe veszi, sőt igénybe veszi, és minőségi előrelépés általában csak mindkettő lecserélésével képzelhető el. A tartalom és a forma általános érvényű összhangja itt is alapkövetelmény és biztosítása a hardver, valamint a ráépülő szoftver teljesítőképességének felső határán az emberi kreativitás minden elemét igénybe is veszi.

Ebben a megközelítésben az, hogy a tartalmat megtestesítő program RAM-ba vagy ROM-ba van-e írva, teljesen formális körülmény.

Mitöbb, a nagy kapacitású és gyors háttértárolók birtokában a lehető legkisebb ROM alkalmazása az előnyös, amelynek csak arra kell képesnek lennie, hogy a szükséges programokat a háttértárolóból a memóriába töltsse.

Azonban a személyi számítógépek esetében -- így a TV-Computernél is -- kissé más a helyzet.

Ezek a gépek a felhasználók rendkívül heterogén tábora számára készülnek, ahol gyakori az alapgép szinte önálló alkalmazása. Így indokolt, hogy a jelentős írható memóriaterület biztosítása mellett a gépben olyan fix programot is elhelyezzenek, ami valamilyen magas szintű nyelvvel, a gép fizikai adottságait a legteljesebb módon kihasználó operációs rendszerrel, s hatékony felügyelőrutinokkal ellátva, magas minőségi színvonalon támogassa a felhasználók legszélesebb körének munkáját.

Egy ilyen gépben, mint minden munka szervezője, az ember parancsainak értelmezője és végrehajtója, a gép alapintelligenciájának hordozója, a gép egész működésének funkcionális értelemben vett agya -- valóban a ROM, a beleírt programmal.

Gyakran hallható ellenvetés ezzel szemben azoknak a gépi kódú programoknak az esete, amelyek nagyon intelligensek lehetnek, s működésük a ROM-tól látszólag független.

Nos, először is valóban csak látszólag! Nehezen képzelhető el ui., és valószínűleg felesleges is olyan gépi kódú programokat írni, amelyek a ROM-beli operációs rendszert is teljesen kikapcsolják. (Persze lehet!)

Másrészt örüljünk annak, hogy a ROM irányító szerepét készségesen átengedi a sokszor erre méltatlan programoknak, sőt saját rutinjaival még azok rendelkezésére is áll.

A félreértések egy része persze egyszerűen abból adódik, hogy vannak, akik a ROM-ba írt teljes programot azonosítják annak BASIC értelmező és végrehajtó részével, ami természetesen teljesen helytelen. (Jóllehet sok programozási minta onnan is elleshető és számos fontos feladat programozása e ROM-területhez intézett egyszerű szubrutinhívással elintézhető.)

Minket persze az eddigi előtanulmányaink után ezek a nézeteltérések nemigen érdekelnék.

Sokkal inkább arra kell fordítanunk a figyelmünket, hogy mi és hogyan zajlik le a TV-Computer bekapcsolása után, hogyan is beszélget a ROM az emberrel. Sajnos, ez utóbbi kérdés megválaszolásától még meglehetősen messze vagyunk, ezért visszatérünk a legelső mondatához: előttünk a TV-Computer agya -- a könyv lapjai által képviselt térben és főleg időben --, s végre az a dolgunk, hogy nézzük az első "agysejteket"!

3.1.2 A start

```
C000 C32902      JP      0229      Lehet, hogy elsőre furcsának  
tűnő indulás, de nagyon  
logikus!
```

Ugrás a 0229H címre! Ebben nem az a furcsa, hogy ugró utasítás, hiszen jó megoldásnak tartható, hogy a ROM elején bizonyos fontos táblázatok álljanak, ezeket pedig át kell lépni.

Nem is a cím abszolút értéke a meglepő, hiszen elvileg mindegy, hogy hol folytatódik a végrehajtás.

Meggondolandó azonban, hogy miért egy RAM-cím szerepel az ugrási parancsban!

Ne lapozzon a kedves Olvasó az előző fejezethez! Az U0 RAM-nak ezen a részén nem talál semmi érdekeset (az editor ASCII puffere)!

De nem is erről van szó!

A CPU ezt az utasítást ritkán hajtja végre: bekapcsoláskor vagy ha megnyomjuk a TVC alján található RESET gombot.

Azonban vegyük sorra az eseményeket!

A számítógép bekapcsolása előtt a készülék minden része halott.

A bekapcsolás pillanatában inicializáló impulzus fut át a számítógép minden fontos hardveregységén, ami egyebek mellett a portok minden bitjét is nullázza. Emiatt pedig a 02-es port a SYS-U1-VID-CART memóriákat teszi a CPU elé.

A 0. lapon tehát a ROM 16 kbyte-os főrésze címeződik, utána a 4000...7FFFH tartományban az U1 RAM, a 8000H címtől videomemóriát látja a CPU, a 3. lapon pedig a programmodult, ha van.

Elindult a mikroprocesszor működése. Mivel a program számláló regiszter (PC) értéke is 0, ezért a 0. lap 0000H címén található utasítás olvasódik be, azaz a ROM első három byte-ja.

```
0000 C3 29 02 JP 0229
```

azaz ugrás a 0. lap 0229 címére, ami most a ROM megfelelő byte-ja.

Így már sokkal tisztább a kép.

Megkezdődik a rendszer inicializálása.

Az egész gép üres, ill. a RAM határozatlan tartalmú, tehát a normális működéshez nagyon sok mindent el kell intézni. Elég, ha csak az U0 RAM előző fejezetben leírt tartalmára gondoltunk, azokat most mind meg kell írni. Emellett számos egyéb tennivaló is akad, pl. be kell állítani a képmegjelenítést vezérlő egység (CRT kontroller) összes regiszterét.

Nem vágunk elébe a dolgoknak -- rövidesen módunk lesz az egész inicializáló program megismerésére -- csak felhívtuk a figyelmet az elvégzendő munka fontosságára és sokféleségére.

Ugyanez a helyzet akkor is, ha a számítógép működése közben a készülék alján levő RESET gombot nyomjuk meg.

Ilyenkor a felhasználó kifejezett szándéka a gép alaphelyzetbe állítása. Vagy azért, mert menet közben sikerült a gépet áttekinthetetlenül bonyolult állapotba hozni, s ettől a zűrzavartól akarunk megszabadulni, vagy pedig azért, mert már nincs is más megoldás: a gép irányíthatatlanná vált. Az új programok fejlesztése, a kísérletezések időszakában ez előfordulhat -- jóllehet a gép BASIC-je és operációs rendszere igyekszik megfelelő hibakezeléssel kivédeni katasztrofális tévedéseinket és figyelmetlenségeinket. Ez azonban természetesen csak bizonyos határokon belül lehetséges.

Amikor a gépünk önálló életet kezdett, még segíthet a RESET gomb. Egyszeri megnyomására többnyire visszaállítható a működőképes, friss alapállapot anélkül, hogy memóriába írt korábbi programjaink elvesznének. Az ilyenkor beindult inicializáló program kideríti, hogy miről van szó -- majd látni fogjuk, hogy hogyan - és csak a legszükségesebb alapbeállításokat hozza helyre.

Ha azonban az ügyködéseink révén már az U0 RAM is

jelentős, megengedhetetlen sérüléseket szenvedett, akkor a rendszer a további próbálkozásokat eleve kilátástalannak minősíti és könyörtelenül, a teljes rendszerinicializáló programot hajtja végre. Ekkor már nincs mit tenni! Ezután a számítógép ugyanolyan állapotba kerül, mintha éppen akkor kapcsoltuk volna he.

Mindenképpen tanácsos, hogy bonyolult, hosszadalmas és fáradságos fejlesztési munkáink kipróbálása előtt a beírt programokat mentjük háttértárolóra. A gép teljes eltérése esetén egyszerűen visszatölthetjük a beteg programot és máris folytatható a hibakeresés.

A RESET gomb rövid időn belüli kétszeri lenyomása -- egyébként eleve nagyreset -- a teljes inicializáló program lefutását idézi elő.

Akár így, akár úgy, a rendszer újraindítása mindenképpen a megbeszélte módon kezdődik.

A ROM tehát a 0. lapon van -- a 0000...3FFF címtartományon -- s a processzor ugrik a 0229 címre.

3.1.3 A szorzótáblák

Fussuk át gyorsan, hogy mi van a ROM közbülső címein. (Tekintettel arra, hogy a rendszer működésének túlnyomó részében a ROM a 3. lapon van, a következőkben mindig az ezen a lapon érvényes címeket adjuk meg. Az inicializálás első pillanataiban a címek első hexa-jegye tehát C helyett 0.)

Szorzó táblák 10-es számrendszerben:

C003	00	00	00	00	00	00	00	00	00	00
C00B	00	01	02	03	04	05	06	07	08	09
C017	00	02	04	06	08	10	12	14	16	18
C021	00	03	06	09	12	15	18	21	24	27
C02B	00	04	08	12	16	20	24	28	32	36
C035	00	05	10	15	20	25	30	35	40	45
C03F	00	06	12	18	24	30	36	42	48	54
C049	00	07	14	21	28	35	42	49	56	63
C053	00	08	16	24	32	40	48	56	64	72
C05D	00	09	18	27	36	45	54	63	72	81

Ezt a táblázatot a rendszer a lebegőpontos számolási műveletekhez használja.

Ezután a ROM-ban egy -- a BASIC-hez tartozó -- alapvető jelentőségű táblázat következik.

3.1.4 BASIC előzetes

A BASIC a felhasználó számára lehetővé teszi, hogy a feladatokat a gép számára valóságos emberi szavak - vagy azok meghatározott rövidítései -- formájában fogalmazhassa meg. Az értelmező és végrehajtást vezérlő program (BASIC interpreter) feladata az ilyen szavakból felépített program lebontása a CPU számára végrehajtható elemi utasításokra.

A ROM-nak ez a része, mint majd látni fogjuk, érthetően, meglehetősen nagy terjedelmű.

A használható szavak száma nem túl sok, így lehetőség van arra, hogy elő lépésben minden szónak egy-egy 0...255 közötti számot feleltessünk meg. Ez rövidebb és egyszerű kezelhetőséget biztosít. Ezeket a számokat nevezzük tokeneknek (token=szimbólum, jelkép).

A BASIC-ben írt programok mondatokból állnak, amelyeket kettőspont választ el egymástól. A mondatok bekezdéseket alkotnak, s egy-egy bekezdést sorszámmal látunk el. Ez a szerkezet nagyon szigorú követelmény.

Ha egy bekezdést sorszám nélkül írunk be, akkor a végét jelentő RETURN lenyomása után az interpreter azonnal hozzálát a beírt mondatok értelmezéséhez és tüstént végre is hajtja a benne előírt feladatokat. A sorszámmal ellátott bekezdéseket ezzel szemben egy program részének tekinti és csak tárolja a memória megfelelő részén.

A mondatok első szavai mindig megkülönböztetett szerepet töltenek be: ezek határozzák meg a mondatban előírt feladat típusát. A mondat többi része pontosítja, kiegészíti a megnevezett funkciót. Ezeket a kezdőszavakat, ill. a nekik megfelelő számszimbólumukat a továbbiakban elsődleges tokeneknek nevezzük, a mondatban előforduló egyéb, BASIC-hez tartozó utasításokat pedig másodlagos tokeneknek. Ez utóbbiak önmagukban sohasem értelmesek, csakis egy elsődleges token után állhatnak.

Például a

"SOUND PITCH 2000, VOLUME 11, DURATIUN 9"

mondat egy hang előállítását kéri. Ezt a funkciót az első szó jelöli ki (sound=hang megszólaltatás, az angol nyelvet beszélők számára az egész BASIC szinte eleve triviális). A mondat többi része a hangmagasságot (pitch), hangerőt (volume) és az időtartamot (duration) specifikálja.

A ROM a billentyűzetről beolvasott előbbi karaktereket először az ennek pontosan megfelelő:

D5 20 BB 20 32 30 30 30 A4 20 B3 20 31 31 A4 20 C6 20 39

rövidített formára hozza. Ebben D5: a sound-ot jelentő elsődleges token; BB, B3, C6 és A4: a pitch, a volume, a duration és a vessző tokenje; a többi karakter pedig a betűközök (20H) és a számok ASCII kódja.

Végrehajtáskor az elsődleges token alapján választja ki a rendszer azt a programrészt, amelyet a CPU-nak végre kell hajtania a feladat megoldásához. A ROM itt következő táblázata az ehhez tartozó ugrótáblát tartalmazza.

Az elsődleges tokenek ugrótáblája:

ROM-cím	Rutin-cím	Szó,jel	Token	ROM-cím	Rutin-cím	Szó,jel	Token
C067	DBBB	!	FF	C097	E452	LOMEM	E7
C069	DBBB	!	FE	C099	DE08	NEW	E6
C06B	DB80	:	FD	C09B	E4B6	NEXT	E5
C06D	DBBB	REM	FC	C09D	DB06	OK	E4
C06F	DFF2	DATA	FB	C09F	E332	ON	E3
C071	E89B	CLOSE	FA	COA1	E8C9	OPEN	E2
C073	DFFC	CLS	F9	COA3	E573	OUTPUT	E1
C075	DD65	CONTINUE	F8	COA5	E542	OUT	E0
C077	E002	DEF	F7	COA7	E754	PLOT	DF
C079	DD93	DELETE	F6	COA9	E553	POKE	DE
C07B	E053	DIM	F5	COAB	E573	PRINT	DD
C07D	E104	ELSE	F4	COAD	E6C7	RANDOMIZE	DC
C07F	E10E	END	F3	COAF	E21B	READ	DB
C081	E15C	FOR	F2	COB1	E6F4	RESTORE	DA
C083	E910	GET	F1	COB3	E70F	RETURN	D9
C085	E382	GOSUB	F0	COB5	DE1B	RUN	D8
C087	E3B2	GOTO	EF	COB7	E982	SAVE	D7
C089	E733	GRAPHICS	EE	COB9	E790	SET	D6
C08B	E2EE	IF	ED	COBB	E833	SOUND	D5
C08D	E1CB	INPUT	EC	COBD	FFA3	STOP	D4
C08F	E3C1	LET	EB	COBF	DE31	TRACE	D3
C091	DD85	LIST	EA	COC1	E9D3	VERIFY	D2
C093	DD80	LLIST	E9	COC3	E117	EXT	D1
C095	E951	LOAD	E8	COC5	E570	LPRINT	D0

A kulcsszavak azonosítását végző programrész - amint majd látni fogjuk -- a tokenekből igen egyszerű eljárással eljut ehhez a táblázathoz, majd a megadott kezdőcíme ugrik, ahol a tokenhez tartozó feladatot megvalósító gépi kódú utasítássorozat található. Ezekben a programrészekben ellenőrzi a rendszer a feladat pontos

értelmezését jelentő másodlagos tokenek és egyéb paraméterek meglétét, valamint azok helyességét is.

Megjegyezzük, hogy a tokenek bevezetése most már a harmadik példa arra, hogy a CPU tiszta gépi kódú utasítássorozata és a magasabb szintű nyelvek (pl. BASIC) közé egy közbülső nyelvet hozunk létre.

Ilyen volt az RST 30 egybyte-os szubrutinhívásra épített funkcióhívások esete, amely -- mint már megbeszéltük -- a fizikai és logikai eszközökkel kapcsolatos leggyakoribb feladatok megoldását teszi lehetővé nagyon rövid, kétbyte-os hivatkozásokkal.

Már említettük, de rövidesen ismét beszélünk az RST 18-ra épített közbülső nyelvről, amely logikailag hasonló szerepet tölt be, mint az említett másik kettő.

Szerepüket tekintve ezek a megoldások egy-egy nagyszerű, közbülső állomásokat jelentenek azon az úton, amelyek révén végül is fölé emelkedhetünk a részleteknek. Általuk nem kell többé a CPU rendkívül szorgos és az ember számára folyamatosan követhetetlenül aprólékos munkáját szem előtt tartanunk, erőinket átcsoportosíthatjuk a minket érdeklő feladatok megoldására.

Amíg azonban a funkcióhívások a számítógép operációs rendszerével, a tokenek pedig a ROM BASIC értelmezőjével kapcsolatos hivatkozások -- tehát meglehetősen speciális funkciók -- szolgálatában állnak, addig a most következő táblázat kapcsán ismertetésre kerülő közbülső nyelv már gyakorlatiasabb feladatok megoldását segíti.

3.1.5 A BASIC veremről

Mielőtt továbbmennénk, a későbbiek megértéséhez ismét tisztáznunk kell néhány alapvető fogalmat és a TV-Computerben alkalmazott megoldást.

A gép a BASIC programok végrehajtása közben keletkező részeredményeket -- szövegeket, részletszámításokat, ugrási és visszatérési címeket, ciklusparamétereket, stb. -- egy külön RAM-területen: a BASIC veremben tárol. Ez a terület a BASIC program működése szintjén funkcióját tekintve sokban hasonlít a mikroprocesszor verem területéhez. Ám amíg a processzor vereme CPU szintű adatkezelési célokat szolgál -- kizárólag 16 bites adatok átmenetű tárolása --, addig a BASIC veremben hosszú szövegek, számláncok tárolása folyik, benne bonyolult műveletek zajlanak.

Tekintsük át a BASIC veremmel kapcsolatos legfontosabb tudnivalókat.

Ez a terület mindig a használható legmagasabb RAM-címen kezdődik, s mint a processzor verem, az is lefelé épül. Hibátlan RAM-okat feltételezve, a 64 kbyte-os gépek esetén tehát a BASIC verem kezdőcímer 0BFFFH. Címmutatóként a TV-Computer a mikroprocesszor IY regiszterét használja, amely így a veremben utoljára használt memóriarekeszre mutat.

A verembe kerülő elemek most nagyon különböző méretűek, tartalmúak és funkciójúak, ezért azokat megfelelő jelekkel azonosítani kell. Minden elem legalacsonyabb című byte-ja egy ilyen azonosító.

Majd láthatjuk, hogy a BASIC nagy körültekintéssel adminisztrálja ezt a területet, hiszen működése közben ez alapvető fontossága információk gyűjtőhelye és forrása. Ezt a tényt a felhasználónak is illik és érdemes tudomásul vennie: a BASIC -- amikor csak szóhoz jut - valószínűleg használja (átírja!) az IY mutatta memóriaterületet!

A BASIC veremmel kapcsolatos tudnivalókat, az elemek azonosítóit és szerkezetét a "BASIC programozási segédlet" tartalmazza.

Példaként tegyük fel, hogy a BASIC veremben a következőt találjuk:

09 01 90 78 61 57 32 00 44

és IY a 09-re mutat. Mi ez az elem?

A 09-ből következik, hogy számról van szó. Elmegyünk tehát a 6. byte-ig, s az első jegyet egésznek, a többi tizedes jegyek véve leírjuk a szám ún. mantisszáját:

3.25761789001

Ezután megnézzük a 8. byte-ot. Kivonunk belőle 40H-t (vagy decimális értékéből 64-et), s amit kapunk, az előző számot 10 annyiadik hatványával kell szorozni. Példánkban ez 4, tehát az előbb felírt számot 10 000-rel kell még szorozni. A veremben levő szám tehát:

32576.1789001

Ez a számábrázolás viszonylagos egyszerűsége mellett igen nagy tartományban teszi lehetővé a számok kezelését, s ugyanakkor pontossága is imponáló.

3.1.6 Ismerkedés az RST 18 elemeivel

A BASIC verem kezelésére, az ott elhelyezett számok mozgására, közöttük meghatározott műveletek elvégzésére szolgál -- speciális szimbolikus nyelvként -- az RST 18 hívásokra épített utasításstruktúra, amelyet programjainkban mi is bátran használhatunk.

Az RST 18 utasítást követő byte-ra egy rutinsorszámot írva a rendszer felismeri a kért hivatkozást, s a sorszámmal meghatározott feladatot hajtja végre. Az így meghívott rutinok sok esetben meglehetősen bonyolultak -- mindegyiket részletesen megismerjük. Hívásuk történhetne természetesen CALL utasításokkal is, azonban az egybyte-os hivatkozás lehetősége még egyszerűbb programozást tesz lehetővé: kisebb a memóriaigény és valamivel javult az áttekinthetőség is.

Az RST 18-ra épített struktúra alapvető sajátossága, s ebben már jelentősen továbblép az RST 30-cal működő funkcióhívásokon, hogy az RST 18 után nem csak egyetlen sorszám megadására van lehetőség. Az egymást követő byte-okon a rutinsorszámok tetszőlegesen hosszú sorozatát helyezhetjük el: mondhatjuk, hogy a tiszta gépi kódú programba egy logikailag magasabb szintű nyelvet ékelhetünk be -- hozzá kell tenni természetesen, hogy ezek hatóköre lényegesen kevésbé általános, mint a CPU elemi utasításkészlete.

Az RST 18 után írt rutinsorszámokat, funkcióbyte-okat azzal zárjuk le, hogy az utolsó funkcióbyte 7. bitjébe 1-et írunk.

Az elsődleges tokenek ugrótáblája után a ROM-ban az RST 18-cal hívható rutinok ugrótáblája következik. A következőkben megadjuk a ROM-címet, a funkcióbyte-ot, a végrehajtó rutin kezdőcímét, valamint a funkció későbbiekben használt rövidítését és leírását.

A BASIC veremben elhelyezett számokat az n1, n2 szimbólumokkal jelöljük, az indexek az elhelyezés sorrendjét is megadják, tehát az n2 az utoljára elhelyezett adat.

Ha csak n-t írunk, az mindig a verembe utoljára tett számot jelenti.

X és Y a két aritmetikai regiszter.

Di az i-edik konstanst jelenti a ROM 0C111...C228H területén tárolt számok közül.

A két számmal végzett műveletek után mindkét eredeti érték elveszik, a veremben csak az eredmény marad és IY az eredményre mutat.

Ha a funkció leírásában az = szerepel, akkor ez azt jelenti, hogy az utána álló elem a veremből elveszik.

Egy elem puszta leírása az illető elem verembe való betöltését jelenti.

Az alább ismertetett funkcióbyte-okkal megvalósítható számítások, ill. inkább ezek kivitelezésének formája sok tekintetben hasonlít a FORTH programozási nyelv logikájához. Ha Ön, kedves Olvasó ismeri ezt a nyelvet, akkor a TV-Computer BASIC funkcióbyte-okkal megvalósított programrészeit könnyebben fogja tanulmányozni.

A BASIC verem RST 18-cal hívható alpműveletei:

ROM-cím	Funkció-byte	Ugrási cím	Funkció-szimb.	Leírás
C0C7	00	F493	+	$n = n1 + n2$
C0C9	01	F5FB	/	$n = n1 / n2$
C0CB	02	F512	*	$n = n1 * n2$
C0CD	03	F48E	-	$n = n1 - n2$
C0CF	04	F726	NEG	$n = -n$
C0D1	05	EA82	Di	Az i. számot teszi a verembe. Az indexet egyszerűen a funkcióbyte után kell írni
C0D3	06	EA9F	X	X értékét teszi a verembe
C0D5	07	EA9A	Y	Y értékét teszi a verembe
C0D7	08	EAD2	SX	n-t X-be másolja
C0D9	09	EACD	SY	n-t Y-ba másolja
C0DB	0A	EAC3	SXD	SX, majd n-t törli
C0DD	0B	EABE	SYD	SY, majd n-t törli
C0DF	0C	FA92	DUP	megismétli a számot
C0E1	0D	FB28	VAR	n-t a szimbólumtáblába másolja
C0E3	0E	EA68	NUM	egy zárójeles kifejezés értékét teszi a verembe

Ezek a műveletek -- mint majd látni fogjuk -- a különböző beépített függvények értékének kiszámításánál tesznek nagyon jó szolgálatot.

Nagyon lényeges ugyanakkor, hogy a felhasználó saját programjaiból is egyszerűen hívhatja ezeket a rutinokat és segítségükkel kis memóriaigényű, bonyolult, s mégis jólstrukturált számításokat végezhet.

A következő néhány byte a fejlesztő cég szerzői jogára emlékeztet, ASCII kódú szöveggel:

```
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
COEO                      43 6F 70 79 72 69 67 68 74 20 28
COFO 63 29 20 31 39 38 34 20 20 49 6E 74 65 6C 6C 69
C100 67 65 6E 74 20 53 6F 66 74 77 61 72 65 20 4C 74
C110 64
```

azaz: "Copyright (c) 1984 Intelligent Software Ltd".

3.1.7 Számkonstansok

A ROM C111...C228H című byte-jai a különböző függvények kiértékeléséhez használt konstansokat tartalmazzák abban a formátumban, amelyet a BASIC veremmel kapcsolatban megbeszéltünk. Egyetlen eltérés, hogy a két túlcsorduló számjegy számára fenntartott byte itt -- célszerűen -- nem szerepel.

Az RST 18 hívásokkal megvalósított veremműveletek korábban leírt szimbólumai közül a Di ezen konstansok egyikét jelöli ki.

A következőt táblázatban a ROM-címet, a megfelelő Di szimbólumot, a lebegőpontos BCD számot tartalmazó byte-okat, a szám értékét és esetenként egy-egy rá vonatkozó megjegyzést bocsát az Olvasó rendelkezésére.

A számok értékének leírásában használt E szimbólum azt jelenti, hogy a közölt értéket 10 annyiadik hatványával kell szorozni (10-es alapra vonatkozó exponens), tehát például:

```
1.13 E 2 = 113
3.14 E-3 = 0.00314
```

Kontansok

ROM- cím	Dn	BCD								Érték
C111	D0	00	00	00	00	00	50	3F	0.5	
C118	D1	00	00	00	00	00	10	40	1	
C11F	D2	31	24	19	49	79	26	3F	0.267949192431	
C126	D3	57	07	08	05	32	17	40	1.73205080757	
C12D	D4	69	75	80	50	20	73	3F	0.732050807569	
C134	D5	74	48	34	08	40	14	C0	-1.44008344874	
C13B	D6	98	88	84	26	00	72	BF	-0.720026848898	
C142	D7	19	89	03	25	20	43	40	4.32025038919	
C149	D8	99	45	58	22	52	47	40	4.75222584599	
C150	D9	07	38	96	88	85	86	3F	0.868588963807	
C157	D10	00	00	00	00	51	11	40	1.151	
C15E	D11	23	70	49	46	25	29	3C	2.92546497023 E-4	
C165	D12	06	95	88	64	44	50	42	504.464889506	
C16C	D13	63	75	99	82	00	14	41	14.0082997563	
C173	D14	16	65	64	73	28	33	3E	3.32873646516 E-2	
C17A	D15	01	79	97	92	08	10	43	1008.92977901	
C181	D16	97	10	08	94	20	11	42	112.094081097	
C188	D17	99	92	50	58	02	23	40	2.30258509299	
C18F	D18	90	37	14	68	15	29	C0	-2.91568143790	
C196	D19	57	15	49	03	63	31	40	3.16303491557	
C19D	D20	78	14	60	81	35	67	BF	-0.673581601478	
C1A4	D21	42	95	06	04	07	10	C1	-10.0704069542	
C1AB	D22	21	40	81	69	96	16	41	16.9669814021	
C1B2	D23	67	54	04	80	90	81	C0	-8.19080045467	
C1B9	D24	07	38	96	88	85	86	3F	0.868588963807	
C1C0	D25	88	60	66	66	66	16	BF	-0.166666666088	
C1C7	D26	56	20	07	33	33	83	3D	8.33333072056 E-3	
C1CE	D27	31	82	32	08	84	19	BC	-1.98408328231 E-4	
C1D5	D28	78	06	71	39	52	27	3A	2.75239710678 E-6	
C1DC	D29	60	40	46	83	86	23	B8	-2.38683464060 E-8	
C1E3	D30	00	00	00	07	36	22	3F	0.223607	
C1EA	D31	00	00	00	27	44	89	3F	0.894427	
C1F1	D32	17	60	76	27	62	31	3F	0.316227766017	
C1F8	D33	31	51	79	57	29	57	41	57.2957795131	
C1FF	D34	59	53	26	59	41	31	40	3.14159265359	
C206	D35	79	26	63	79	70	15	40	1.57079632679	
C20D	D36	20	51	75	19	47	10	40	1.04719755120	
C214	D37	98	55	77	98	35	52	3F	0.523598775598	
C21B	D38	00	00	00	80	76	32	44	32768	
C222	D39	00	99	99	99	99	99	7E	9.999999999 E 62	

Ez a 40 konstans a különböző beépített függvények értékeinek nagy pontosságú számításához szükséges.

Nem bocsátkozunk annak részletes taglalásába, hogy miért éppen ezekkel történik a számítás, mert nagyon messzire vezetne. Az egyes függvények számolási eljárásainak tanulmányozásánál módunk lesz rá, hogy néhány ezzel kapcsolatos problémára rámutassunk, de az egyes függvényekre vonatkozó matematikai eljárások elméletének taglalása túlnőne e könyv keretein, s alapjában véve nem is feltétele a számítógép működése jobb megértésének.

Ezek már a nyitány utolsó hangjai.

Sok mindenről szó esett ebben a részben. Beszéltünk a rendszert indító inicializálásról, kifejtettünk a ROM-mal kapcsolatos néhány általános gondolatot stb.

Most már a mű kezdetén állunk.

Olyan tételek következnek, amelyekben már ott vibrál a rendszer teljes működésének mindenre kiterjedő, mindent megmozgató dinamizmusa.

Íme: a kezdet.

3.2 A rendszerinicializálás

3.2.1 Az első gombnyomás előtt

Ebben a részben még nem jutunk el az első gombnyomásig. Azonban ennek már csak bizonyos formai okai vannak.

A ROM maga nem egybefüggően tartalmaz bizonyos funkciókhoz tartozó programrészeket. Mégpedig azért, mert egy-egy utasítássorozatra a működés más fázisaiban is sor kerül. A program a ROM különböző részein -- sőt a SYS mellett az EXT-ben is -- váltakozva fut. Mi ezt a könyv lapjain ebben a formában nem tudjuk követni.

A szigorúan funkcionális működési sorrend követése -- mint szempont -- sokszor érvényesíthető lenne és a megfelelő részletekről logikusabb képet festene -- mégsem tesszük ezt.

Egyrészt -- mert ez a koncepció elvileg sem lenne maradéktalanul keresztülvihető. A működés közben adódó, sokszor nagyszámú feltételhez kötődő elágazások időben egymással párhuzamosan futó lehetőségeket jelentenek. A ROM-ban futó program e sokdimenziós világát a könyv egymást tanulmányozása során ismerjük meg.

követő soraiban és oldalain, egyetlen logikai dimenzióban nem lehet szemléltetni!

Másrészt -- s ez már gyakorlati kérdés -- a működést még lineárisnak feltételezve sem lehetne a rendszer egész munkáját a funkciók kivitelezésének sorrendjében végigkövetni, mert ez sok-sok programrész megengedhetetlen ismételtetésével járna.

Szerencsére a helyzet az, hogy a számítógép megismerésének nem ez az egyetlen, s ráadásul nem a leghatékonyabb módja.

Az alkotó és kombináld emberi értelem már eleve több dimenzióssá domborítja az elé táruló, egysíkúvá alakított logikai képet.

Azt a megoldást választjuk tehát, ami a legegyszerűbb és az előbbieket tudomásul véve még némiképp logikus is: a ROM növekvő címei felé haladunk, s a más területekre történő átmeneti hivatkozásoknál igyekszünk az ott elvégzett munkáról globális ismertetést adni.

Így legalább a címek növekvő sorrendje egyértelmű folyamatát adja munkánknak. Az Olvasó viszonylag könnyen megtalálja az Őt érdeklő részleteket, de az ott található hivatkozások egyben felhívják figyelmét a más területekkel fennálló összefüggésekre is.

A ROM első részének tanulmányozásakor láttuk, hogy benne elsősorban többé-kevésbé fontos táblázatok, adatok találhatóak. Mégsem egy merev, passzív terület ez: gondoljunk az ugrótáblákra. Igaz, hogy tényleges, fizikai munka itt nem folyik, de ezek a számítógép belső eligazító pontjai, vezérlőtermei. Amikor már tisztázódik, hogy mi a soros, végrehajtandó feladat, a CPU innen kapja meg azokat a címeket, ahol a munkát ténylegesen le kell bonyolítani.

Az indulás mindenesetre nagyon energikus volt: ugrás a 0229H címre!

Most innen nézzük tovább, s ettől kezdve már keményebb munka vár ránk.

3.2.2 Az inicializálás

Ott tartunk, hogy most kapcsolták be a gépet, vagy pedig már be volt kapcsolva, de éppen megnyomták a RESET gombot. A 02-es port minden bitje 0, tehát a ROM a 0000. címre került. Az 1-es lapon a CPU az U1 RAM-ot, a 8000H címtől a videomemóriát látja, a felső címeken pedig a gép bal oldalán bedugaszolt memóriát -- ha van.

Következik az inicializálási.

Tisztázni kell, először is, hogy mi okozta ezt az állapotot. Ha első bekapcsolásról van szó, akkor rengeteg a tennivaló. Ha csak egyszer nyomták le a RESET-et, akkor némi ellenőrzés és beállítás után mehet minden tovább. Ám ha kétszeri RESET volt, újra kell indítani az egész gépet. Figyelni kell, mert a RESET-et most is, közben is újra megnyomhatják!

Rendszerinicializálás belépési pont.

C229 F3	DI		Egyelőre tiltja a
C22A ED56	IM	1	megszakításokat és rááll a véglegesen használt IT-módra
C22C 3E40	LD	A,40	A 3. lapra is a SYS ROM kerül,
C22E D302	OUT	(02),A	tehát a konfiguráció SYS-U1-VID-SYS
C230 C33302	JF	0233	Ez a következő sor a 0. lapon.
C233 3E00	LD	A,C0	Ezután a 3. lapra az EXT-t teszi és ugrás az EXT ROM-ba!
C235 D302	OUT	(02),A	
C237 C33DF1	JF	F13D	

A rendszer itt pár soron tisztázza, hogy milyen állapotban került sor az inicializálásra:

-- a 0. lapra átmenetileg az U0-t hozza és megvizsgálja a funkcióhívásokat kezelő utasítássorozatot hibátlan meglétét (bekapcsoláskor természetesen nincs ott);
-- értékeli a 0B21 címen levő WARM-FLAG tartalmát.

Az eredményt a WARM-FLAG-ben és az A' regiszterben adja vissza:
-- ha az U0 ép és korábban még nem volt reset állapot, akkor (0B21)=FF és A'=FF: meleg reset lehet;
-- ha az U0 tartalma nem megfelelő vagy már ezelőtt is reset állapot volt, akkor pedig (0B21)=00 és A'=00, tehát hideg reset kell.

A korábbi reset állapotot egyébként éppen az adja, ha a WARM-FLAG-ben már a vizsgálat elején FF van. Ez a körülmény éppen azt jelenti, hogy a CPU már másodszor jár ezen a programrészen, tehát a RESET gombot meleg reset közben másodszor is megnyomták. Az inicializálás után ui. a rendszer a WARM- és COLD-FLAG-et nullázza.

Végül az EXT-ben futó programrész a 0. lapra visszateszi a SYS-t és a munka ugyanitt folytatódik.

C23A 3E40	LD	A, 40	A SYS ROM-ot a 3. lapra is
C23C D302	OUT	(02), A	felteszi és már a
C23E C341C2	JP	C241	3. lapon ugrik a következő
			sorra
C241 3E50	LD	A, 50	U0-U1-VID-SYS konfiguráció,
C243 D302	OUT	(02), A	tehát behozza U0-t
C245 AF	XOR	A	
C246 016004	LD	BC, 0460	A 60H...63H portokra 00,
C249 ED79	OUT	(C), A	tehát minden
C24B 0C	INC	C	palettaregisztert feketére
C24C 10FB	DJNZ	C249	állít
C24E 2145C5	LD	HL, C545	Miközben a képernyő teljesen
C251 0610	LD	B, 10	fekete, feltölti a
C253 78	LD	A, B	képmegjelenítést vezérlő
C254 3D	DEC	A	egység (CRT) minden
C255 D370	OUT	(70), A	regiszterét a 0C545H címtől
C257 7E	LD	A, (HL)	kezdvé tárolt alapérté-
C258 D371	OUT	(71), A	kekkel.
C25A 23	INC	HL	Figyeljük meg a feltöltés
C25B 10F6	DJNZ	C253	technikáját!
C25D 3E80	LD	A, 80	Alaphelyzetbe állítja a
C25F D306	OUT	(06), A	nyomtató adatérvényesítő
			(STRB) jelét és a hangokat
			tiltja (0 hangerő)

C261 3A220B	LD	A,(0B22)	COLD-FLAG. FF esetén hideg reset kell
C264 3C	INC	A	Valóban az?
C265 2004	JR	NZ,C26B	Ha nem: továbblép, ha igen: beállítja a nem meleg reset jelzést,
C267 32210B	LD	(0B21),A	
C26A 3E08	LD	A,08	
C26C 3C	INC	A	s mivel így A=09 lett, nem ugrik el tehát folytatja a hideg reset ággal
C26D 284C	JR	Z,C2BB	

Figyeljük meg: ha a 0B22 COLD-FLAG-ben eredetileg nem 0FFH van, tehát ez önmagában nem teszi kötelezővé a hideg resetet, akkor ugyanez a program teljesen másképpen fut le. Akkor a C264-re írt INC A eredménye nem lesz 00, így a CPU a C26B címre, tehát az előbbi LD A,08 utasítás közepébe, annak 08 tartalmú byte-jára ugrik. Ez lesz az utasításkód, s ugyanaz a program most így fest:

C26B 08	EX	AF,AF'	Vesszük tehát az EXT-ben előkészített A'-t, így a továbbiakban a reset minősége attól függ, hogy milyen volt az U0 állapota és a WARM-FLAG
C26C 3C	INC	A	
C26D 284C	JR	Z,C2BB	Ha A'=FF volt, tehát a meleg reset valóban mehet, akkor ugrás előre!

A hideg resethez tartozó memóriaellenőrzések következnek. A 0C33EH című szubrutin a HL regiszterpárban megadott kezdőcímű RAM szegmenst teszteli ismételt írásokkal, olvasásokkal, s hibátlan szegmens esetén Z=1-et állít be, a HL-ben pedig minden esetben az utolsóként jónak talált memóriarekesz utáni címet adja vissza.

C26F 31FFBF	LD	SP,BFFF	A vermet egyelőre a VID tetejére tesszük, hiszen az U0-t kelt tesztelni
C272 210000	LD	HL,0000	
C275 0B3EC3	CALL	C33E	Az U0 jó?
C278 2805	JR	Z,C27F	Ha igen: tovább!
C27A 3D	DEC	A	Amennyiben az U0 hibás, a rendszer működése lehetetlen.
C27B D300	OUT	(00),A	A program megáll, a bajt a borderszín váltakozó villogása jelzi
C27D 18FB	JR	C27A	

Ha az U0 RAM jó, a memóriateszt folytatódik:

C27F 31AC16	LD	SP,16AC	Most már a verem a helyére kerülhet
C282 210080	LD	HL,8000	A videomemória vizsgálata következik.
C285 CD3EC3	CALL	C33E	Ha a VID is jó: tovább!
C288 2804	JR	Z,C28E	A videomemória hibájára vörös border figyelmeztet. Nagy baj ez is, de végül is nem jelent teljes működésképtelenséget
C28A 3E88	LD	A,88	Ez már U0-U1-U2-SYS
C28C ED79	OUT	(C),A	alapkonzfiguráció
C28E 3E70	LD	A,70	Ellenőrzi U1-et, majd ha U1
C290 D302	OUT	(02),A	jó, még U2-t is, s az így
C292 210040	LD	HL,4000	kiadódó legmagasabb használ-
C295 CD3EC3	CALL	C33E	ható RAM címet beírja HI-MEM-
C298 CC3EC3	CALL	Z,C33E	be
C29B 2B	DEC	HL	
C29C 22190B	LD	(0B19),HL	

Hátra van még az U3 RAM.

C29F 3E40	LD	A,40	Ehhez először SYS-t a 0. lapra
C2A1 D302	OUT	(02),A	kell tenni
C2A3 31FFBF	LD	SP,BFFF	A 0. lap foglaltsága miatt a
C2A6 C3A902	JP	02A9	verem a VID-be kerül, majd
C2A9 3E80	LD	A,80	ugrás a következő sorra, de a
C2AB D302	OUT	(02),A	0. lapon
C2AD 2100C0	LD	HL,C000	így a 3. lapra behozható az U3
C2B0 CD3803	CALL	0338	A szokásos tesztelés, aminek
C2B3 3E00	LD	A,00	eredményét az A' hordozza:
C2B5 2801	JR	Z,C2B8	A'=00, ha jó, A'=FF, ha az U3
C2B7 3D	DEC	A	rossz
C2B8 08	EX	AF,AF	Visszatér az A-hoz.

Ezzel a memóriatesztek lezárultak.

C2B9 1807	JR	C2C2	Átlépi a meleg resethez tartozó néhány sort.
-----------	----	------	--

Ha meleg reset volt, akkor a memóriatartalmat is törlő tesztek elmaradnak, a program itt folytatódik:

C2BB 67	LD	H,A	Ez egyszerű számlálás a HL-
C2BC 6F	LD	L,A	ben: 65536-tól visszafelé: 0-
C2BD 2B	DEC	HL	ig.
C2BE 7C	LD	A,H	Kb. fél másodpercig arra
C2BF B5	OR	L	várunk, hátha még egyszer
C2C0 20FB	JR	NZ,C2BD	megnyomják a RESET-et

Az inicializáló program ezután a hideg és meleg reset esetén ugyanúgy fut tovább.

C2C2 3E40	LD	A,40	A SYS-U1-VID-SYS lapozás csak
C2C4 D302	OUT	(02),A	a hideg resetnél jelent
			ismétlést
C2C6 C3C902	JP	02C9	Ez a 0. lapon a következő sor.
C2C9 3EC0	LD	A,C0	EXT-t hozzuk a 3.lapra. Úgy
C2CB D302	OUT	(02),A	tűnik, hogy a verem a ROM-ba
C2CD 31AC16	LD	SP,16AC	került, ám a program az EXT-
C2D0 C300F0	JP	F000	ben folytatódik, s ott a 0.
			lapra már természetesen RAM-ot
			tesz

Az EXT ROM-ban az U0 feltöltésével, majd a csatolókétyák és a programmodul csatlakozóhely vizsgálatával folytatódik az inicializáló program. Munkája nagyvonalakban a következő:

- Ha hideg reset volt, akkor 0B1B-re (U3-STAT) beíródik az A'-ből az U3 állapotát jellemző érték, 00: jó, FF: rossz.
- Átmásolódik U0-ba a definiálható karakterek közül az ékezetes betűk és a félgrafikus karakterek mátrixa (32 db), a többi helyét nullázza (64 db).
- A program az EXT ROM-ból az U0-ba másolja az input-output funkciók hozzárendelési tábláját.
- Ezután a 0030...003FH területre a funkcióhívások és az IT-kezelő rutin belépési utasításait írja.
- 0B23H-től a fenti rutinok U0 RAM-ba tartozó részét másolja át.
- A kurzor-IT által kiszolgálható eszközökként a video-kezelőt és a billentyűzetet jelöli meg és csak a kurzor-IT-t engedélyezi (0B10 és 0B1F rendszerváltozók).
- A 06-os port kópiájába is betölti a 80H-t (hangerő: 0, nyomtató STROBE jel nyugalmi szintje: 1).
- A 0B17H változóba betölti a verem képkitöltésnél megengedett alsó határát: 0F10H-t.

Ezután a csatolókétyák vizsgálata következik:

- Az 5AH-s portról beolvasott kártyaazonosítók alapján feltölti a 4 csatoló számára biztosított 0040H, 0070H, 00A0H és 00D0H kezdőcímmű I/O pufferek elejét:
 - (a) azonosított interface-kártyák esetén a megfelelő névhosszbyte után írt RS232 (soros vonal), VGB (játék modul) vagy DISK (floppy csatoló) nevekkal;

(b) bekötéssel nem azonosított (vagy üres) csatlakozó esetén ellenőrzi a kártyán elhelyezett memóriában a MOPS szó jelenlétét, s ha megtalálta, akkor az utána megadott azonosító névvel;

(c) azonosítatlan kártya és a MOPS szó hiánya esetén pedig egy 0FFH érték beírásával.

- Ha soros vonal van a rendszerhez csatlakoztatva, akkor a funkcióosztály -- I/O hozzárendelési táblába a csatolókárttyákhoz ezt rendeli hozzá (ha több soros vonali kártya is van, akkor a legkisebb ilyen csatlakozó sorszámát írja be). Egyúttal a bővítőkárttya alaphozzárendelést tartalmazó rendszerváltozóba (0B1CH) is bejegyzi ezt az értéket. Ha nincs soros vonal, akkor a beírt érték: 0FFH.
- A bővítők számára biztosított 0040H, 0070H, 00A0H és 00D0H I/O RAM-területek 8. byte-jaira feljegyzi az egység számokat: ez különböző típusú kártyák esetén 00, több azonos kártya esetén 00...03 lehet.
- Inicializálja a soros vonalat. Ez abból áll, hogy az átviteli sebességet a (0B69H) BAUD változóban 04-re (1200 baud) állítja, az USART üzemmódját meghatározó (0B6AH) FORMAT-ba pedig a 0EEH értéket írja be.
- Ezt követően a többi azonosított eszköz inicializáló rutinját hívja meg. E rutinokat maguk az eszközök tartalmazzák. Előírás szerint a 3. lapra belapozott memóriájuk 0C00BH címén kell, hogy elhelyezzék a saját inicializáló rutinjuk kezdőcímét. Az azonosított kártyák memóriáit a rendszer a 3. lapra hozza, s a C00BH-n talált cím alapján sorra meghívja ezeket a rutinokat. Világos, hogy a működőképesnek szánt rendszerek kötelesek e formai követelményeknek eleget tenni.

Az összes eszköz inicializálása után az EXT ROM-ban futó program visszaállítja az U0-U1-U2-SYS memórialapozási alapkonfigurációt, majd visszatér a ROM fő részébe, ahol a programvégrehajtás a 0C2D3H címen folytatódik.

A leírt részletek iránt a program utasításai mélységéig érdeklődő Olvasó a 3.7.1 ponthoz lapozva az előbbi és az ott található leírás alapján könnyűszerrel kielégítheti kíváncsiságát.

Az inicializálás ezután a ROM fő részében folyik tovább.

C2D3 AF	XOR	A	
C2D4 32110B	LD	(0B11),A	A 03-as port kópiája
C2D7 D303	OUT	(03),A	Alaphelyzet
C2D9 D307	OUT	(07),A	A kurzor/hang IT-t nyugtázza,
C2DB D358	OUT	(58),A	a csatolókarttyák
C2DD D359	OUT	(59),A	megszakításkérő jeleit pedig
C2DF D35A	OUT	(5A),A	törli, ill. le is tiltja
C2E1 D35B	OUT	(5B),A	
C2E3 D300	OUT	(00),A	Fekete border, ezt már tárolja
C2E5 324F0B	LD	(0B4F),A	is.
C2E8 3E80	LD	A,80	Hangerő és nyomtató alap-
C2EA D306	OUT	(06),A	helyzet, most már a 06-os
C2EC 32130B	LD	(0B13),A	kópiába is.

Ezután a beépített eszközök inicializáló rutinjait hívja meg a rendszer. Itt is figyelemre méltó az az eljárás, ahogyan az egymástól független, az egyes eszközmeghajtókhoz tartozó négy szubrutint aktivizáljuk.

A C555 címen egy speciális ugrótábla kezdődik. Ennek első 8 byte-ján a video-, a billentyűzet-, a hang- és a magnetofon-inicializáló rutinok címei találhatóak. Az ugrótábla felhasználásával a 4 különböző, nem azonos lapozási konfigurációban elérhető rutin egységesen, közönséges ciklusba szervezve kezelhető.

C2EF 2155C5	LD	HL,C555	Az ugrótábla eleje
C2F2 017004	LD	BC,0470	B=04, a ciklusfej;
			C=70, a lapozási
			alapkonzfigurációhoz kell,
C2F5 C5	PUSH	BC	ezeket a verembe tesszük
C2F6 5E	LD	E,(HL)	Az ugrócím-táblázatból
C2F7 23	INC	HL	betöltjük DE-be a következő
C2F8 56	LD	D,(HL)	szubrutin címét, majd az
C2F9 23	INC	HL	aktuális HL-t tárolva, a
C2FA E5	PUSH	HL	kiolvasott
C2FB EB	EX	DE,HL	címet HL-be tesszük át
C2FC CD21C3	CALL	C321	A C321-en JP HL van, tehát
			meghívjuk a kívánt szubrutint
C2FF E1	POP	HL	Annak lefutása után elővesszük
C300 C1	POP	BC	a tárolt értékeket,
C301 79	LD	A,C	visszaállítjuk a lapozási
C302 D302	OUT	(02),A	konfigurációt és mehet tovább
C304 10EF	DJNZ	C2F5	

Foglalkozunk röviden az egyes eszközök inicializáló rutinjaival.

Az első szubrutin címe: C9F2, a képmegjelenítőt hozza alaphelyzetbe. Ez a következőt jelenti:

- az L-MODE (0B4B), PAPER (0B4E) és V-FLAG (0B50) rendszerváltozóba 00-t töltve felülírást megvalósító vonalkeresztezési módot, a képernyőn fekete alapszint és teljes karakter felülírást állít be;
- az L-STYLE (0B4C) változóba 01-et töltve, folyamatosan húzott vonaltípust ír elő;
- az INK (0B4D) változó értékét 01-re állítja: az aktuális tintaszint az 1. palettaregiszter tartalmazza;
- 4 színű üzemmódot állít be, ezt feljegyzi a 0B73H rendszerváltozóba és a 06-os port kópiáján (0B13) is adminisztrálja;
- a színikapcsoló állapotát az 59H-s portról beolvassva, annak helyzetétől függően, a palettaregiszterekben 0...3 sorrendben, fekete-fehér állapot esetén a fekete-fehér-zöld-vörös, színes kapcsolóállapotban pedig a fekete-zöld-vörös-kék színek kódjait tölti;
- törli a képernyőt.

A második szubrutin a 0D5ECH címen kezdődik és a billentyűzetet inicializálja:

- beállítja a tartósan lenyomott billentyűk funkcióismétlésével kapcsolatos 0B65 és 0B67H rendszerváltozókat, valamint a LOCK-KEY (0B66) byte-ra 00-t írva, a billentyűzetet alapállapotba hozza;
- nullázza a billentyűzet állapotát jellemző PICTURE (0B51) és OLD-PIC (0B5B) byte-jait, valamint a teljes billentyű-munkaterületet.

A következő, hanggal kapcsolatos inicializáló rutin semmit nem csinál: a meghívott D960-as címen egyetlen RET áll. (A hangerőt már korábban beállítottuk 0 értékre.)

Az utolsóként meghívott D9E2-es kezdőcímű szubrutin a magnetofon inicializálására szolgál.

A magnetofont kezelő alacsony szintű rutinok az EXT ROM-ban találhatóak. A megfelelő memórialapozás után a rutin működése a következő:

- a CPU először törli a teljes munkaterületet (0BF0);
- a PROTECT (0B6D), EOF (0B6E) és MUDDLE (0B6F) rendszer-változókat törli, így "nem írásvédett file", "nincs file vége" és "az ellenőrző összeg kezdőértéke: 0" jelzéseket állít be;
- a REMRED (0B6C) változóba 80H-t írva mind írásra, mind olvasásra a bal oldali magnetofoncsatlakozóval összekötött készüléket jelöli ki;
- a 05-ös port b7-b6 bitjeit törölve mindkét magnetofon motorját kikapcsolja, s ezt a tényt PORT05 0 portkópián is adminisztrálja.

A négy beépített eszköz beállítása után az inicializáló rutin a TV-Computerhez oldalról dugaszolható programmodul csatlakozó állapotát vizsgálja meg.

C306 3E60	LD	A,60	Mivel a CART szegmens is legegyszerűbben a 3. lapon vizsgálható, ezért először: SYS-U1-U2-SYS, majd már a 0. lapon a következő sorra ugorva: SYS-U1-U2-CART lapozást állítunk be
C308 D302	OUT	(OZ),A	
C30A C30D03	JP	O30D	
C30D 3E20	LD	A,20	
C30F D302	OUT	(OZ),A	

A rendszer csak akkor foglalkozik közvetlenül, érdeemben a modullal, ha annak első byte-jain a MOPS azonosítószt elhelyezték. Ennek hiányában a modul csatlakozót üresnek tekinti. A CART memóriájával szemben is fontos további formai követelmény, hogy a MOPS szót követő byte-on már a rendszer vezérlését átvevő utasítások legyenek. A MOPS sód azonosítása után ui. a rendszer mindenképpen megkísérli a vezérlésátadást, s ha a modul erre nem lenne felkészítve, az a működés összeomlását okozhatja.

C311 2100C0	LD	HL,C000	A CART első byte-ja
C314 113403	LD	DE,0334	A 0. lapon vagyunk, tehát ez egy ROM-cím: itt tárolja a rendszer a MOPS szó karaktereit.
C317 0604	LD	B,04	4 betűt keresünk
C319 1A	LD	A,(DE)	Egy karakter kódja
C31A BE	CP	(HL)	Ez van CART-ban is?
C31B 2005	JR	NZ,C322	Ha nem: az egész tárgytalan, ugrás előre
C31D 13	INC	DE	Ha eddig egyezik, vesszük a következő byte-okat és nézzük tovább
C31E 23	INC	HL	
C31F 10F8	DJNZ	C319	
C321 E9	JP	HL	Ha mindegyik betű megvan: ugrás a HL=C004 címre, a többi már a modul feladata

Ha bármelyik karakternél az eltérés miatt az összehasonlítás félbeszakad, akkor a rendszer a ciklusból kilép. Ekkor a vezérlés a gépen belül marad. Mivel már minden lényeges dolog el van intézve, egyszerűen nincs semmi tennivaló. A magára maradott rendszer átadja a vezérlést a BASIC-nek: szenttelen várakozás kezdődik arra, hogy az ember, aki a számítógépet bekapcsolta, kezdeményezzen valami munkát.

A vezérlés átadása az embernek a következők szerint történik.

Mivel a programmodul memóriája elején nem szerepelt a MOPS szó, vissza kell állítani a memórialapozási alapkonfigurációt:

C322 3E60	LD	A,60	Először a SYS-t behozzuk a 3.
C324 D302	OUT	(02),A	lapra is
C326 C329C3	JP	C329	Ugrás a következő sorra,
			de most már a 3. lapon
C329 3E70	LD	A,70	Beállítjuk az
C32B D302	OUT	(02),A	alapkonfigurációt (U0-U1-U2-
C32D 320300	LD	(0003),A	SYS) és most már tároljuk is
C330 FB	EI		Ettől kezdve jöhetnek az
			engedélyezett megszakítások,
			s ezzel együtt belépés a
C331 C3EFD9	JP	D9EF	BASIC-be

Mint majd látni fogjuk, ott még lesz egy-két intézni való az első billentyű lenyomásáig.

Mi azonban -- amint ez a címekből látszik -- még hosszú ideig nem foglalkozhatunk ezzel a bizonyára izgalmasnak tűnő kérdéssel. Ebben a pillanatban pl. még a rendszerinicializálás nagyon fontos, a memóriák tesztelését végzőt szubrutinjával is adósok vagyunk.

Erről a szubrutinról korábban már mindent leírtunk, ezért most egyszerű dolgunk lesz.

Előtte azonban itt az összehasonlításhoz használt MOPS szó:

```
C334      4D 4F 50 53      "MOPS"
```

A memóriaszegmensek ellenőrzését a rendszer a következő szubrutinnal végzi. Két belépési pont van, mert ma rutin hívásakor a SYS a 0. lapon is lehet, s ilyenkor a szubrutinhívásban szereplő abszolút cím más.

HL-ben minden esetben a tesztelendő memóriaszegmens kezdőcíme van.

C338 E5	PUSH HL	Belépési pont a 0. lapon
		Ott természetesen a 0338 címen
		hívható
C339 CD4803	CALL 0348	Minden byte-ba 55H-t ír, majd
		1-gyel csökkenti és
		visszaolvassa. Ha a teljes 16
		kbyte-os szegmens jó: Z=1

C33C 1804	JR	C342	Ettől kezdve a rutin a 0. és a 3. lapon is ugyanaz
C33E E5	PUSH	HL	Belépési pont a 3. lapon
C33F CD48C3	CALL	C348	Az előbb, 0348-nál leírtuk. Ez ugyanaz a rutin, csak a 3. lapon a címe más
C342 D1	POP	DE	A szegmens tárolt kezdőcíme
C343 C0	RET	NZ	Ha a memória hibás: RET, ilyenkor HL az első hibás címre mutat
C344 EB	EX	DE,HL	Ha az első írás-olvasás sikeres volt, HL-be megint a szegmens elejét teszi
C345 3EAA	LD	A,AA	Másodszor ez lesz a tesztadat

Ismét egy sajátos eljárásnak lehetünk tanúi. Amikor a CPU ezt az utasítássorozatot hajtja végre, akkor a következő sornak semmi funkciója nincs:

C347 013E55	LD	BC,553E	Hatástalan!
-------------	----	---------	-------------

Fentebb azonban láttuk, hogy az első tesztelésnél a vezérlésátadási cím: C348, tehát az előbbi utasításnak egy belső byte-ja, operandusa.

Így viszont a CPU a megadott kódokat teljesen másképp értelmezi:

C348 3E55	LD	A,55	Az első tesztadat!
-----------	----	------	--------------------

Ez az eljárás korrekt értékadást valósít meg, ugyanakkor feleslegessé teszi a szubrutinba egy többsoros, feltételes vezérlésátadás beépítését.

A BC regiszterpár betöltése a program működésére semmiféle hatással nincs.

Ezután következik a tényleges memóriateszt: minden memóriarekesz írása, dekrementálása, s a jól működő rekeszek nullázása:

C34A E5	PUSH	HL	A szegmens eleje
C34B 5D	LD	E,L	
C34C 54	LD	D,H	HL, DE és BC megfelelő
C34D 13	INC	DE	paraméterezése után az A-beli
C34E 77	LD	(HL),A	adattal teleírja az egész
C34F 01FF3F	LD	BC,3FFF	szegmenst
C352 EDB0	LDIR		

C354	E1	POP	HL	Ismét a szegmens eleje
C355	0640	LD	B,40	BC=4000H
C357	3D	DEC	A	A beírt érték-1, és
C358	35	DEC	(HL)	memóriadekrementálás is
C359	EDA1	CPI		Még mindig egyezik?
C35B	2B	DEC	HL	(Mindenesetre igyekszik nul-
C35C	3600	LD	(HL),00	lázni)
C35E	C0	RET	NZ	Ha nem jó: visszatér, ha
C35F	23	INC	HL	egyezett: veszi a következő
				memóriarekeszt,
C360	E0	RET	PO	s ha már a szegmens végére
				ért: rendben!
C361	18F5	JR	C358	Addig azonban folytatja a
				vizsgálatot

Készen vagyunk !

Amint látja kedves Olvasó, valóban nem jutottunk el addig, hogy lenyomjuk az első gombot. De majdnem!

Ám ami késik, nem múlik: logikailag közel állunk már az első gombnyomáshoz. S egyszerűen azért nem vagyunk még közelebb, mert a ROM-ban előttünk áll a TV-Computer operációs rendszerének két monumentális bástyája: az egyik a funkcióhívások minden létező és leendő eszközre és részletre kiterjedő fő része, a másik pedig a megszakításokat kezelő alprogram, amely méreteit és bonyolultságát tekintve ugyan sokkal szerényebb, ám a számítógép "legjobban" futtatott programja, így az őt megillető tisztelettel kell majd foglalkoznunk vele.

3.3 A funkcióhívások kezelése

3.3.1 Bevezetés

A funkcióhívások kiszolgálását nagyszámú alacsony szintű szubrutin együtműködése jellemzi. Rövidesen ilyen rutinok tömegével ismerkedünk meg, s még később láthatjuk, hogy minden, ami a számítógépben történik, így vagy úgy kapcsolatban van ezekkel.

Először a funkcióhívások vezérlőprogramjával kell foglalkoznunk. Az előttünk álló feladat lényegileg kettős:

- Egyrészt utána kell járnunk annak, hogy a funkcióhívást megvalósító kétbyte-os parancsot a rendszer hogyan dolgozza fel. Hogyan jut el a funkciókódtól a feladatokat megvalósító alacsonyszintű rutinokhoz.

Ez önmagában egyszerű dolognak látszik, ám aránylag sokféle feltételt kell alapos vizsgálat alá venni, s ez kissé elbonyolítja a dolgot.

- Másrészt részletesen megismerkedünk a TV-Computer egyes eszközeit kiszolgáló, a konkrét feladatokat teljesen elemi műveletekre lebontó szubrutinokkal.

Egy-két esetben azonban még a szóban forgó eszköz működtetésével kapcsolatos elvi mechanizmusokat is tisztázni kell (pl. editor, magnetofonkezelő).

A másik, az operációs rendszer megszakításokat kiszolgáló alprogramja, helyileg és funkcionálisan is több ponton kapcsolódik a funkcióhívásokhoz.

Azt mondtam, hogy ez a számítógép "legjobban futtatott" programja: árnyaltabb megfogalmazásban persze arról van szó, hogy a dolog lényegéből adódóan ez a legtöbbször futtatott szubrutin. Mint tudjuk, általános esetben minden képmegjelenítési periódus végén, tehát körülbelül 20 ms-onként keletkezik egy megszakítást kérő jel, amelynek hatására a CPU az éppen kiszolgált program működésébe beszúrja az IT-alprogram egyszeri lefuttatását.

A megszakítást kiszolgáló program alapvető fontosságú feladatai közé tartozik a képmegjelenítés és a billentyűzet eszközeinek kiszolgálása. Ezek az ember és a gép közötti folyamatos kommunikáció nélkülözhetetlen eszközei.

Emellett azonban egyéb feladatokat is el kell látni, s egy sor másfajta feltételt is meg kell vizsgálni.

3.3.2 A funkcióhívások vezérlése

Maga az azonosítás rendkívül egyszerűen megtörténik azzal, hogy minden funkcióhívás az RST 30H utasítással kezdődik.

A 0030H címen elhelyezett JP 0B23 parancs a CPU-t a megfelelő címre viszi, ahol az ott elhelyezett utasítássorozat hatására az RST 30 utáni byte-ból kiolvasszódik az előírt funkciókód, amely a másodlagos A'-be kerül.

Az A' eredeti tartalmát és a 0003 címen elhelyezett P-SAVE értékét a rendszer a veremben tárolja és az U0-U1-U2-SYS memórialapozási alapkonfiguráció beállítása után adódik át a vezérlés a ROM-beli 0C363H címre.

Eddig jutottunk el az U0 megfelelő részeinek tanulmányozásánál és most innen megyünk tovább.

Belépéskor a funkcióhívások feldolgozásához szükséges paraméterek a DE és BC regiszterpárokban vannak és bizonyos járulékos információkat is ezekben adunk vissza. A legfontosabb azonban az előírt funkció ellátása után az A regiszterben visszaadott hibakód. Ez 00 kell legyen akkor, ha a funkcióhívás rendben zajlott le, míg bármilyen rendellenesség esetén a hiba természetére utaló, a hibát a lehető legpontosabban azonosító érték.

A funkciók azonosítása és a kért feladat végrehajtásának megszervezése közben figyelembe kell vennünk a következőket:

- a funkcióosztályok és az eszközök egymáshoz rendelése a megfelelő táblázat átírásával megváltoztatható;
- bizonyos funkciók csak meghatározott eszközökkel hajthatók végre;
- a csatolóártyák kezelése rendkívül dinamikusán, nagyon változatos formákban történhet;
- a 7. funkcióosztály (kernel) hívásai nem irányíthatók át semmilyen más eszközhöz.

A felsorolás még nem teljes, de így is jól látszik, hogy a megadott funkciókód tényleges célállomásának, valamint érvényességének vizsgálata a szervező szubrutin egyik legjelentősebb feladata.

A vezérlés

C363 E5	PUSH HL	Ijesztően nagyszabású
C364 DDE5	PUSH IX	előkészületek,
C366 FDE5	PUSH IY	
C368 D9	EXX	figyelemreméltó a HL
C369 C5	PUSH BC	regiszterpár mentése
C36A D5	PUSH DE	
C36B E5	PUSH HL	
C36C D9	EXX	
C36D 08	EX AF,AF	A-ban a funkciókód
C36E CD7EC3	CALL C37E	Ez a feldolgozást szervező
		fő szubrutin
C371 D9	EXX	
C372 E1	POP HL	
C373 D1	POP DE	A munkát befejezve
C374 C1	POP BC	visszatöltjük a tárolt
C375 D9	EXX	regisztereket
C376 FDE1	POP IY	
C378 DDE1	POP IX	
C37A E1	POP HL	
C37B C3370B	JP OB37	és kész

Következik a fő szervező rész.

C37E F5	PUSH AF	Először is átmenetileg
C37F C5	PUSH BC	tároljuk a funkciókódot és a
C380 D5	PUSH DE	belépő paramétereket
C381 4F	LD C,A	A funkciókódot és a rá jellemző
C382 B7	OR A	jelzőbiteket AF'-be menti
C383 08	EX AF,AF	
C384 21010B	LD HL,OB01	Ezután elsőként a billentyűzet
C387 7E	LD A,(HL)	funkcióosztályát nézzük.
C388 FE02	CP 02	Gyakori, hogy a karakterbe-
C38A 2001	JR NZ,C38D	olvasást az editorra bizzuk
C38C 35	DEC (HL)	(02. eszköz). Ha a táblázatban
		most is ez volt, vissza-
		állítjuk az alaphozzárendelést

C38D 11000B	LD DE,0B00	Ezután a funkciókód b7 bitje
C390 79	LD A,C	alapján DE-t a megfelelő
C391 07	RLCA	input- vagy output-táblázat
C392 3803	JR C,C397	elejére
C394 11080B	LD DE,0B08	állítjuk
C397 E6E0	AND E0	Csak a funkcióosztályt
C399 07	RLCA	kijelölő biteket hagyjuk meg,
C39A 07	RLCA	s 3-szor balra forgatva a
C39B 07	RLCA	tényleges értéket kapjuk,
C39C 47	LD B,A	ezután tehát B-ben a
C39D 79	LD A,C	funkcióosztály
C39E E60F	AND 0F	Utána a C-ben tárolt
C3A0 4F	LD C,A	funkciókód alsó bitjeit hagyva
		meg, előállítjuk C-ben a
		rutinsorszámot

Következik a megadott funkcióhívás végrehajthatóságának értékelése:

C3A1 78	LD A,B	A funkcióosztály
C3A2 FE07	CP 07	Kernel hívás?
C3A4 2832	JR Z,C3D8	Ha igen, ugrás előre

A kernel-hívásokat csak a kernel hajthatja végre, ezért ilyenkor át lehet ugrani a hozzárendelések megváltoztatásával foglalkozó vizsgálatokat.

C3A6 68	LD L,B	A funkcióosztály számát
C3A7 AF	XOR A	hozzáadva a megfelelő táblázat
C3A8 67	LD H,A	elejéhez, a kapott cím a hívott
C3A9 19	ADD HL,DE	osztály rekeszére mutat
C3AA 3D	DEC A	A funkció értelmezett?
C3AB BE	CP (HL)	Ha igen, a következő vizsgálat
C3AC 2005	JR NZ,C3B3	átléphető

Amennyiben a kijelölt funkcióosztályhoz a táblázatban FF van írva, ez azt jelenti, hogy a karakterműveletek erre az osztályra nem értelmezettek (pl. output a billentyűzethez), ilyenkor tehát a funkció száma csak 2-nél nagyobb lehet.

C3AE 79	LD A,C	A funkció száma
C3AF FE03	CP 03	2-nél nagyobb?
C3B1 301D	JR NC,C3D0	Ha igen, mehet előre, ha nem,
		akkor itt marad, de a következő
		szűrőn az FF miatt már hibát
		okoz

C3B3 7E	LD	A, (HL)	A hozzárendelt eszköz a táblázatból.
---------	----	---------	--------------------------------------

Ezzel kapcsolatban azt kell vizsgálni, hogy esetleg nem 6-nál is nagyobb-e, hiszen legfeljebb 6 féle eszköz lehet. Bonyolítja viszont a dolgot, hogy közvetlen csatolókartya hozzárendelés esetén a b7 bit 1, tehát ezt a vizsgálat idejére érvényteleníteni kell.

C3B4 CBBF	RES	7,A	Ettől az FF jelzés 7F-re változik, de még így is bőven hibát okoz!
C3B6 FE07	CP	07	Ha az eszköz száma 0...6 közé esik: mehet
C3B8 3806	JR	C,C3C0	Egyébként beállítja a "hozzárendelési hiba" jelzést, visszatölti a tárolt regisztereket és ugrás a hibakezelő ágra.
C3BA 3EFE	LD	A,FE	
C3BC D1	POP	DE	
C3BD C1	POP	BC	
C3BE 183C	JR	C3FC	

Ami még fent maradt a rostán, tovább vizsgáljuk:

C3C0 CB7E	BIT	7, (HL)	Az eszköz csatolókartya?
C3C2 2806	JR	Z,C3CA	Ha nem, átugorja a kártyakezelő sorokat, ha igen, ugrás az EXT ROM-ba, a 0F166H címre
C3C4 2166F1	LD	HL,F166	
C3C7 C3F0FF	JP	FFFO	

A közvetlen csatolókartya hozzárendelések esetén a funkcióhívások kiszolgálását az EXT ROM végzi. Munkáját nagyvonalakban e rész után ismertetjük.

C3CA 79	LD	A,C	A funkció sorszám
C3CB FE03	CP	03	A karakteres- és blokk- I/O funkció a kívánt,
C3CD 3001	JR	NC,C3D0	a többi (03...0FH) az osztály saját eszközén hajtódik végre
C3CF 46	LD	B, (HL)	
C3D0 78	LD	A,B	A-ban a véglegesen kijelölt eszköz száma
C3D1 FE06	CP	06	Csatolókartya?
C3D3 216CF1	LD	HL,F16C	
C3D6 28EF	JR	Z,C3C7	Ha igen, ugrás az EXT-be

Tudjuk, hogy egy adott osztály funkcióhívásai kétféle módon rendelhetők csatolókartya-hoz: a táblázatba közvetlenül a csatoló számát és b7=1-et írunk, vagy pedig a táblázatba csak a kártyahozzárendelés tényét jegyezzük be 06-os eszközszámmal, s ekkor a táblázat utolsó elemébe még be kell írunk a csatlakozóhely számát.

A funkcióhívás lebonyolítását az EXT ROM végzi, az első esetben a 0F166H, a második esetben a 0F16CH belépési címmel, a következőképpen:

- ellenőrzi a megadott csatlakozószámot, s ha az 3-nál nagyobb, beállítja a 0B1C-n nyilvántartott bővítőkártya-alaphozzárendelést, majd a hozzárendelési hibát jelezve visszatér a SYS ROM 0C3FCH címére;
- belapozza a kért csatolóhely memóriáját, s rááll az U0-ban a hozzárendelt területre (0040H, 0070H, 00A0H vagy 00D0H);
- ha a csatoló üres, akkor a fent leírt módon és hibával visszatér:
- RS232 soros vonal esetén az EXT ROM-beli ugrótáblát, egyéb külső eszközök esetén pedig azok memóriájának 0C00DH címén kezdődő ugrótábláját nézi. Ez a címszerkezet a külső eszközök memóriaszervezésére kötelező előírás;
- az ugrótáblák alapján ellenőrzi, hogy a hívott funkció száma az adott eszköz esetén létezik-e. Ha nem, akkor a bővítőkártya alaphozzárendelés visszaállítása nélkül, "nemlétező funkciókód" hibával visszatér (SYS, C3FC);
- az IX regisztert ráállítja a bővítő számára az U0-ban fenntartott pufferre (0048H, 0078H, 00A8H vagy 00D8H), s az ugrótábla szerinti címre ugorva végrehajtja az ott elhelyezett szubrutint;
- végül visszatér a SYS ROM C3FC címére.

Amennyiben a funkcióhívás nem egy bővítőkártyát jelöl ki, akkor a végrehajtás itt folytatódik:

C3D8 115DC5	LD DE,C55D	Ez a funkcióhívásokhoz tartozó ugrótábla eleje
C3DB 68	LD L,B	
C3DC 2600	LD H,00	HL = a kijelölt eszköz sorszáma
C3DE 29	ADD HL,HL	*2, a kétbyte-os címek miatt
C3DF 19	ADD HL,DE	Most HL a kijelölt eszközhöz tartozó címre mutat. Minden eszköznek saját ugrótáblája van (az egyes rutinokhoz), most ezt tölti DE-be, onnan pedig HL-be
C3E0 5E	LD E,(HL)	
C3E1 23	INC HL	
C3E2 56	LD D,(HL)	
C3E3 EB	EX DE,HL	
C3E4 7E	LD A,(HL)	Az ugrótáblák első byte-ján a hívható rutinok száma áll
C3E5 3D	DEC A	
C3E6 B9	CP C	Ha ennél nagyobbat kértek:
C3E7 3EFF	LD A,FF	"nemlétező funkciókód"
C3E9 38D1	JR C,C3BC	hibával visszatér

Az egyes eszközök ugrótáblájában a hívható rutinok száma után azok kezdőcímei állnak.

C3EB 23	INC HL	Rááll a 0, szubrutin
C3EC EB	EX DE,HL	kezdőcímére,
C3ED 69	LD L,C	majd a kért rutinsorszám
C3EE 2600	LD H,00	szerinti eltolással az adott
C3F0 29	ADD HL,HL	funkcióhoz tartozó címet
C3F1 19	ADD HL,DE	tartalmazó byte-ra
C3F2 5E	LD E,(HL)	
C3F3 23	INC HL	Onnan a kért szubrutin címét a
C3F4 56	LD D,(HL)	DE regiszterpárba, majd a HL-
C3F5 EB	EX DE,HL	be tölti.
C3F6 08	EX AF,AF	Átteszi A-ba a funkciókódot,
		F-be az I/O irány eldöntéséhez
		szükséges előjel-bitet (S),
C3F7 D1	POP DE	végül visszatölti a verembe
C3F8 C1	POP BC	mentett bemenő paramétereket
		és
C3F9 CD21C3	CALL C321	ugrás a HL címre

Ezzel a dolog már el is van intézve. A 0C321H cím -- már láttuk -- egyetlen JP HL-t tartalmaz. Továbbiakban a szubrutin lefutása már kizárólag az aktivizált eszközön, ill. a DE-ben és BC-ben átadott paraméterek jóságán múlik. A funkcióhívások szervezését végző rutinra már csak annyi maradt, hogy a meghívott rutinok lefutása után, amikor a vezérlés visszaadódik a 0C3FCH címre, rendezze a hibakód szerinti visszatérést.

Nézzük meg, ez hogyan történik.

C3FC B7	OR A	Az A-ban 00 van?
C3FD E1	POP HL	A funkciókódot kidobjuk és
C3FE C8	RET Z	amennyiben nem volt hiba,
		kész!
C3FF F5	PUSH AF	Hiba esetén a kódot eltesszük
C400 3A200B	LD A,(0B20)	Tisztázzuk, hogy IT-
C403 3C	INC A	kiszolgálás van-e?
C404 280A	JR Z,C410	IT esetén átlépjük az
		alaphozzárendeléseket
		visszaállító sorokat

A következőkben, mivel a végrehajtás közben hiba lépett fel, vissza kell állítani az alaphozzárendeléseket. Ezt ismét az EXT ROM-beli program végzi, a 0F1E6H címtől.

Nem változtatja meg azonban a közvetlen csatolókartya-hozzárendeléseket (amikor tehát a funkcióosztályhoz tartozó táblázatelembe a csatoló számát és b7=1-et írunk).

Ha valamelyik funkcióosztályhoz a 06-os eszközszámmal rendeltek csatolókarttyát, akkor a táblázat megfelelő utolsó elemébe szintén az alaphozzárendelés kerül.

C406 C5	PUSH BC	Tárolja a paraméterek
C407 D5	PUSH DE	átadására szolgáló
C408 7C	LD A,H	regisztereket, s a
		funkciókódot A-ba tölti
C409 21E6F1	LD HL,F1E6	Az EXT ROM-ban elvégzi az
C40C 18B9	JR C3C7	alaphozzárendelések
		visszaállítását a megbeszél
		módon, majd itt folytatja:
C40E D1	POP DE	visszatölti a regisztereket,
C40F C1	POP BC	majd
C410 F1	POP AF	a tárolt hibakódot és ezzel
C411 C9	RET	végképp kész!

3.3.3 Az ugrótáblák

Ennyi volt tehát a funkcióhívások felismerését, a hívások korrektségének ellenőrzését végző vezérlő rutin munkája. Gondolom, jól látszik, hogy a munka fő részét a nagyfokúan rugalmas működtetés feltételeinek biztosításával járó, nagyszámú feltételrendszer alapos vizsgálata jelentette.

Ismét felhívom a kedves Olvasó figyelmét az előbbi szubrutinban több helyen is előforduló ugrótábla-kezelés technikájára, vagy méginkább az ebben rejlő lehetőségek értékelésére, s hasonló feladatok megoldása esetén módszer önálló alkalmazására.

Segítségével, láthatóan, egymástól teljesen független szubrutinok kezelhetők egyetlen, nagyon egyszerű, áttekinthető és logikus sémában, nem is beszélve a memória helytakarékos kihasználásáról.

Röviden szólnunk kell még -- a pontos megértés kedvéért -- ezeknek az ugrótábláknak a szerkezetéről.

Először is minden eszközhöz tartozik egy-egy ilyen tábla. Ezek első byte-jain az eszköz kiszolgálásához tartozó rutinok száma található, az azt követő byte-okon pedig a megfelelő szubrutinok címeinek egyszerű felsorolása. A kért rutin sorszámának ismeretében ezek a címek igen egyszerűen kiolvashatók.

Az egyes eszközökhöz tartozó ugrótáblák, ezek sorszáma szerinti sorrendben:

00	C974	video (képmegjelenítés)
01	D5E3	billentyűzetkezelő
02	CF98	editor (képernyőszerkesztő)
03	D92A	hangképzésvezérlő
04	D8FF	nyomtatókezelő
05	D9BE	magnetofonkezelő
06	0000	csatolókartya kiszolgáló
07	C4AC	kernel (belső rendszerszervezési funkciók)

A csatolókartákat természetesen másképpen kezeli a rendszer, de a helyét -- a helyes számolás miatt -- itt is ki kellett hagyni.

Az összes eszköz ugrótábláját pedig a rendszer egy másik táblázatban tartja nyilván, ennek kezdőcíme: 0C55DH. A hívott funkciót kiszolgáló eszköz tisztázása után innen olvasható ki a hozzá tartozó ugrótábla címe.

Az egész mechanizmus alapjában véve rendkívül egyszerű, hatékony, gyors és gazdaságos.

Az egyes eszközökhöz tartozó kiszolgáló alacsony szintű rutinok a SYS ROM és az EXT ROM hatalmas területét foglalják el. A megfelelő címekhez érve valamennyit meg fogjuk ismerni.

Előbb azonban a másik -- alapvető fontosságú - ROM területtel, a megszakításokat kiszolgáló programmal kell foglalkoznunk.

3.4 A megszakításkezelő főprogram

A mikroprocesszor INT (interrupts) vezetékére érkező vezérlőjel az újabb megszakításkérések tiltását váltja ki és a CPU a memória 0038H címére ugrik.

Láttuk, hogy ott az AF regiszterpár tárolása, majd az U0-U1-U2-SYS memórialapozási alapkonfiguráció beállítása történik meg, s ezután ugrás a ROM 0C412H címére.

Az IT rutin befejezése ismét az U0 RAM-ban, a 0B41H címtől található, már ugyancsak ismert.

Most megismerjük, mi van a kettő között.

C412 3A0300	LD	A,(0003)	Az IT rutinnak a CPU-ban lévő
C415 F5	PUSH	AF	minden információt tárolnia
C416 E5	PUSH	HL	kell, s a végén majd ezeket
C417 D5	PUSH	DE	vissza kell állítania. Ezért
C418 C5	PUSH	BC	az itteni sorokhoz nem kell
C419 DDE5	PUSH	IX	semmiféle kommentár
C41B FDE5	PUSH	IY	
C41D 08	EX	AF,AF	
C41E F5	PUSH	AF	
C41F D9	EXX		
C420 E5	PUSH	HL	
C421 D5	PUSH	DE	
C422 C5	PUSH	BC	
C423 CD37C4	CALL	C437	Ez a szubrutin lesz természetesen a lényeg!

A végrehajtás után jöhetnek a helyreállítások:

C426 C1	POP	BC	A veremből a CPU minden
C427 D1	POP	DE	regiszterét visszaállítjuk a
C428 E1	POP	HL	megszakítás előtti értékre.
C429 D9	EXX		Ezzel elérjük, hogy a
C42A F1	POP	AF	megszakított program nem is
C42B 08	EX	AF,AF	tud arról, hogy közben a CPU mi
C42C FDE1	POP	IY	minden mással foglalkozott
C42E DDE1	POP	IX	
C430 C1	POP	BC	
C431 D1	POP	DE	
C432 E1	POP	HL	
C433 F1	POP	AF	
C434 C3410B	JP	0B41	

A 0B41H RAM-címen kezdődő utasítássorozat visszaállítja a felhasználói memórialapozást és engedélyezi az újabb megszakításokat. E programrész átírásával az IT-rutin munkáját tetszőlegesen bővíthetjük!

Kövessük nyomon a megszakítások kiszolgálását!

C437 3EFF	LD	A,FF	Ez egy fontos jelzés arról,
C439 32200B	LD	(0B20),A	hogyan most az IT rutin fut
C43C 2A1D0B	LD	HL,(0B1D)	
C43F 23	INC	HL	A belső óra léptetése
C440 221D0B	LD	(0B1D),HL	
C443 3A4F0B	LD	A,(0B4F)	majd a BORDER beállított
C446 D300	OUT	(00),A	színének érvényesítése
			következik (tehát minden
			megszakításnál)

Az előbbi adminisztrációk elvégzése után indulhat a munka lényegi része.

C448 DB59	IN	A,(59)	Az 59H-s port alsó 5 bitje 0
			értékkel jelzi az IT kérés
			forrását
C44A 4F	LD	C,A	
C44B CB67	BIT	4,A	A 4. bit a kurzor vagy a hang
			IT kérését jelzi.
C44D CBE1	SET	4,C	Miután megnéztük, vissza is
			állítjuk 1-re,
C44F D307	OUT	(07),A	majd 07-es port írásával
			nyugtázzuk.
C451 CC7DC4	CALL	Z,C47D	Ha valóban a kurzor vagy a
			hang kérte az IT-t, akkor
			először ezt szolgáljuk ki

Ezt a szubrutint majd végignézzük, most azonban vegyük sorra a többi eszköz IT-kéréseinek feldolgozását:

C454 79	LD	A,C	A b7...b4 biteket 1-be
C455 F6F0	OR	F0	állítjuk: így könnyebb lesz az
			alsó 4 bit vizsgálata

A b3...b0 bitek 0 értéke a csatoló kártyákról jövő megszakításkéréseket jelzi.

C457 3C	INC	A	Először is megnézzük, hogy
			van-e egyáltalán ilyen kérés.
C458 281E	JR	Z,C478	Ha nincs, ugrás a befejezéshez

Ha azonban valamelyik kártya IT-kérést jelzett, akkor tovább kell nézni.

C45A 3A1F0B	LD	A, (OB1F)	IRQ-STAT-ból kiolvashatjuk, hogy a kért IT-kiszolgálás megengedett-e.
C45D 0F	RRCA		Az alsó 2 bit a kurzorra és a hangra vonatkozik, számunkra érdektelen
C45E 0F	RRCA		
C45F 61	LD	H,C	Az eredeti kéréseket tovább mentjük C-ből, mert a 4 kártya vizsgálatára szervezett ciklusban C-t a portok címzésére használjuk.
C460 69	LD	L,C	
C461 015804	LD	BC,0458	
C464 57	LD	D,A	A két RRCA után a b3...b0 a 4 csatoló IT-kéréseinek engedélyezését jelzi
C465 AF	XOR	A	Ez a ciklus 4-szer fut le, s figyelemreméltóan szellemes tömörséggel oldja meg a 4 kártya IT-engedélyezésével kapcsolatos összes vizsgálatot és adminisztrációt
C466 C80A	RRC	D	
C468 1F	RRA		
C469 C80C	RRC	H	
C46B 3802	JR	C,C46F	
C46D ED79	OUT	(C),A	
C46F 0C	INC	C	
C470 10F3	DJNZ	C465	

Elemezzük komolyabban az előző sorokat!

A D regiszter az IT-k engedélyezését tartalmazza, így:

b7	b6	b5	b4	b3	b2	b1	b0
HANG	KURZOR	-	-	3.cstl	2.cstl	1.cstl	0.cstl

Amelyik engedélyezett, a hozzá tartozó bit értéke: 1.

A H regiszter alsó bitjei pedig a csatoló kártyák tényleges IT kéréseit tartalmazzák, de itt a bit 0 értéke az aktív.

A csatolók IT-jének engedélyezéséhez a 0...3 sorrendben az 58H...5BH-s portok b7 bitjébe kell 1-et írni. Ezeket a portokat címzi meg sorban a C regiszter.

Nézzük ezek után az előbbi sorok működését!

A D-t jobbra léptetve, a carry jelzőbit közvetítésével, az egyébként üres A regiszter 7. bitjébe lép a ciklusban soronkövetkező kártyára beírt IT-engedélyező bit. Így az A értéke éppen 80H lesz akkor, ha a kártya IT-kérésének kiszolgálása megengedett, míg 00, ha nem. Az A így előállított értéke tehát közvetlenül alkalmas a megfelelő C-beli port írására.

C értéke pedig szinkronban van az éppen vizsgált csatlakozóval.

Az engedélyező bit kiküldésére azonban csak akkor kerülhet sor, ha a kártya valóban kért IT-t. Ezt oldja meg a H regiszter bitjeinek jobbra léptetése: a carry-be léptetett bit 0 értéke jelzi az IT-kérést, s ekkor megtörténik az aktuális port írása, egyébként pedig ezt az utasítást átugorjuk.

Összesen 6 sorban megtettünk minden vizsgálatot, s minden szükségesintézkedést!

A portok ilyen előkészítése után a vezérlést a csatolókértékártyák IT-rutinjainak adjuk át.

C472 4D	LD	C,L	Visszatöltjük az eredeti IT kéréseket
C473 2178C4	LD	HL,C478	Ez lesz a visszatérési cím (alább megnézzük), s ugrás EXT ROM-ba
C476 182B	JR	C4A3	

A C4A3 címen egyszerű intézkedés történik a megfelelő vezérlésátadásra. Az összes IT kiszolgálása után a munka a következőképpen fejeződik be:

C478 AF	XOR	A	Megszünteti az "IT
C479 32200B	LD	(0B20),A	kiszolgálása folyik" jelzést és kész.
C47C C9	RET		

A C47D-n kezdődő szubrutin a kurzor vagy a hang által kért megszakítások kiszolgálását végzi.

C47D C5	PUSH	BC	C az eredeti IT kéréseket tartalmazza, ezt tárolni kell
C47E 3A100B	LD	A,(0B10)	INT-DES, benne a kiszorgálandó eszközök

Ez a rendszerváltozó b7...b0 sorrendben a következő eszközök kiszorgálását jelzi: 3.cstl. -- 2.cstl. -- 1.cstl. -- 0.cstl. -hang -- editor -- billentyűzet -- video. A bitek 0 értéke aktív. Először a négy utóbbi, belső eszközzel foglalkozunk.

C481 4F	LD	C,A	C-ben az eszközök
C482 0604	LD	B,04	Ismét 4-es ciklus
C484 CB19	RR	C	A carry-ben 0 van, ha az eszközt ki kell szorgálni

A bitek jobbra léptetése miatt video--billentyűzet--editor--hang a kiszorgálás sorrendje.

C486 C5	PUSH BC	A ciklusszámlálót és a C bitjeit megőrizzük, s ha az eszközt nem kell kiszolgálni, a következő sorokat átlépjük
C487 3810	JR C,C499	
C489 3E04	LD A,04	
C48B 90	SUB B	A kivonás után B-ben az aktuális eszköz sorszáma
C48C 47	LD B,A	
C48D 0E00	LD C,00	

Ily módom a B és C regiszterek állapota éppen megfelel az egyes eszközök funkcióhívásainál leírtaknak, a vezérlést valóban át is adjuk, mégpedig jól láthatóan a C=00 funkciószerű rutinoknak.

C48F 2199C4	LD HL,C499	Ez lesz a visszatérési cím, többszöri lerakása a korrekt veremkezeléshez szükséges.
C492 E5	PUSH HL	
C493 E5	PUSH HL	
C494 E5	PUSH HL	
C495 E5	PUSH HL	Utána közvetlen ugrás a funkcióhívásokat kezelő szubrutin ugrótábla beolvasó soraira
C496 C3D8C3	JP C3D8	

A 0-s funkció végrehajtása az egyes eszközök esetében a következő műveleteket jelenti:

-- video:	semmi
-- billentyűzet:	teljes letapogatás, a lenyomott billentyű kódjának előállítása 0BE9H-ra, a tartósan lenyomott gombok aktuális ismétlési paramétereinek figyelembevételével, a CTRL+P funkció ellátása;
-- editor:	a kurzor villogtatása;
-- hang:	megszólaltatott hang esetén a megadott időtartam nyomonkövetése, majd letiltása és az ezzel kapcsolatos adminisztrációk.

C499 3E70	LD A,70	A funkció ellátása után visszaállítja az eredeti lapozási konfigurációt, behozza a tárolt ciklusparamétereket és folytatja tovább
C49B D302	OUT (02),A	
C49D C1	POP BC	
C49E 10E4	DJNZ C484	

Amikor ez kész, a csatolóártyák kiszolgálása következik.

```

C4A0 21AAC4      LD   HL,C4AA
C4A3 E5          PUSH HL
C4A4 2127F2      LD   HL,F227
C4A7 C3FOFF      JF   FFF0

```

Ez lesz a visszatérési cím. A csatolókérták kiszolgálását röviden áttekintjük.

A munka az EXT ROM-ban folyik, az F227H belépési címmel. A C regiszter alsó 4 bitje mindkét esetben az egyes kártyák aktivitását jelzi.

- Amelyik bit nem aktív, azzal a csatolóval nem foglalkozik.
- Belapozza az aktív kártya memóriáját a 3. lapra, a C000H területre.
- Szigorú követelmény, hogy a memóriák a 0C00E-C00FH helyen az IT rutinjuk címét tartalmazzák. A program meghívja ezt a rutint.
- Helyreállítja a korábbi memórialapozást.

A csatolókérták kiszolgálása után:

```

C4AA C1          POP  BC      Visszatölti az eredeti IT
C4AB C9          RET                    kérések bitjeit és visszatér a
                                         C454H címre

```

Ez volt tehát a TV-Computer operációs rendszerének IT-kezelő alprogramja.

Összefoglalva elmondható, hogy mind a csatolókértákkal a géphez kapcsolt, mind pedig a TV-Computer saját belső fizikai és logikai eszközeinek IT-szintű kiszolgálását az operációs rendszer kurzor vagy hang által vezérelt programmechanizmusa kimerítően megoldja.

Emellett lehetővé teszi a csatolókérták eszközeinek a saját megszakítási szándékaik érvényesítését is, ha azok áramkörileg erre fel vannak készítve.

A TV-Computer megszakításkezelő alprogramja a különböző rendszerváltozók (INT-DES, IRQ-STAT és INT-FLAG), és I/O portok (03H, 05H, 07H, 58H, 59H, 5AH, 5BH, 70H és 71H) megfelelő bitjeinek segítségével az IT szintjén megoldandó feladatok terén a fizikai adottságok, a felmerülő igények és a felhasználói befolyásolhatóság magas színvonalú összhangját valósítja meg.

Mindezt pedig ügyes programozói fogásokkal, az alkalmazott mikroprocesszor utasításkészletének sajátosságaira, a portok céltudatos bitkiosztására építve éri el. A hardver és a szoftver itt is megnyilvánuló tökéletes összhangja követésre és elismerésre méltó teljesítmény.

3.5 A funkcióívások végrehajtása

Elérkeztünk a ROM-nak ahhoz a területéhez, amely az egyes eszközökhöz intézett funkcióhívások különböző szintű rutinjait tartalmazza. Az itt elhelyezkedő rövidebb-hosszabb, egyszerűbb vagy logikailag nehezebben követhető programok legtöbbször közvetlen kapcsolatban vannak a hozzájuk tartozó eszközzel. Vagy azok fizikai működésére hatnak a portok közvetítésével, vagy pedig -- logikai eszközök esetén -- éppen e rutinok működése jelenti magát az eszközt.

A következő oldalakon -- a ROM-beli elhelyezkedésüket követve -- sorra vesszük az egyes eszközöket. Azokban az esetekben, amikor a teljes kezelőprogramot az EXT ROM tartalmazza, egyelőre természetesen megelégszünk azok rövid, szóbeli jellemzésével. Azonban a megfelelő EXT ROM-rész tárgyalásakor részletesen elemezzük az ott megírt rutinokat, utalva az itt leírtakra is.

Az Olvasó persze megteheti, hogy az ilyen részeket jellemző leírásokkal párhuzamosan fellapozza a hivatkozott rutinok listáit is és azokat is alaposan elemzi. A leírtak megértéséhez, vagy inkább annak tudomásul vételéhez azonban erre általában nincs szükség.

És még valami. A könyv terjedelmére megszabott -- érthető -- korlátozások miatt az egyes rutinok kifejtése nem lesz egyformán részletes. Rossz esetben előfordulhat, hogy pont az Olvasót leginkább érdeklő részletek lesznek hiányosak. Ilyen esetben a megadott támpontok alapján kérem, hogy az Olvasó tegye próbára erejét, az utalások ott is segítenek, s bízom benne, hogy rövid idő alatt maga is választ tud adni a működés részleteivel kapcsolatos kérdésekre. Bízom abban, hogy erre valóban ritkán kerül sor.

3.5.1 A kernel funkciók

Az elnevezés kicsit megtévesztő, mert nem a kernel teljes tartalmának megfelelő funkciók hívásáról van szó (kernel = belső mag, valaminek a lényegi része, a számítástechnikában azoknak a programoknak az általános elnevezése, amelyek a gép működésének logikai felügyeletét, annak szervezését látják el, biztosítják az erőforrásokhoz való konfliktusmentes hozzáférést stb.

Tehát lényegében a számítógép programszintű használatának elemi feltételeit teremtik meg).

A hívható kernelfunkciók ezzel szemben néhány egyszerű táblázat-, ill. rendszerváltozó értékátírást helyeznek a funkcióhívások egységes formai köntöse alá.

A kernelfunkciók ugrótáblája a 0C4ACH címen helyezkedik el. A funkcióhívások vezérlőrutinjánál láttuk, hogy a 07-es osztályhoz intézett hívások esetén a vezérlő közvetlenül az ugrótáblákhoz fordul: a kernelhívások nem rendelhetők hozzá semmilyen más eszközhöz, az input-output hozzárendelési táblázat 7. osztályhoz tartozó rekeszeiben nem is ezzel kapcsolatos adatok vannak (hanem a csatolókartákra vonatkozó információk).

A vezérlő a kernel-osztály sorszámából a megfelelő fő táblázat (C55DH...) alapján eljut a kernel ugrótáblájához, amely -- a már ismert struktúra szerint -- a következő:

ROM-cím	Tartalom	Jelentés
C4AC	05	A hívható kernelfunkciók száma
C4AD	C509	0. funkcióhoz tartozó rutin címe
C4AF	C4B7	1. funkció (HI-MEM-SET)
C4B1	C4D0	2. funkció (SLOT-ASN)
C4B3	C4E2	3. funkció (IO-ASN)
C4B5	C50E	4. funkció (SLOT-NUM)

A 0. funkciók -- mint tudjuk -- az operációs rendszer IT-kezelő programja számára vannak fenntartva (bizonyos feltételek mellett persze mi is hívhatjuk).

A kernel esetében ehhez semmiféle tényleges feladat nem tartozik, de az egységes táblázatkezelési technika megkívánja minden eszköznél a 0. funkció megjelölését. Ez az ilyen esetekben általában egy RET utasítás címe.

Amint az ugrótáblából ez kiderül, a ROM programja az 1. kernelfunkció rutinjával folytatódik.

A rutinok belépési pontjánál, most és a továbbiakban mindig, megadjuk az eszköz nevét, mellette a rutin sorszámát, alatta a hívásnál használható funkciókódot, továbbá a funkció nevének -- többnyire angol nyelvű - rövidítését és a funkció leírását. Ugyancsak emlékeztetésül a beállítandó, valamint a kilépő, jellemző regiszterértékeket is megadjuk, kiemelten kezelve persze az A regiszterbeli hibakódot.

KERNEL 01		HI-MEM-SET (high memory-address set)
71H, F1H		A legmagasabb használható memóriacím beállításával egy kért memóriaterület biztosítása
	in:	DE=a kért memória mérete
	out:	DE=a biztosított memóriaterület kezdőcíme
		A=00: rendben
		A=FB: a kért memóriaméret túl nagy
C4B7 2A190B	LD HL,(0B19)	HI-MEM-ből az aktuális legnagyobb RAM-cím
C4BA 3EFB	LD A,FB	Előkészíti a hibakódot
C4BC B7	OR A	Carry=0, a kivonáshoz
C4BD ED52	SBC HL,DE	Ha a kért memória mérete az összesnél is nagyobb:
C4BF D8	RET C	nyilvánvaló hiba!
C4C0 E5	PUSH HL	Nézi a maradékot
C4C1 11AC1E	LD DE,1EAC	Ha a kért terület
C4C4 ED52	SBC HL,DE	lefoglalása után a maradék
C4C6 E1	POP HL	01EACH alá csökkenne (2
C4C7 D8	RET C	kbyte a verem után): nem
C4C8 54	LD D,H	lehet!
C4C9 5D	LD E,L	DE=a lefoglalt memóriaterület
C4CA 13	INC DE	eleje
C4CB 22190B	LD (0B19),HL	Az új HI-MEM
C4CE AF	XOR A	"Hibátlan" jelzés és
C4CF C9	RET	kész

KERNEL 02		SLOT-ASN (slot assign)
72H, F2H		Azonosítóval megadott csatolókartát rendel a 00. funkcióosztályhoz
	in:	B=FF: inputra kijelölés
		bármilyen más érték: outputra kijelölés
		C=egységyszám (több, azonos kártya esetén 0...3)
		DE=az azonosítót tartalmazó memória kezdőcíme (azonosító: névhossz + név)
	out:	HL=az input- vagy output-táblázat 7. elemének címe
		A=00 nincs hiba, ekkor
		IX=a kártyához rendelt I/O puffercím
		A=FD nincs ilyen kártya

C4D0 21070B	LD HL,0B07	Input-tábla 7. rekesze
C4D3 04	INC B	Input kell?
C4D4 2B03	JR Z,C4D9	Ha igen, a következő sort átlépi
C4D6 210F0B	LD HL,0B0F	Output-táblázat vége
C4D9 E5	PUSH HL	

C4DA CDOEC5	CALL C50E	Meghívja a 4. funkciót: ez C-ben visszaadja a cstl. számát és IX-t a pufferre állítja
C4DD E1	POP HL	
C4DE B7	OR A	Hiba van?
C4DF C0	RET NZ	Ha igen: visszatér,
C4E0 71	LD (HL),C	ha hibátlan: megvalósítja a hozzárendelést és
C4E1 C9	RET	kész

KERNEL 03
73H, F3H

IO-ASN (input-output assign)
A funkcióosztály -- eszköz hozzárendelés megváltoztatása
in: B=funkcióosztály
C=eszköz száma
D=FF, ha az irány input
bármilyen más érték, ha output
out: A=00: hibátlan, ekkor
HL=az I/O táblázat megfelelő címe
A=FE: hozzárendelési hiba

C4E2 21000B	LD HL,0B00	A D-ben megadott művelet szerint HL az in- vagy out-tábla elejére áll.
C4E5 14	INC D	
C4E6 2803	JR Z,C4EB	
C4E8 21080B	LD HL,0B08	
C4EB E5	PUSH HL	Az I/O táblázat eleje
C4EC 3E06	LD A,06	
C4EE B8	CP B	Az osztály sorszáma nagyobb 6-nál?
C4EF 3819	JR C,C50A	Ha igen: nem lehet!
C4F1 58	LD E,B	
C4F2 1600	LD D,00	
C4F4 19	ADD HL,DE	HL az osztály rekeszére mutat
C4F5 5E	LD E,(HL)	Az osztályra a funkció értelmezett? (FF: nem)
C4F6 1C	INC E	
C4F7 2811	JR Z,C50A	Ha nem, hiba!
C4F9 D1	POP DE	
C4FA E5	PUSH HL	Tárolja az osztály címét
C4FB EB	EX DE,HL	HL=az I/O táblázat eleje
C4FC B9	CP C	Az eszköz száma 6-nál nagyobb?
C4FD 380B	JR C,C50A	Ha igen: nem lehet!
C4FF 59	LD E,C	
C500 AF	XOR A	
C501 57	LD D,A	
C502 19	ADD HL,DE	HL a kért eszköz rekeszére mutat

C503	5E	LD	E, (HL)	A funkció az eszközre
C504	1C	INC	E	értelmezett? (FF: nem)
C505	2803	JR	Z, C50A	Ha nem: hiba!
C507	E1	POP	HL	Ha eddig minden jó, akkor
C508	71	LD	(HL), C	elvégzi a hozzárendelést és
C509	C9	RET		A=00-val visszatér
C50A	E1	POP	HL	Hiba esetén beállítja a
C50B	3EFE	LD	A, FE	hibakódot
C50D	C9	RET		és így is kész!

KERNEL 04
74H, F4H

SLOT-NUM (slot number)
Az azonosítóval megadott kártya csatlakozó számának meghatározása

in: DE=az azonosítót tartalmazó memóriaterület kezdőcíme (azonosító: névhossz+név)
C=egységszám (azonos kártyákra 00...03)

out: A=00: hibátlan, ekkor
C=a kártya pozíciója (00...03)
IX=a kártya I/O pufferének címe
A=FD: nincs ilyen kártya

C50E	D5	PUSH	DE	A 0. csatlakozó I/O puffere
C50F	DD214000	LD	IX, 0040	
C513	0604	LD	B, 04	4 kártyahelyet nézünk
C515	DDE5	PUSH	IX	
C517	E1	POP	HL	HL-az aktuális puffercím
C518	C5	PUSH	BC	
C519	1A	LD	A, (DE)	A megadott névhossz
C51A	3C	INC	A	
C51B	47	LD	B, A	Ciklusban ellenőrizzük a
C51C	1A	LD	A, (DE)	megadott és az I/O pufferbe
C51D	BE	CP	(HL)	beírt azonosítók egyezését
C51E	2017	JR	NZ, C537	Ha nem azonos, ugrás a
C520	23	INC	HL	következő csatolóra,
C521	13	INC	DE	egyébként folytatja a nevek
C522	10F8	DJNZ	C51C	ellenőrzését
C524	C1	POP	BC	Ha az azonosítók egyeznek,
C525	DDE5	PUSH	IX	még ellenőrizni kell az
C527	E1	POP	HL	egységszámot
C528	110700	LD	DE, 0007	Az I/O puffer eleje
C52B	19	ADD	HL, DE	
C52C	7E	LD	A, (HL)	Az egységszám helye
C52D	B9	CP	C	A tényleges érték
				Egyezik a megadottal?

C52E 2008	JR	NZ,C538	Ha nem: ugrás a következő csatlakozóra
C530 3E04	LD	A,04	Ha a két egységszám is azonos, akkor a kért csatolószám már egyszerű kivonással adódik
C532 90	SUB	B	
C533 4F	LD	C,A	
C534 AF	XOR	A	
C535 D1	POP	DE	
C536 C9	RET		
C537 C1	POP	BC	
C538 113000	LD	DE,0030	Ha bármi eltérés volt, tovább kell menni a következő csatolópufferre
C53B DD19	ADD	IX,DE	Az azonosító megadott memóriacíme
C53D D1	POP	DE	
C53E D5	PUSH	DE	
C53F 10D4	DJNZ	C515	
C541 3EFD	LD	A,FD	Ha egyik sem jó: hibakód és kész
C543 18F0	JR	C535	

Áttekintve a négy kernelfunkciót, jól látszik, hogy mindegyiket helyettesíteni lehet egy-két byte átírásával, a megfelelő HI-MEM rendszerváltozóban vagy az input-output táblázatban.

Ami mégis e funkciók hívása mellett szól, egyedül az, hogy a rutinok ellenőrzik a paraméterezések helyességét, s a nagy tévedéseinket következmények nélkül védik ki: nem hajtják végre a meghatározott byte-ok átírását.

E könyv szerzője sohasem hívta programból ezeket a funkciókat!

3.5.2 Az ugrótáblák

A ROM-ban ezután a képmegjelenítést vezérlő 6845-ös CRT inicializálásához szükséges paraméterek vannak tárolva. Nem lenne célszerű az adatok ismételt értelmezése, ezért egyszerűen csak felsoroljuk ezeket, a CRT 16 regisztere sorrendjében.

A képmegjelenítőt inicializáló értékek:

```
0C545H:  FE 0E 00 00 03 03 03 00 42 3C 02 4D 32 4B 40 63
CRT reg.: 0F 0E 0D 0C 0B 0A 09 08 07 06 05 04 03 02 01 02
```

Ezután még két tábla következik, mindkettő ugrótábla és mindkettőt ismerjük, használtuk már.

Az első tábla 4 címet tartalmaz. A rendszer inicializálása során az egyes eszközök első, alapbeállítását végzik el az itt megadott kezdőcímmű rutinok.

ROM-cím	A rutin címe	A rutin funkciója
C555	C9F2	A képmegjelenítő alaphelyzetbe állítása
C557	D5EC	A billentyűzet inicializálása
C559	D960	Hang alaphelyzet
C55B	D9E2	A magnetofonkezelő inicializálása

A második táblázat a funkcióhívások fő ugrótáblája. Ennek alapján történik az elágazás az egyes eszközöket meghajtó rutinok felé. A táblázatban a funkcióosztályok szerinti besorolás sorrendjében találjuk az egyes eszközök ugrótábláinak címeit.

Megjegyezzük itt is, hogy a csatolókarták helyén álló nullák a működés szempontjából teljesen érdektelenek, csak arról van szó, hogy a helyes táblakezelés érdekében a 06. osztálynak is fenn kell tartani 2 byte-ot.

A funkcióhívások fő ugrótáblája:

ROM-cím	Eszköz száma	Ugrótábla címe	Az eszköz neve
C55D	00	C974	Képmegjelenítő (VIDEO)
C55F	01	D5E3	Billentyűzet (KEY)
C561	02	CF98	Editor (EDITOR)
C563	03	D92A	Hangképzés (SOUND)
C565	04	D8FF	Nyomtató (PRINTER)
C567	05	D9BB	Magnetofon (CAS)
C569	06	0000	Csatolókarták (EXT)
C56B	07	C4AC	Kernel (KERNEL)

A két táblázat után hatalmas munka következik. Amennyire egyszerű helyzetünk volt a kernelrutinok vizsgálatánál, most annál bonyolultabb elemzések előtt állunk.

3.5.3 A video eszközmeghajtó rutinjai

A video eszközmeghajtó rutinok talán az egész ROM legnagyobb területét foglalják el: a C56DH... CF97H címtartományt! Ami egyébként nem csoda: a háromféle üzemmód, a változó színkezelési lehetőségek, a különböző grafikus

szolgáltatások, mind önmagában is eléggé bonyolult feladatot jelentenek. A hívható funkciókat nagyszámú kisebb-nagyobb szubrutin támogatja, amelyek viszont ismét más programrészeket aktiválnak. Emellett sok az egymással is összefonódó programrész, nem mindig követhető könnyen a CPU regisztereinek belépéskori és befejező állapota. Emiatt nehéz a működés minden fázisára érvényes logikai struktúrába szervezni a részfunkciókat ellátó programokat.

Megijedni azonban nem kell, természetesen a végére lehet járni mindennek, talán csak helyenként az eddig megszokottnál nagyobb figyelemre és kitartásra van szükség.

Az elmondottakkal szemben némi pihenésre ad majd módot, hogy ezen a ROM területen helyezkedik el az összes nem definiálható, tehát a TV-Computer stabil karakterkészletét meghatározó 96x10 byte-os adatterület.

3.5.3.1 A blokkfunkciók vezérlése

A ROM video eszközmeghajtóhoz tartozó része két egyszerű szubrutinnal indul: a karakterblokkok input-output-jának vezérlését végzik el ezek a rutinok úgy, hogy közben ellenőrzik a legmagasabb használható memóriacím túllépését.

A két szubrutin általánosan használható abban az értelemben, hogy a tényleges I/O-t végrehajtó szubrutinok címe paraméterként adható meg, így bármely eszközhöz jók.

Ismerjük meg a szubrutinokat!

OUT-CHARS	Meghatározott méretű adatblokk kiküldése	
in:	BC=a kiküldendő karakterek száma DE=a karakterek kezdőcíme a memóriában HL=a kiküldést megvalósító szubrutin címe	
out:	A=00 nincs hiba FA a HI-MEM átlépésének jelzése xx az átvitel sajátosságaitól függő, az egyes eszközökre is jellemző hibakód	
C56D EB	EX DE,HL	
C56E E5	PUSH HL	memóriacím
C56F D5	PUSH DE	rutincím
C570 C5	PUSH BC	hossz

C571 E5	PUSH HL	Ismét a memóriacím és a
C572 D5	PUSH DE	rutin kezdőcíme
C573 EB	EX DE,HL	Most DE=memóriacím
C574 2A190B	LD HL,(0B19)	HL=HI-MEM
C577 B7	OR A	Carry=0, a kivonáshoz
C578 ED52	SBC HL,DE	HI-MEM túllépés?
C57A 3EFA	LD A,FA	Beállítja a hibakódot
C57C D1	POP DE	rutincím
C57D E1	POP HL	memóriacím
C57E D8	RET C	Hiba esetén visszatérés, de hová?!

Teljesen meglepő programozási hiba!

Eddig sokszor voltunk tanúi professzionális szintű megoldásoknak, ahol a hardver és a szoftver a legmagasabb szintű harmóniában együttműködve valósítja meg a kívánt feladatokat!

Most viszont a naiv hibából még csak tanulni sem lehet!

A veremben tárolt 5 db kétbyte-os adatból a RET pillanatában még 3 db ott maradt, s a kiviendő karakterek száma semmiképpen nem tekinthető visszatérési címnek! A CPU mindenesetre elmegy az eredetileg BC-ben tárolt címre, de hogy ott mi történik, arra semmit sem lehet mondani: az eltérés igazi és kegyetlen!

Sajnos, a legtöbb számítógép ROM-jában találkozni lehet kisebb-nagyobb hibákkal. Köztük kellemetlenebbekkel, nehezen kideríthetőkkel. Itt egyszerűen tudomásul kell venni, hogy mindazoknak a blokk-output funkcióknak a hívásakor, amelyek erre a rutinra épülnek, nekünk, magunknak kell elvégezni az egyébként nem is túlságosan indokolt vizsgálatot, ha nem akarunk nagy kellemetlenségeket előidézni. Ha program ezen a ponton túlmegegy, akkor utána már helyesen működik.

C57F 4E	LD C,(HL)	A kijelölt memóriaterületről veszi a következő kiküldendő karaktert
C580 EB	EX DE,HL	Most DE=memóriacím, HL=rutincím
C581 CD8EC5	CALL C58E	Egy JP HL utasítással meghívja a végrehajtó rutint,
C584 C1	POP BC	majd a regiszterek vissza-
C585 D1	POP DE	töltésével előkészül a
C586 E1	POP HL	továbbiakra

C587 B7	OR	A	Ha a karakter kivitele közben
C588 C0	RET	NZ	hiba lépett fel, akkor ezzel
			a hibakóddal visszatér (must
			már helyesen)!
C589 EDA1	CPI		Egyébként tovább lép (HL+1,
			BC-1)
C58B E0	RET	FO	Ha már a teljes blokkot
			kiküldte visszatér,
C58C 18E0	JR	C56E	egyébként folytatja

Ezután következik egy byte-on a vezérlés átadása a végrehajtó rutinnak:

C58E E9	JP	HL	Ugrás a HL mutatta címre
---------	----	----	--------------------------

Most nézzük az előbbi szubrutin párját: ez a kiválasztott eszközzel meghatározott számú karaktert olvas be, s azokat a memória meghatározott területére tárolja.

IN-CHARS Meghatározott méretű adatblokk beolvasása

in:	BC=a beolvasandó karakterek száma
	DE=a kezdőcím a memóriában
	HL=a beolvasást megvalósító szubrutin címe
out:	A=00 nincs hiba
	FA a HI-MEM átlépésének jelzése
	xx a beolvasás sajátosságaitól függő, az
	egyes eszközre is jellemző hibakód

C58F E5	PUSH HL	rutincím
C590 D5	PUSH DE	memóriacím
C591 C5	PUSH BC	hossz
C592 E5	PUSH HL	
C593 D5	PUSH DE	
C594 2A190E	LD HL, (0B19)	HL=HI-MEM
C597 B7	OR A	
C598 ED52	SBC HL, DE	HI-MEM túllépés?
C59A 3EFA	LD A, FA	A hibakód előkészítése
C59C D1	POP DE	
C59D E1	POP HL	
C59E D8	RET C	Programozási hiba!
		HI-MEM túllépés esetén
		emiatt a rutin eltérülést
		okoz!
C59F CD8EC5	CALL C58E	Egyébként ugrás a HL
		kezdőcímű rutinra

C5A2 08	EX	AF,AF	Tárolja a hibakódot és a
C5A3 79	LD	A,C	beolvasott karaktert, majd
C5A4 C1	POP	BC	visszatölti a veremből a
C5A5 D1	POP	DE	regisztereket.
C5A6 E1	POP	HL	
C5A7 08	EX	AF,AF	Hibakód
C5A8 B7	OR	A	
C5A9 C0	RET	NZ	Hiba esetén visszatér,
C5AA 08	EX	AF,AF	egyébként a beolvasott
C5AB 12	LD	(DE),A	karaktert lerakja a memória
			következő címére.
C5AC AF	XOR	A	Visszatéréshez előkészíti a
			"rendben" jelzést
C5AD EB	EX	DE,HL	
C5AE EDA1	CPI		HL+1, BC-1
C5B0 EB	EX	DE,HL	
C5B1 E0	RET	PO	Ha már beolvasta a megadott
			számú karaktert:
			visszatér,
C5B2 18DB	JR	C5BF	egyébként folytatja

3.5.3.2 A fix karakterek definíciói

A ROM következő, jelentős részét a 20H...7FH közötti ASCII kódú karaktereket definiáló byte-ok töltik be. A következőkben megadjuk a 10x8 ponttal meghatározott karakterek egyenként 10-10 byte-os definícióit, amelyek értelmezése teljesen azonos a felhasználói karakterekkel kapcsolatban a 2.3 alfejezetben elmondottakkal.

ASCII kód		Cím	Karakter	Definíció
HEX	DEC			
20	(32)	C5B4	üres	00 00 00 00 00 00 00 00 00 00
21	(33)	C5BE	!	00 18 18 18 18 18 00 18 00 00
22	(34)	C5C8	"	00 36 36 36 00 00 00 00 00 00
23	(35)	C5D2	#	00 36 36 7F 36 7F 36 36 00 00
24	(36)	C5DC	\$	00 18 3E 58 3C 1A 7C 18 00 00
25	(37)	C5E6	%	00 60 66 0C 18 30 66 06 00 00
26	(38)	C5F0	&	00 10 28 28 30 54 48 34 00 00
27	(39)	C5FA	'	00 18 18 30 00 00 00 00 00 00
28	(40)	C604	(00 0C 18 30 30 30 18 0C 00 00
29	(41)	C60E)	00 30 18 0C 0C 0C 18 30 00 00
2A	(42)	C618	*	00 00 10 54 38 38 54 10 00 00
2B	(43)	C622	+	00 00 18 18 7E 18 18 00 00 00
2C	(44)	C62C	,	00 00 00 00 00 00 18 18 30 00
2D	(45)	C636	-	00 00 00 00 7C 7C 00 00 00 00

ASCII kód		Cím	Karakter	Definíció											
HEX	DEC														
2E	(46)	C640	.	00	00	00	00	00	00	18	18	00	00		
2F	(47)	C64A	/	00	00	06	0C	18	30	60	00	00	00		
30	(48)	C654	0	00	3C	66	6E	7E	76	66	3C	00	00		
31	(49)	C65E	1	00	18	38	18	18	18	18	18	00	00		
32	(50)	C668	2	00	3C	66	06	1C	30	60	7E	00	00		
33	(51)	C672	3	00	7E	06	0C	1C	06	46	3C	00	00		
34	(52)	C67C	4	00	0C	1C	2C	4C	7E	0C	0C	00	00		
35	(53)	C686	5	00	7E	60	7C	06	06	46	3C	00	00		
36	(54)	C690	6	00	3C	60	60	7C	66	66	3C	00	00		
37	(55)	C69A	7	00	7E	06	0C	18	30	60	60	00	00		
38	(56)	C6A4	8	00	3C	66	66	3C	66	66	3C	00	00		
39	(57)	C6AE	9	00	3C	66	66	3E	06	0C	38	00	00		
3A	(58)	C6B8	:	00	00	00	18	18	00	18	18	00	00		
3B	(59)	C6C2	;	00	00	00	18	18	00	18	18	30	00		
3C	(60)	C6CC	<	00	06	0C	18	30	18	0C	06	00	00		
3D	(61)	C6D6	=	00	00	00	7C	00	7C	00	00	00	00		
3E	(62)	C6E0	>	00	30	18	0C	06	0C	18	30	00	00		
3F	(63)	C6EA	?	00	3C	66	06	0C	18	00	18	00	00		
40	(64)	C6F4	@	00	3E	63	67	6B	6F	60	3C	00	00		
41	(65)	C6FE	A	00	1C	3E	63	63	7F	63	63	00	00		
42	(66)	C708	B	00	7E	63	63	7E	63	63	7E	00	00		
43	(67)	C712	C	00	3E	63	60	60	60	63	3E	00	00		
44	(68)	C71C	D	00	7E	33	33	33	33	33	7E	00	00		
45	(69)	C726	E	00	7E	60	60	7C	60	60	7E	00	00		
46	(70)	C730	F	00	7E	60	60	7C	60	60	60	00	00		
47	(71)	C73A	G	00	3E	63	60	60	67	63	3E	00	00		
48	(72)	C744	H	00	63	63	63	7F	63	63	63	00	00		
49	(73)	C74E	I	00	3C	18	18	18	18	18	3C	00	00		
4A	(74)	C758	J	00	06	06	06	06	66	66	3C	00	00		
4B	(75)	C762	K	00	63	66	6C	78	6C	66	63	00	00		
4C	(76)	C76C	L	00	60	60	60	60	60	60	7E	00	00		
4D	(77)	C776	M	00	63	77	6B	63	63	63	63	00	00		
4E	(78)	C780	N	00	66	66	76	6E	66	66	66	00	00		
4F	(79)	C78A	O	00	3E	63	63	63	63	63	3E	00	00		
50	(80)	C794	P	00	7E	63	63	7E	60	60	60	00	00		
51	(81)	C79E	Q	00	3E	63	63	63	6B	67	3E	01	00		
52	(82)	C7A8	R	00	7E	63	63	7E	6C	66	63	00	00		
53	(83)	C7B2	S	00	3E	63	60	3E	03	63	3E	00	00		
54	(84)	C7BC	T	00	7E	5A	18	18	18	18	18	00	00		
55	(85)	C7C6	U	00	63	63	63	63	63	63	3E	00	00		
56	(86)	C7D0	V	00	63	63	63	63	36	1C	08	00	00		
57	(87)	C7DA	W	00	63	63	63	6B	6B	3E	14	00	00		
58	(88)	C7E4	X	00	66	66	3C	18	3C	66	66	00	00		
59	(89)	C7EE	Y	00	66	66	3C	18	18	18	18	00	00		
5A	(90)	C7F8	Z	00	7E	06	0C	18	30	60	7E	00	00		
5B	(91)	C802	[00	3C	30	30	30	30	30	3C	00	00		
5C	(92)	C80C	\	00	00	60	30	18	0C	06	00	00	00		
5D	(93)	C816]	00	3C	0C	0C	0C	0C	0C	3C	00	00		
5E	(94)	C820	^	00	18	3C	66	42	00	00	00	00	00		

ASCII kód		Cím	Karakter	Definíció
HEX	DEC			
5F	(95)	C82A	_	00 00 00 00 00 00 00 7E 00 00
60	(96)	C834	^	00 30 30 18 00 00 00 00 00 00
61	(97)	C83E	a	00 00 00 3C 06 3E 66 3E 00 00
62	(98)	C848	b	00 60 60 7C 66 66 66 7C 00 00
63	(99)	C852	c	00 00 00 1E 30 30 30 1E 00 00
64	(100)	C85C	d	00 06 06 3E 66 66 66 3E 00 00
65	(101)	C866	e	00 00 00 3C 66 7E 60 3C 00 00
66	(102)	C870	f	00 0C 18 18 3C 18 18 18 00 00
67	(103)	C87A	g	00 00 00 3E 66 66 66 3E 06 3C
68	(104)	C884	h	00 60 60 7C 66 66 66 66 00 00
69	(105)	C88E	i	00 18 00 38 18 18 18 3C 00 00
6A	(106)	C898	j	00 18 00 38 18 18 18 18 18 70
6B	(107)	C8A2	k	00 60 60 66 6C 78 6C 66 00 00
6C	(108)	C8AC	l	00 18 18 18 18 18 18 18 00 00
6D	(109)	C8B6	m	00 00 00 76 6B 6B 6B 6B 00 00
6E	(110)	C8C0	n	00 00 00 7C 66 66 66 66 00 00
6F	(111)	C8CA	o	00 00 00 3C 66 66 66 3C 00 00
70	(112)	C8D4	p	00 00 00 7C 66 66 66 7C 60 60
71	(113)	C8DE	q	00 00 00 3E 66 66 66 3E 06 06
72	(114)	C8E8	r	00 00 00 36 38 30 30 30 00 00
73	(115)	C8F2	s	00 00 00 1E 30 1C 06 3C 00 00
74	(116)	C8FC	t	00 18 18 3C 18 18 18 0C 00 00
75	(117)	C906	u	00 00 00 66 66 66 66 3E 00 00
76	(118)	C910	v	00 00 00 66 66 66 3C 18 00 00
77	(119)	C91A	w	00 00 00 63 63 6B 3E 14 00 00
78	(120)	C924	x	00 00 00 66 3C 18 3C 66 00 00
79	(121)	C92E	y	00 00 00 66 66 66 66 3E 06 3C
7A	(122)	C938	z	00 00 00 7E 0C 18 30 7E 00 00
7B	(123)	C942	{	00 0E 18 18 70 18 18 0E 00 00
7C	(124)	C94C		00 18 18 18 00 18 18 18 00 00
7D	(125)	C956	}	00 70 18 18 0E 18 18 70 00 00
7E	(126)	C960	~	00 00 00 33 6B 66 00 00 00 00
7F	(127)	C96A		00 7E 7E 7E 7E 7E 7E 7E 00 00

Az előbbi karakterdefiníciókat a ROM megjelenítő rutinja használja, természetesen alkalmas esetben a saját programjainkban is felhasználhatjuk.

Az itt közölt byte-okat -- ismerve már a megjelenítés alapvető mechanizmusát -- a videomemória 10, egymástól 40H távolságú byte-jaiba írva, 2-es üzemmódban a kiválasztott karakter képe jelenik meg a képernyőn. A 2-es mód lényeges, hiszen a videomemória byte-jait csak ez az üzemmód értelmezi bitenként egy pontnak.

Következik a képmegjelenítéssel kapcsolatos hívások ugrótáblája. Elérése a funkcióhívások vezérlőrutinjából, a 0C55DH főtáblázat segítségével, már ismert módon történik.

3.5.3.3 A video ugrótábla

ROM	Tartalom	Jelentés
C974	0D	A hívható VIDEO funkciók száma
C975	C9A9	0. funkcióhoz tartozó rutin címe
C977	CC94	1. funkció (VID-CHOUT)
C979	CC86	2. funkció (VID-BKOUT)
C97B	CF4B	3. funkció (CH-POZ)
C97D	C9F4	4. funkció (VID-MODE)
C97F	CA49	5. funkció (CLS)
C981	CADA	6. funkció (B-ABS)
C983	CAD7	7. funkció (B-REL)
C985	CBF3	8. funkció (B-ON)
C987	CBFF	9. funkció (B-OFF)
C989	CD48	10. funkció (PAINT)
C98B	CF2C	11. funkció (CH-DEF)
C98D	CA38	12. funkció (PAL-DEF)

3.5.3.4 Videorutinok

Néhány gyakran használt, apró szubrutinnal kezdődik a videofunkciók kidolgozása. Ezeket, amikor a környezetétől való elválasztás indokolt, a V betűjellel és utána írt sorszámmal azonosítjuk.

V1 Ez a nagyon fontos és sokszor hívott szubrutin érdekes
C98F módon oldja meg az összes videomemóriában dolgozó többi rutin memórialapozással kapcsolatos tennivalóit. A programokból meghívva belapozza a videomemóriát, majd visszatér a hívó programhoz és ott dolgozik a rutint lezáró RET-ig. Végül visszaállítja az eredeti memóriakonfigurációt, s az egyel magasabb hierarchiaszintű programhoz tér vissza (vagyis oda, amely az általa is kiszolgált rutint hívta).

Ügyes megoldás és garantált védelmet nyújt a szerteágazó rutinok használata közben a memórialapozással kapcsolatos esetleges mulasztásokkal szemben.

C98F E1	POP HL	Kiemeljük a visszatérési címet (a hívó rutinhoz)
C990 3A0300 C993 F5	LD A, (0003) PUSH AF	Tároljuk az aktuális lapozási konfigurációt

C994 3E50	LD	A,50	Belapozzuk a videomemóriát,
C996 320300	LD	(0003),A	természetesen a P-SAVE-be is
C999 D302	OUT	(02),A	beírva
C99B E5	PUSH	HL	Most visszatevesszük a hívó
			rutinhoz vezető címet és
C99C 21A1C9	LD	HL,C9A1	azonnal ki is cseréljük a
C99F E3	EX	(SP),HL	C9A1H címmel
C9A0 E9	JP	HL	Látható, hogy most a vezérlés
			visszakerül a hívó rutinhoz,
			de az ottani RET visszahozza,
			ide.
C9A1 08	EX	AF,AF	Ezután az A-t tárolja,
C9A2 F1	POP	AF	visszaállítja az eredeti
C9A3 320300	LD	(0003),A	memórialapozási
C9A6 D302	OUT	(02),A	konfigurációt,
C9A8 08	EX	AF,AF	visszatölti az A-t
C9A9 C9	RET		és ugrás a felsőbb szintű
			rutinhoz

Amint az ugrótáblából kitűnik, a 0. videofunkcióhoz mindössze ez a lezáró RET tartozik.

VIDEO 00		Az IT alprogram kiszolgálása
00		
C9A9 C9	RET	Semmi

A következő szorgos kis szubrutin ugrótáblákat kezel.

V2	A	HL regiszterpárban megadott kezdőcímű
C9AA		ugrótáblából az A regiszterben megadott sorszámú
		címet a DE regiszterpárba teszi

in: HL=ugrótábla eleje
A=sorszám
out: DE=cím az ugrótáblából

C9AA 87	ADD	A,A	Mivel a cím kétbyte-os, ezért
C9AB 5F	LD	E,A	az A-beli érték kétszerese
C9AC 1600	LD	D,00	adja a táblázat elejéhez
C9AE 19	ADD	HL,DE	viszonyított eltolást
C9AF 5E	LD	E,(HL)	
C9B0 23	INC	HL	
C9B1 56	LD	D,(HL)	DE-be beolvassa a címet
C9B2 C9	RET		és kész

A következő két rutin egyszerű koordináta-transzformációkat végez. Közismert, hogy a grafikus pontokra való hivatkozásokban a képernyő bal alsó sarkán levő ponthoz rendeljük a (0,0) koordinátapárt.

Ezzel szemben sokszor kell a sor jobb széléhez és a képernyő felső sorához viszonyított helyzet. Az ehhez szükséges kivonásokat végzik el az itt következő rutinok.

V3 A képernyő jobb széléhez viszonyítva megfordítja
C9B3 a pontokhoz tartozó fizikai x koordinátákat

in: BC=fizikai x koordináta
out: HL= 03FFH - BC (dec. 1023-BC)

```
C9B3 21FF03       LD   HL,03FF
C9B6 B7           OR   A
C9B7 ED42        SBC  HL,BC
C9B9 C9           RET
```

V4 A képernyő felső sorához viszonyítva megfordítja
C9BA a pontokhoz tartozó fizikai y koordinátákat

in: DE=fizikai y koordináta
out: HL= 03BF - DE (dec. 959-DE)

```
C9BA 21BF03       LD   HL,03BF
C9BD B7           OR   A
C9BE ED52        SBC  HL,DE
C9C0 C9           RET
```

A következő szubrutin a HL regiszterpárbeli értéket 2-vel, 4-gyel, 8-cal, 16-tal stb.-vel osztja el. Tudjuk, hogy az egyes üzemmódok közötti értékelterésekhez ilyen osztásokra gyakran szükség van.

V5 A HL regiszterpárban hozott értéket az üzemmódnak
megfelelően: 2-vel, 4-gyel vagy 8-cal osztja
(ezzel pl. a fizikai x koordinátákból kiszámítja
a logikai értéket).

```
C9C1 3A730B       LD   A,(0B73)           üzemmód: 0, 1, 2
C9C4 47           LD   B,A
C9C5 04           INC  B                B=1, 2 vagy 3
```

V51 Ha itt lépünk be a rutinba, akkor a HL értékét
C9C6 annyiszor osztja 2-vel, amennyi a B értéke.

C9C6 CB3C	SRL H	H és L bitjeit jobbra
C9C8 CB1D	RR L	lépteti: osztás 2-vel (B- szer)
C9CA 10FA	DJNZ C9C6	
C9CC C9	RET	

Az itt következő szubrutin már komoly előkészítéseket végez. Sajnos 4 másik szubrutint is hív, amelyeket csak később ismerünk meg. Ezekről itt természetesen megadjuk a lényeges tudnivalókat azzal, hogy rövidesen sor kerül ezek részletes ismertetésére is.

V6 A megjelenítéshez szükséges fontos változók és
C9CD IY előkészítése

C9CD CD6DCB	CALL CB6D	V13: az L-STYLE (0B4C)-ben megadott vonaltípust megváltoztató VID értéket tölt 0B83H-ba
C9D0 CD05CC	CALL CC05	V20: az üzemmódnak megfelelő, PAPER színű vonal VID értékét adja
C9D3 32930B	LD (0B93),A	A PAPER színű vonalat tárolja
C9D6 114000	LD DE,0040	
C9D9 CD8FCB	CALL CB8F	V14: az L-MODE (0B4B)-ben megadott keresztezési módnak megfelelő címre állítja IY-t
C9DC CD0DCC	CALL CC0D	V21: az üzemmódnak megfelelő INK színű vonal VID értékét adja
C9DF 32940B	LD (0B94),A	Az INK színű vonalat tárolja.
C9E2 3A750B	LD A,(0B75)	A megcímezett ponthoz tartozó byte tartalma a videomemóriában, így a ponthoz tartozó VID-érték
C9E5 4F	LD C,A	
C9E6 2A760B	LD HL,(0B76)	Az adott byte VID címe
C9E9 C9	RET	

Hát ez így egy kicsit súlyos, nagyon sok az új információ! De nemsokára tisztulni fog!

Most egy rövid adatterület következik: a rendszer inicializálásakor a video-eszközmeghajtó alapbeállításához szükséges palettaregiszter értékeket innen olvassa be a rendszer.

A palettaregiszterek alapértékei 4-es üzemmódban:

ROM-cím	palettaregiszter (színes beállítás)				palettaregiszter (fekete-fehér beállítás)			
	00	01	02	03	00	01	02	03
C9EA	00	50	44	41	00	55	50	44

Az ezt követő cím az inicializáló videorutin belépési pontja. A C regiszterbe 01 értéket írva a program tulajdonképpen a 4-es funkcióhívás rutinját hajtja végre és C=01 miatt triviálisan 4 színű üzemmódot kapcsol.

VIDEO INIC. 4-es üzemmód beállítása
C9F2

C9F2 0E01 LD C,01 A rutin a 4-es funkcióhíváshoz tartozó résszel folytatódik, a következő sorban

VIDEO 04 VID-MODE
04 A video üzemmód beállítása

in: C=az üzemmód kódja
 00: 2-es mód
 01: 4-es mód
 02: 16-os mód
out: A=00 rendben
 F7 nincs ilyen üzemmód

C9F4 79 LD A,C A megadott üzemmód
C9F5 FE03 CP 03
C9F7 3EF7 LD A,F7 Ha 2-nél nagyobb
C9F9 D0 RET NC visszatérés hibakóddal

C9FA AF XOR A Alaphelyzetbe állítja a
C9FB 324B0B LD (0B4B),A vonalkeresztet,
C9FE 324E0B LD (0B4E),A PAPER szint,
CA01 32500B LD (0B50),A karakterfelülírási módot és a
CA04 3C INC A
CA05 324C0B LD (0B4C),A vonaltípust

CA08 B9	CP	C	
CA09 300A	JR	NC,CA15	A=01, tehát akkor ugrik előre, ha 2-es vagy 4-es üzemmód kell: INK=1
CA0B DB59	IN	A,(59)	Itt a b6 bit a színikapcsoló állapota
CA0D 07	RLCA		
CA0E 07	RLCA		A 6. bitet nézi
CA0F 3E0F	LD	A,0F	Fekete-fehér állásban a fehér (0F), színesben a zöld (0C) szint állítja be A-ban
CA11 3002	JR	NC,CA15	
CA13 3E0C	LD	A,0C	
CA15 324D0B	LD	(0B4D),A	INK: 2-es és 4-es módban az 1. palettaregiszter, 16-osban: fehér vagy zöld szín lesz
CA18 79	LD	A,C	
CA19 32730B	LD	(0B73),A	Feljegyzzi az üzemmódot
CA1C C5	PUSH	BC	
CA1D CD49CA	CALL	CA49	Képernyőtörlés
CA20 C1	POP	BC	
CA21 79	LD	A,C	
CA22 21130B	LD	HL,0B13	A 06-os port kópiája: az üzemmódot a b1-b0 bitekbe kell írni (a többi nem változhat!)
CA25 AE	XOR	(HL)	b7...b2-re a többi bit,
CA26 E6FC	AND	FC	b1 és b0 =0,
CA28 A9	XOR	C	hozzáírja az üzemmódot,
CA29 77	LD	(HL),A	az új üzemmód szerinti bitekkel visszatölti és kiküldi a portra, ezzel ténylegesen elvégzi az üzemmód kapcsolását
CA2A D306	OUT	(06),A	

Még a paletta regisztereket kell beállítani.

CA2C DB59	IN	A,(59)	A 6. bit a színikapcsoló állapotát mutatja
CA2E 07	RLCA		
CA2F 07	RLCA		Ezt visszük a carry-be
CA30 11EEC9	LD	DE,C9EE	A színikapcsoló állapota szerint DE-t a megfelelő, palettaregiszterbe töltendő adatokra állítjuk
CA33 3003	JR	NC,CA38	
CA35 11EAC9	LD	DE,C9EA	

A program ezután átfolyik a 12-es videofunkciót végrehajtó szubrutinra, Belépéskor ez a rutin éppen a DE regiszterpárban várja a palettaregiszterbe betöltendő adatok címét, tehát úgy, ahogyan most elő van készítve.

VIDEO 12 PAL-DEF
0CH Palettaszínek definiálása

in: DE=annak a 4 byte-os memóriaterületnek a címe,
amely a palettaregiszterbe betöltendő értékeket
tartalmazza
out: A=00
(mindig végrehajtja)

CA38 1A	LD	A, (DE)	
CA39 D360	OUT	(60), A	
CA3B 13	INC	DE	Amint látszik, itt semmi
CA3C 1A	LD	A, (DE)	különös nem történik: a DE
CA3D D361	OUT	(61), A	mutatta területről egymás
CA3F 13	INC	DE	után beolvassuk az adatokat,
CA40 1A	LD	A, (DE)	amelyeket a 60H-61H-62H-63H
CA41 D362	OUT	(62), A	portokra küldve, betöltjük a
CA43 13	INC	DE	0., 1., 2. és 3.
CA44 1A	LD	A, (DE)	palettaregisztereket. A töb-
CA45 D363	OUT	(63), A	bi már a hardver dolga
CA47 AF	XOR	A	
CA48 C9	RET		

Ismét egy jelentős ponthoz érkezünk: a ROM következő címén az 5-ös funkcióhívás végrehajtó szubrutinja, a képernyőtörlés kezdődik.

VIDEO 05 CLS
05H Képernyőtörlés

in: -
out: -

CA49 CD8FC9	CALL	C98F	V1: a lapozások intézése. Már ismerjük
CA4C CDD4CF	CALL	CFD4	EDITOR INIC. Ez a rutin az editort hozza alapállapotba. Látható jele: a kurzor alaphelyzete
CA4F CD05CC	CALL	CC05	V20: az üzemmódnak megfelelő, PAPER színű vonal VID értékét adja.
CA52 210080	LD	HL, 8000	A videomemória kép megjelenítésre használt részét (15 kbyte) a papírszínnek megfelelő értékkel tölti fel. A képernyő most már üres
CA55 110180	LD	DE, 8001	
CA58 77	LD	(HL), A	
CA59 01FF3B	LD	BC, 3BFF	
CA5C E0B0	LDIR		

CASE CDFFCB	CALL CBFF	A 09-es videofunkciót hívja: B-OFF, "felemelt tollmozgatás" jelzése
CA61 47	LD B,A	A=00
CA62 4F	LD C,A	BC=0000 és DE=0000:
CA63 57	LD D,A	a fizikai koordináták
CA64 5F	LD E,A	alapértéke

Ettől kezdve a rutint más helyekről is hívjuk, de a CLS is használja.

V31
CA65 A fizikai pontkoordinátákból a videomemória írásához szükséges paraméterek előállítására és tárolására. Következésképpen rendkívüli jelentőségű feladatokat ellátó rutin

CA65 ED537E0B	LD (0B7E),DE	Beállítja a belső változókat
CA69 ED437C0B	LD (0B7C),BC	
CA6D 2A7C0B	LD HL,(0B7C)	A fizikai x (0...1023)
CA70 E5	PUSH HL	(röv.: xf)
CA71 0604	LD B,04	V5.1: xf-t 16-tal osztja, így a pont soron belüli byte-címét kapja (00...40H)
CA73 CDC6C9	CALL C9C6	

CA76 E3	EX (SP),HL	Ezt egyelőre a verembe teszi, elővéve ismét xf (figyeljük meg a csere egyszerűségét!)
---------	------------	---

CA77 CDC1C9	CALL C9C1	V5: az üzemmód szerint, xf osztása után a logikai x koordinátát kapja (0..512, 0..256, 0..128)
-------------	-----------	--

CA7A 22780B	LD (0B78),HL	és tárolja
CA7D 4F	LD C,A	V5 A-ban az üzemmód kódját hagyta (0-1-2)

CA7E 7D	LD A,L	A=a logikai x koordináta (x1) alsó byte-ja
---------	--------	--

CA7F 21B4CA	LD HL,CAB4	Most HL egy olyan byte-ra mutat, amely a VID-ben egy byte-hoz tartozó bal szélső pontot jelenítene meg
CA82 09	ADD HL,BC	

CA83 E607	AND 07	A byte-on belüli pontpozícióhoz elég az x1 alsó 3 bitje: 8-anként biztosan ismétlődik
CA85 3C	INC A	A=01...08 (DJNZ-hez!)
CA86 47	LD B,A	

Ez a cikluskezelési megoldás rendkívül szellemes. Figyeljük meg azt a technikát, ahogyan a DJNZ-ciklus kellemetlenségeit a program lehengetlő egyszerűséggel és könnyedséggel kezeli!

CA87 7E	LD A,(HL)	A-ban az üzemmód szerinti szélső pontot kijelölő érték (80-88-AAH)
CA88 1E0F	LD E,0F	

Ez így üres utasítással egyenértékű! De így lefut a DJNZ első menete, helyrehozva az INC A hatását, s utána pedig a ciklus már: a 0FH kódra tér vissza:

CA89 0F	RRCA	Tehát végül is a ciklus (B-1)-szer végez RRCA-t
CA8A 10FD	DJNZ CA89	
CA8C 32750B	LD (0B75),A	Az így nyert byte a VID megfelelő byte-jában már a megadott sorszámú pontot jelöli ki
CA8F ED5B7E0B	LD DE,(0B7E)	Fizikai koordináta: yf
CA93 21BF03	LD HL,03BF	
CA96 AF	XOR A	Az irányt megfordítjuk (most már fentről le)
CA97 ED52	SBC HL,DE	
CA99 0602	LD B,02	
CA9B CDC6C9	CALL C9C6	V5.1: 4-gyel osztva kapjuk a logikai y-t (yl), amit tárolunk.
CA9E 227A0B	LD (0B7A),HL	

Ezután előállítjuk a tényleges videomemóriabeli címet.

CAA1 65	LD H,L	yl*256 (dec.)
CAA2 CB3C	SRL H	
CAA4 1F	RRA	H,A-ban yl*128 (dec.)
CAA5 CB3C	SRL H	
CAA7 1F	RRA	
CAA8 6F	LD L,A	HL=yl*64 (dec.)
CAA9 C1	POP BC	BC=a soron belüli byte száma
CAAA 09	ADD HL,BC	HL=yl*64+(a soron belüli byte címe) (dec.),
CAAB 110080	LD DE,8000	
CAAE 19	ADD HL,DE	ez pedig már a keresett VID cím!
CAAF 22760B	LD (0B76),HL	Tároljuk a pont VID-címét
CAB2 AF	XOR A	Hibátlan (valóban!)
CAB3 C9	RET	

Egy nagyon ötletesen kivitelezett és jól megszerkesztett rutint láttunk egy nem éppen könnyű feladat megoldására.

Nyilvánvaló, hogy az előbbieken nem a CLS működése volt a legfontosabb, hanem inkább a V31-es belépési ponttól követett működés. Többször fogunk még rá hivatkozni.

A rutin után itt az a három érték, amely az egyes üzemmódokban a byte-on belüli bal szélső pontot jelöli ki a videomemóriában:

ROM- cím	üzemmód:		
	2-es	4-es	16-os
CAB4	80	88	AA

Majd pedig arra a szubrutinra kerül a sor, amely már közvetlenül a videomemória byte-jait írja meg tetszőleges megjelenítési szituációban. A változatos funkciónak megfelelően a rutinba több ponton lehet belépni.

A HL-regiszterpár minden esetben a videomemória kiválasztott byte-jára mutat. A C regiszterben az új képtartalomnak megfelelő érték van, ezenkívül IY a vonalkeresztezési módnak megfelelő, a képátírást megvalósító utasítássorozat kezdetére mutat.

A többféle funkció ellátása és az előart feltételek figyelembevételére itt is figyelemreméltó rövidséggel valósul meg.

V7
CAB7

A karakterek kiírásának segédrutinja: egy pont megjelenítését végzi

in: Carry=1, ha INK színű pont következik
A=karakterfelülírási mód
HL=az aktuális byte címe a videomemóriában
C=az aktuális pont byte-on belüli pozícióját kijelölő érték
IY=a tényleges videomemória-írást a megadott vonalkeresztezési mód szerint végző rutin címe (0: CACE, 1:CAD3, 2: CACD és 3: CACF)

CAB7 300C	JR	NC,CAC5	Ha nem INK pontról van szó, ugrás előre
CAB9 E601	AND	01	Az INK pontnak érvényesülnie kell?
CABB 2803	JR	Z,CAC0	Ha igen: ugrás előre,
CABD C9	RET		egyébként kész!

A következő cím a különböző stílusú vonalak húzásához használt belépési pont, a V7 rutin is ezzel folytatódik.

V8 A vonalhúzáshoz használt segédrutin: egy pontot
 CABE dolgoz fel
 in: Carry, HL, C és IY tartalma ugyanaz mint V7-nél

CABE 3008 JR NC,CAC8 Ha nem INK pont, ugrás

V8.1 A tintaszínű pont lerakása.
 CAC0 in: HL, C és IY tartalma ugyanaz mint V7-nél

CAC0 3A940B LD INK színű vonalbyte
 A, (0B94) A pont megjelenítése
 CAC3 FDE9 JP IY

A karakterkiírás PAPER színű pont esetén itt folytatódik.

CAC5 E602 AND 02 A PAPER szín érvényesül?
 CAC7 C0 RET NZ Ha nem: kész

CAC8 3A930B LD PAPER színű vonalbyte
 A, (0B93) A pont megjelenítése
 CACB FDE9 JP IY

Ehhez már csak az egyes pontok tényleges megjelenítésének módszerét kell megismernünk. A következő szubrutin 4 belépési ponttal -- a 4 féle vonalkeresztezési módnak megfelelően -- oldja meg ezt a feladatot. A videomemória adott byte-jában lévő, a többi ponthoz tartozó biteket nem szabad megváltoztatni.

V9 Egy pont megjelenítéséhez tartozó bitek átírása
 CACD a videomemóriában
 in: HL=a byte címe a videomemóriában
 C=a megjelenítendő pont kijelölő bitek
 A= INK vagy PAPER színű vonalat adó byte aszerint,
 hogyan milyen pontot ábrázolunk

CACD A6 AND (HL) A MODE 2 szerinti belépési pont

CACE AE XOR (HL) MODE 0 esetén itt lép be

CACF A1 AND C MODE 3 belépés

CAD0 AE XOR (HL)

CAD1 77 LD (HL),A

CAD2 C9 RET

CAD3 A1 AND C MODE 1-hez tartozó belépési
 pont

CAD4 B6 OR (HL)

CAD5 77 LD (HL),A

CAD6 C9 RET

Az egyes bitek sorsát itt nem követjük nyomon a különböző vonalkeresztelési módok szerint. A leírt egyszerű bitműveletek alapján a működést a kedves Olvasó maga is átgondolhatja.

Összefoglalva: a V7 - V9 rutinok tehát a videomemória HL regiszterpárban címzett byte-ján, a C regiszterbeli bitekkel meghatározott pontot dolgozzák fel, INK vagy PAPER színű pontként jelenítik meg. Közben figyelembe veszik a karakterfelülírási és vonalkeresztelési módokkal a felhasználó által előírt, speciális körülményeket is.

Ezeket a rutinokat használja valamennyi olyan program ész, amely közvetlenül a videomemóriába kell, hogy dolgozzon.

```
VIDEO 07          B-REL
07H              A  "toll"   elmozgatása   megadott   relatív
                  koordinátákkal

                in:  BC=x irányú relatív elmozdulás
                   DE=y irányú relatív elmozdulás
                out: A=00 nincs hiba;
                   F9 lemenne a képernyőről
```

A funkcióhíváshoz önállóan mindössze egy szubrutinhívás tartozik, a folytatás már a másik funkcióhívás része.

```
CAD7 CD7ACC      CALL CC7A          Előállítja az abszolút
                  koordinátákat (V24)
```

```
VIDEO 06          B-ABS
06              A  sugár (beam) elmozgatása megadott abszolút
                  koordinátapozícióra

                in:  BC=az új x koordináta
                   DE=az új y koordináta
                out: A=00 rendben;
                   F9 az adott pozíció nincs a képernyőn
```

```
CADA CDBFC9      CALL C98F          V1: a lapozások intézése
CADD 3EF9        LD   A,F9
CADF CDB3C9      CALL C9B3          V3: ellenőrzi x értékét
CAE2 D4BAC9      CALL NC,C9BA        V4: ellenőrzi y értékét
CAE5 D8          RET   C          Ha bármelyik rossz: a
                                  hibakóddal visszatér
```


CAE6 3A740B	LD	A, (0B74)	A toll helyzete
CAE9 B7	OR	A	Felemelt toll?
CAEA CA65CA	JP	Z, CA65	Ha igen: V31. Előállítja és betölti a kijelölt ponthoz tartozó változókat és kész is

Ha azonban a toll le van téve, akkor a megelőző pontból az új pontba való mozgás közben egyenes vonalat kell húzni!

Sajnos helyhiány miatt nincs mód arra, hogy az itt következő egyenesrajzoló rutin matematikai vonatkozásait megismerjük. Valójában ez egy speciális alkalmazói feladat, olyan szolgáltatás, amely nem tartozik a számítógép szigorú értelemben vett alapműködése körébe.

A szerző azonban -- eredetileg matematika tanár lévén -- mégsem tudja megállni, hogy legalább elveiben ne ismertesse a közlésre kerülő algoritmus fő gondolatmenetét.

A többféle lehetséges eljárás közül az itt alkalmazott megoldás lényege a következő.

Figyeljük a kezdő- és a végpont közötti x és y irány eltérések viszonyát és egyesével, lépésenként haladva, arányosan többször lépünk a nagyobb eltérés irányába, mint a másikba. Itt a hangsúly az arány követésén van, mert ez szabja meg a kapott egyenes meredekségét.

A kivitelezés módját illetően pedig csak annyit, hogy az arányos lépegetéshez elég, ha egy kezdőértékből az x irányú eltérést mindig kivonjuk, az y irányú eltérést pedig hozzáadjuk, s az eredmény előjelváltásainál változtatunk lépésirányt.

Az elmondottak aránylag egyszerű szerkezetű programmal megvalósíthatók. A pontok byte-okon belüli léptetése, valamint az előjelekkel meghatározott irányú mozgások realizálása még így is eléggé helyigényes, bár gyors.

CAED D9	EXX		A másodlagos regisztereket
CAEE ED4B780B	LD	BC, (0B78)	használva a verembe töltjük
CAF2 ED5B7A0B	LD	DE, (0B7A)	a kezdőpont x és y logikai
CAF6 C5	PUSH	BC	koordinátáit, IY-t a
CAF7 D5	PUSH	DE	vonalkeresztelési mód
CAF8 CDCDC9	CALL	C9CD	szerinti címre, HL-t a
CAFB D9	EXX		kezdőpont VID címére
			állítjuk és C-be töltjük a
			kezdőpont bitképét (V6).
CAFC CD65CA	CALL	CA65	V31: az előbbi adatok a
			végpontra
CAFF ED4B780B	LD	BC, (0B78)	
CB03 ED5B7A0B	LD	DE, (0B7A)	
CB07 AF	XOR	A	A-ban az előjeleket követjük
			majd

CB08 EB	EX DE,HL	
CB09 D1	POP DE	
CB0A CD61CB	CALL CB61	V12: az y irányú eltérés (előjele A-ban)
CB0D D1	POP DE	Azután ugyanígy, a vízszintes eltérést nézzük.
CB0E E5	PUSH HL	A V12 a képezett különbségek előjelét okosan tartja nyilván (mindig balra lépteti A bitjeit)
CB0F 60	LD H,B	
CB10 69	LD L,C	
CB11 CD61CB	CALL CB61	

Most az A regiszter jelentése:

A=00: mindkét eltérés pozitív (a végpont VID címe nagyobb)
A=01 : az x irányú eltérés negatív (balra kell haladni)
A=02 : az y irányú eltérés negatív (felfelé kell menni)
A=03 : mindkét eltérés negatív (balra és fel kell menni)

Ezekhez a léptetésekhez egy-egy szubrutinra lesz szükség, címeiket itt is célszerű egy ugrótáblába gyűjteni, ahonnan az A előbbi értékeivel egyszerűen elérhetőek lesznek.

CB14 E5	PUSH HL	
CB15 21D8CB	LD HL,CBDE	Az ugrótábla címe
CB18 CDAAC9	CALL C9AA	V2: a rutin címe DE-ben, ezt tároljuk is
CB1B ED53860B	LD (0B86),DE	

Most tisztázzuk a két eltérés viszonyát:

CB1F E1	POP HL	A vízszintes és a függőleges eltérés
CB20 D1	POP DE	
CB21 E5	PUSH HL	
CB22 ED52	SBC HL,DE	Melyik nagyobb?
CB24 E1	POP HL	
CB25 3801	JR C,CB28	DE-ben a nagyobb eltérés, HL-ben és BC-ben pedig a kisebb eltérés marad. Hogy melyik milyen, azt ismét az A azonosítja.
CB27 EB	EX DE,HL	
CB28 44	LD E,H	
CB29 4D	LD C,L	
CB2A D5	PUSH DE	
CB2B CE00	ADC A,00	

Mivel a V2 ugrótáblakezelő az A-t szorozta 2-vel, az eredeti 00...03 értékből ott 00, 02, 04, 06 lett, most pedig a két eltérés viszonyát mutató carry-bitet még hozzáadva 00...07 lehet. Ez tökéletesen le is fedti az összes figyelembeveendő lehetőséget:

A=00: mindkét eltérés pozitív, az x eltérése nagyobb
A=01: mindkét eltérés pozitív, de y eltérése nagyobb
A=02: x eltérése negatív és abszolút értékben ez a nagyobb

A=03: x eltérése negatív és abszolút értékben y eltérése a nagyobb
A=04: y eltérése negatív és abszolút értékben x eltérése a nagyobb
A=05: y eltérése negatív, de ennek abszolút értéke nagyobb
A=06: mindkét eltérés negatív és abszolút értékben x eltérése a nagyobb;
A=07: mindkét eltérés negatív és abszolút értékben y eltérése a nagyobb.

E feltételek szerint kell majd az elején említett összeadásokat és kivonásokat végezni, a szükséges byte-on belüli pontléptetésekkel és alapvető iránytartással együtt. Ehhez ismét egy ugrótáblát célszerű használni.

CB2D ED52	SBC HL,DE	A kisebb és a nagyobb eltérés különbsége. Ez egy negatív szám, ez szolgál majd az egyik irányú számoláshoz.
CB2F E5	PUSH HL	A kisebb, valamint a nagyobb eltérés felének különbsége.
CB30 CB3A	SRL D	Ezt használja a program
CB32 CB1B	RR E	kezdőértékként.
CB34 19	ADD HL,DE	

Ami azt jelenti, hogy ha a nagyobb eltérés a kisebbik kétszeresénél is nagyobb, akkor abban az irányban kezdődnek a léptetések.

CB35 E5	PUSH HL	Itt van az előbb említett ugrótábla
CB36 21E3CB	LD HL,CBE3	
CB39 CDAAC9	CALL C9AA	V2: ugrócím DE-ben, ezt is eltesszük Itt fogjuk tehát az előjelváltásokat figyelni, DE-ben van a kivonásokhoz, BC-ben pedig a növelésekhez tartozó érték. Ez a nagyobbik eltérés, a léptetéseket eszerinti ciklusban végezzük
CB3C ED53840B	LD (0B84),DE	
CB40 E1	POP HL	
CB41 D1	POP DE	
CB42 D9	EXX	
CB43 E3	EX (SF),HL	
CB44 7C	LD A,H	Ciklusparaméterek (most egy byte nem elég) Ez a kezdőpont VID címe, indul a ciklus, az első két sort először átugorjuk
CB45 45	LD B,L	
CB46 04	INC B	
CB47 3C	INC A	
CB48 08	EX AF,AF	
CB49 E1	POP HL	
CB4A 1804	JR CB50	

CB4C 08	EX	AF,AF	
CB4D CDA8CB	CALL	CBA8	V15: az ugrótáblák szerint elvégzi a pontok és a VID címek léptetését
CB50 3A830B	LD	A, (0B83)	A vonaltípushoz tartozó VID byte
CB53 07	RLCA		Ezt is léptetjük.
CB54 32830B	LD	(0B83),A	
CB57 CDBECA	CALL	CABE	V8: a vonaltípus szerint kiteszük a pontot
CB5A 10F1	DJNZ	CB4D	Az előbbieket ismételjük, amíg csak el nem érünk a második pontig
CB5C 08	EX	AF,AF	
CB5D 3D	DEC	A	
CB5E 20EC	JR	NZ, CB4C	
CB60 C9	RET		

Most megismerjük az eltéréseket előállító kis rutint.

V12 A HL és DE különbségének abszolút értékét adja,
CB61 az A regiszterben jelzi a kivonás körülményeit

in: HL=a kisebbítendő
DE=a kivonandó

out: HL=a különbség abszolútértéke
A=az eredeti érték 2-szerese és
+ 0, ha HL nem kisebb DE-nél;
+ 1, ha HL kisebb, mint DE

CB61 ED52	SBC	HL,DE	
CB63 3006	JR	NC,CB6B	Ha HL nem kisebb: ugrás
CB65 EB	EX	DE,HL	
CB66 210100	LD	HL,0001	
CB69 ED52	SBC	HL,DE	Szorzás (-1)-gyel,
CB6B 17	RLA		A és a carry léptetése
CB6C C9	RET		és kész

Az előbbi nagy munka után, pihentetésül még egyszerű segédrutin:

V13 Az L-STYLE (0B4CH) változóba beírt vonaltípusnak
CB6D megfelelő bitképű byte-ot tesz a 0B83H változóba

CB6D 3A4C0B	LD	A, (0B4C)	L-STYLE (1...14 dec.)
CB70 3D	DEC	A	A=00...0DH, ebből csak az alsó
CB71 E60F	AND	0F	4 bit kell

CB73 4F	LD	C,A	
CB74 217FCB	LD	HL,CB7F	A vonaltípus bitképeit tartalmazó táblázat
CB77 0600	LD	B,00	BC=a kért sorszám, így
CB79 09	ADD	HL,BC	HL-ben az elem címe
CB7A 7E	LD	A,(HL)	A kért vonaltípus bitképét a
CB7B 32830B	LD	(0B83),A	belső változóba töltjük és
CB7E C9	RET		kész

A következő címeken pedig itt jön maga a táblázat is:

ROM-cím	Vonaltípus sorszám (dec)	A vonalelem értéke	bitképe	Illusztráció
CB7F	1	FF	11111111	*****
CB80	2	AA	10101010	* * * * *
CB81	3	CC	11001100	** ** ** **
CB82	4	EE	11101110	*** *** **
CB83	5	88	10001000	* * * *
CB84	6	DA	11011010	** ** * ** **
CB85	7	E4	11100100	*** * *** *
CB86	8	F6	11110110	**** ** **** **
CB87	9	FA	11111010	***** * ***** *
CB88	10	FE	11111110	***** *****
CB89	11	FC	11111100	***** *****
CB8A	12	F8	11111000	***** *****
CB8B	13	F0	11110000	**** *****
CB8C	14	EA	11101010	*** * * ** * *
CB8D	(15)	FF	11111111	*****
CB8E	(16)	FF	11111111	*****

Eddig is többször találkoztunk már annak a rutinnak a hívásával, amely a pontok megjelenítéséhez az IY regisztert a vonalkeresztelési módnak megfelelő címre állítja. A ROM-ban most ez a szubrutin következik.

V14 A vonalkeresztelési módnak megfelelő címet IY-ba teszi
CB8F

CB8F D5	FUSH	DE	
CB90 3A4B0B	LD	A,(0B4B)	L-MODE (0...3)
CB93 E603	AND	03	A biztonság kedvéért a beírt értékből csak a b1-b0-t hagyjuk meg
CB95 21A0CB	LD	HL,CBA0	A címtáblázat eleje
CB98 CDAAC9	CALL	C9AA	V2: a cím DE-ben

CB9B D5	PUSH DE	
CB9C FDE1	POP IY	Ez volt a feladat
CB9E D1	POP DE	
CB9F C9	RET	Kész

A rutin munkájához felhasznált táblázat:

ROM-cím	Tartalom	Jelentés
CBA0	CE CA	MODE 0: 0CACEH
CBA2	D3 CA	MODE 1: 0CAD3H
CBA4	CD CA	MODE 2: 0CACDH
CBA6	CF CA	MODE 3: 0CACFH

Ezután következik az egyenesrajzoló programban használt, a pontok léptetését végző segédrutin vezérlő része. Ez a HL pillanatnyi értékének előjele alapján folytatja vagy éppen megváltoztatja az összeadások és kivonások sorozatát, egyúttal intézkedik arról, hogy a két pont közötti eltérések előjele, nagysága és ezek egymáshoz való viszonya alapján történjék a pontok léptetése.

A kétirányú léptetés a főprogramban előállított és megfelelő címeken tárolt ugrócímek felhasználásával történik.

Először közöljük a vezérlő rutint.

V15
CBA8

A kétirányú léptetés vezérlése -- egyenesrajzoló rutinhoz

in: HL'=a lépésirány számolásához tartozó puffer, előjelváltásainál történik az irányváltás
BC'=pozitív növekmény
DE'=a másik irányváltáshoz tartozó negatív tag
HL=aktuális videomemória cím
DE=0040H, sorhossz
C=az aktuális ponthoz tartozó bitek a videomemóriában

out: HL és C az aktuális léptetés után.

CBA8 D9	EXX	
CBA9 CB7C	BIT 7,H	A lépésiránypuffer negatív?
CBAB 2808	JR Z,CB85	Ha nem, ugrás a másik léptető ágra;
CBAD 09	ADD HL,BC	ha még negatív: növeljük
CBAE D9	EXX	

CBAF DD2A840B	LD	IX, (0B84)	Vesszük az adott irányhoz tartozó címet és ugrás a léptető rutinra
CBB3 DDE9	JP	IX	
CBB5 19	ADD	HL, DE	Ha a lépésiránypuffer pozitív: csökkentjük
CBB6 D9	EXX		
CBB7 DD2A860B	LD	IX, (0B86)	Vesszük az ehhez tartozó címet és
CBB8 DDE9	JP	IX	ugrás a megfelelő léptető rutinra

És már is itt vannak a megfelelő belépési pontok. A HL regiszterpár az aktuális videomemória címet, C pedig az aktuális pont bitjeit tartalmazza. Az egyes belépési pontoknál *nnn formában megadjuk annak a funkciónak a rövidítését, amely az adott belépési ponthoz tartozik. A rutinokat követő táblázatban -- egyebek mellett -- ezek pontos jelentését is megadjuk.

CBB0 B7	OR	A	*11x
CBBE ED52	SBC	HL, DE	Egy sorral feljebb
CBC0 CB01	RLC	C	*100, *110 Lépés balra
CBC2 D0	RET	NC	
CBC3 2D	DEC	L	Ha kell: az előző byte
CBC4 C9	RET		és kész
CBC5 CB09	RRC	C	*10x Lépés jobbra
CBC7 3001	JR	NC, CBCA	
CBC9 2C	INC	L	Ha kell: átlép a következő byte-ra
CBCA B7	OR	A	*011, *111
CBCB ED52	SBC	HL, DE	Egy sorral feljebb
CBCD C9	RET		
CBCE CB01	RLC	C	*01x Lépés balra
CBD0 3001	JR	NC, CBD3	
CBD2 2D	DEC	L	Ha kell: az előző byte
CBD3 19	ADD	HL, DE	*001, *101 Lépés le
CBD4 C9	RET		
CBD5 19	ADD	HL, DE	*00x, Lépés lefelé
CBD6 CB09	RRC	C	*000, *010 Lépés jobbra
CBD8 D0	RET	NC	
CBD9 2C	INC	L	Ha kell, átlép a következő byte-ra és kész
CBDA C9	RET		

A különállónak látszó négy szubrutint 12 féle szituációban kezeli a V15 léptetésvezérlő program. A ténylegesen használt két belépési címet az egyenesrajzoló rutin választja ki az összekötendő pontok koordinátái alapján.

A következőkben megadjuk a működés értelmezését megkönnyítő, a rendszer által is használt táblázatot.

ROM- cím	Ugrócím	Szimbólum	A léptetési szituáció
CBDB	CBD5	*00x	Mindkét eltérés pozitív
CBDD	CBCE	*01x	Az x irányú eltérés pozitív az y irányú negatív
CBDF	CBC5	*10x	Az x irányú eltérés negatív az y irányú pozitív
CBE1	CBBD	*11x	Mindkét eltérés negatív
CBE3	CBD6	*000	Mindkét eltérés pozitív és az x irányú eltérés nagyobb
CBE5	CBD3	*001	A két eltérés pozitív és az x irányú eltérés kisebb
CBE7	CBC0	*100	Az x irányú eltérés negatív, az y irányú pozitív és az eltérés x irányban nagyobb
CBE9	CBD3	*101	Az x irányú eltérés negatív, az y irányú pozitív és az eltérés y irányban nagyobb
CBEB	CBD6	*010	Az x irányú eltérés pozitív, az y irányú negatív és az eltérés x irányban nagyobb
CBED	CBCA	*011	Az x irányú eltérés pozitív, az y irányú negatív és az eltérés y irányban nagyobb
CBEF	CBC0	*110	Mindkét eltérés negatív és az x irányban nagyobb
CBF1	CBCA	*111	Mindkét eltérés negatív és az y irányban nagyobb

Ezekután ismét komolyabb dologhoz látunk. A ROM két funkcióhívás rutinjával folytatódik.

VIDEO 08 B-ON (beam on)
 08 Elektron sugár bekapcsolása az adott pontra
 (leteszi a képzeletbeli tollat)

in: -
 out: A=00 rendben

CBF3 CD8FC9	CALL C98F	V1: a lapozások intézése
CBF6 CDD9C9	CALL C9D9	V6.1: előkészíti a pont megjelenítéséhez szükséges paramétereket
CBF9 CDCOCA	CALL CACO	V8.1: az INK-szín és a vonalkeresztelési módot figyelembe véve kigyújtja a pontot
CBFC 3EFF	LD A,FF	A letett toll jelzése
CBFE 26AF	LD H,AF	

Ezzel az utasítással -- amelynek itt semmi funkciója nincs -- a program átmegy a következő funkcióhívás rutinjára. A megfelelő programrész az előző utasítás operandusával, a 0AFH-val indul, amely az A-t nullázza. Itt azonban a CPU nem ezt érezte!

VIDEO 09 B-OFF (beam off)
 09 Az elektron sugár kikapcsolása
 (a képzeletbeli tollat felemeli)

in: -
 out: A=00 rendben

CBFF AF	XOR A	"Felemelt toll" jelzése
CC00 32740B	LD (0B74),A	A toll állapotát tárolja
CC03 AF	XOR A	Hibátlan működés jelzése
CC04 C9	RET	és kész

Ezekután a video eszközmeghajtó sokszor használt, igen fontos, de nem túl komplikált segédrutinjai következnek.

Ezek a beállított üzemmód alapján olyan byte-okat állítanak elő, amelyeket a videomemóriába töltve INK vagy PAPER színű folyamatos vonalat adnak (természetesen az egy byte által képviselt méretben).

Az itt következő rutinok 3 belépési pontban aktivizálhatók. Fontos, hogy mindegyik esetben a vonalbyte-ot meghatározó szín kódját kell megadni, mégpedig 2-es és 4 színű üzemmódban a palettaregiszter sorszámával, 16 színű módban pedig a 0...15 (dec.) színek kódokkal. A szubrutinok ezekből állítják elő a videomemóriába tölthető és ott a meghatározott színű vonalat eredményező értékeket.

V20
CC05 PAPER színű vonal-byte előállítás a beállított üzemmódban
 in: -
 out: A-ban és B-ben az érték

CC05 3A4E0B LD A, (0B4E) PAPER, az alapszín
CC08 CD10CC CALL CC10 V21.1: előállítja A-ban a vonalbyte-ot,
CC0B 47 LD E, A B-be is betölti és
CC0C C9 RET kész

V21
CC0DH INK színű vonalbyte előállítás a beállított üzemmódban
 in: -
 out: A-ban az érték

CC0D 3A4D0B LD A, (0B4D)

A rutin következő sora más helyről (pl.: V20) történő hívása esetén a belépési pont.

V21.1
CC10H Az A regiszterben megadott kódú vonalbyte előállítás a megadott üzemmód szerint
 in: A=feldolgozandó kód
 out: A=a vonalbyte

CC10 2A730B LD HL, (0B73) L-ben az üzemmód: 00..02
CC13 CB3D SRL L 4 színű mód?
CC15 3809 JR C, CC20 Ha igen: ugrás a 4 színű mód feldolgozásához
CC17 CB3D SRL L 16 színű mód?
CC19 3816 JR C, CC31 Ha igen: ugrás a megfelelő programrészhez

Ha a program itt maradt, akkor 2 színű módként kell -- és csak így lehet -- értelmezni.

2 színű mód feldolgozása:

CC1B E601	AND	01	A megadott színekódból most csak a b0 bitet hagyja meg
CC1D 1F	RRA		E két egyszerű utasítással elérjük, hogy minden bit ugyanolyan lesz, mint b0
CC1E 9F	SBC	A,A	
CC1F C9	RET		

4-színű mód feldolgozása:

CC20 E603	AND	03	Most a kód csak a 4 palettaregiszter sorszáma lehet
CC22 1F	RRA		A b0 a carry-be, majd az L regiszter b0 bitjébe
CC23 CB15	RL	L	
CC25 1F	RRA		Az eredeti b1 a carry-be
CC26 9F	SBC	A,A	E művelet után az A minden bitje ugyanaz lesz, mint ami a b1 bit volt
CC27 CB1D	RR	L	Ezután az L b0 bitje visszakerül a carry-be, onnan pedig b7-be, végül a b7-et átírjuk b6-ba, b5-be és b4-be
CC29 1F	RRA		Ezzel elértük, hogy az A felső bitjei az eredeti b0-lal, az alsó bitek pedig b1-gyel egyeznek meg, így kész is
CC2A CB2F	SRA	A	
CC2C CB2F	SRA	A	
CC2E CB2F	SRA	A	
CC30 C9	RET		

16 színű üzemmódban:

CC31 E60F	AND	0F	Csak az alsó 4 bit kell
CC33 2604	LD	H,04	Most a színsorszám bitjei éppen a kapcsolt I-G-R-B állapotot jelentik, így csak minden bitet meg kell kettőzni.
CC35 1F	RRA		Az A-ból kiléptetve ezt hozzuk létre L-ben, majd átírjuk a 2 pont 4 bitjét A-ba és kész.
CC36 CB1D	RR	L	
CC38 CB2D	SRA	L	
CC3A 25	DEC	H	
CC3B 20F8	JR	NZ,CC35	
CC3D 7D	LD	A,L	
CC3E C9	RET		

Ha a kedves olvasó számára ezeknek a bitműveleteknek a célszerűsége nem nyilvánvaló, érdemes azokat egy-egy konkrét próbaértékkel végigjátszani. Az ilyen átalakításoknak magunk is sokszor nagy hasznát vehetjük.

A következő segédrutin érdekes feladatot lát el: az utoljára megcímzett pont tényleges állapotát adja vissza. Használata a felhasználói programokból sokszor nagyon praktikus.

V22 A képernyőn utoljára megcímzett pont állapotának
 CC3F színekódját adja vissza, a beállított üzemmódhoz
 igazodva
 in: - (de a VID-nek belapozva kell lennie)
 out: A=az utolsó pont színekódja

CC3F	CDE2C9	CALL	C9E2	V6 végét hívja: C-be beolvassa az utolsó pont bitjeit, HL-be a videomemória címét
CC42	3A730B	LD	A, (0B73)	
CC45	47	LD	B, A	Az üzemmód
CC46	79	LD	A, C	A vizsgált pont bitjei,
CC47	A6	AND	(HL)	majd A-ban a pont tényleges állapota
CC48	CB38	SRL	B	4 színű mód?
CC4A	380A	JR	C, CC56	Ha igen: ugrás előre
CC4C	CB38	SRL	B	16-os üzemmód?
CC4E	3815	JR	C, CC65	Ha igen: ugrás a megfelelő címre

A színekód előállítás 2-es üzemmódban:

CC50	C6FF	ADD	A, FF	Ebben az a fontos, hogy a carry felvette a vizsgált bit értékét, bármelyik is volt az!
CC52	17	RLA		Így azt a b0-ba léptetve és
CC53	E601	AND	01	csak ezt hagyva meg:
CC55	C9	RET		kész!

4-es üzemmódban:

CC56	C6F0	ADD	A, F0	Most a carry azt jelzi, hogy a felső 4 bitben volt-e 1-es,
CC58	CB11	RL	C	ezt átmenetileg a C 0. bitjébe tesszük
CC5A	E60F	AND	0F	Majd az alsó 4 bit kell
CC5C	C6FF	ADD	A, FF	A carry most is jelzi, hogy volt-e köztük: 1
CC5E	17	RLA		Ezt, majd a felső bitekre jellemző adatot is az A b1-b0 bitjébe visszük,
CC5F	CB19	RR	C	
CC61	17	RLA		

CC62 E603	AND 03	a többbit töröljük, s ezzel
CC64 C9	RET	kész a kód

Végül a 16 színmód feldolgozása így zajlik:

CC65 C6C0	ADD A,CO	A felső 2 bitet nézzük, s ha
CC67 17	RLA	van köztük 1, akkor a carry-n
CC68 E67F	AND 7F	át b0-ba kerül
		A b7=0 legyen!
CC6A C6E0	ADD A,E0	A következő 2 bit: ez
CC6C 17	RLA	eredményt A léptetésével a
		carry-ből A-ba, a korábban
		bevitt bit mellé mozgatjuk
CC6D E63F	AND 3F	A feldolgozott biteket ismét
		töröljük
CC6F C6F0	ADD A,FO	Ezt az eljárást megismételjük
CC71 17	RLA	még kétszer, így végül az A
CC72 E61F	AND 1F	alsó bitje a vizsgált pont
CC74 C6F8	ADD A,F8	állapotát mutatja, ami éppen
CC76 17	RLA	a keresett kód
CC77 E60F	AND 0F	
CC79 C9	RET	

Ennyire egyszerű!

Most egy olyan segédrutin következik, amelyet a B-REL funkcióhívás használ: a megadott relatív elmozdulásból kiszámítja a megcímezett pont abszolút koordinátáit.

V24	Relatív elmozdulásból az abszolút koordináták
CC7A	előállítása
in:	BC=az x irányú relatív elmozdulás
	DE=az y irányú relatív elmozdulás
out:	BC=az új abszolút x koordináta
	DE=az új abszolút y koordináta

CC7A 2A7C0B	LD HL,(0B7C)	A korábbi x koordináta,
CC7D 09	ADD HL,BC	+ az elmozdulás,
CC7E 44	LD B,H	
CC7F 4D	LD C,L	BC-ben az eredmény
CC80 2A7E0B	LD HL,(0B7E)	A korábbi y koordináta,
CC83 19	ADD HL,DE	+ az elmozdulás,
CC84 EB	EX DE,HL	DE-ben az eredmény és
CC85 C9	RET	kész

Ismét alapvető, a karakterkiírási funkcióhívásokat kiszolgáló rutinok következnek.

Először a több egymás utáni karakter megjelenítését végző funkcióhívás rutinja áll, amely az egy karaktert kiíró rutin ismételt meghívásával látja el feladatát.

```
VIDEO 02          VID-BKOUT (video block-output)
02                Szövegkiírás a képernyőre
                  in: DE=a kiírandó szöveg kezdőcíme a memóriában
                    (ASCII kódú karakterek)
                    BC=a szöveg hossza (karakterek száma)
                  out: A=00 kész

CC86 EB          EX    DE,HL          HL=memóriacím
CC87 E5          PUSH HL
CC88 C5          PUSH BC
CC89 4E          LD    C,(HL)        A kiírandó karakter
CC8A CD94CC      CALL CC94          VIDEO 01, a karakter kiírása

CC8D C1          POP  BC
CC8E E1          POP  HL
CC8F EDA1        CPI                    HL+1 és BC-1
CC91 E0          RET  PD              Van még?
                                      Ha nincs több: visszatér
CC92 18F3        JR   CC87          Egyébként folytatja
```

Ennél több dolgunk lesz -- teljesen nyilvánvaló okokból -- az egyes karakterek kiírását végző rutinnal.

```
VIDEO 01          VID-CHOUT (video character-output)
01                Egy karakter kiírása a képernyőre
                  in: C=a kiírandó karakter kódja
                  out: A=00 nincs hiba
```

Eszerint tehát a rutin minden kód kezelésére vállalkozik, legfeljebb nem csinál vele semmit. Valójában az összes nyomtatható karaktert (20H...DFH ASCII kódok), valamint a 0DH és 0AH vezérlőkódokat veszi figyelembe.

Természetesen az editor által használt vezérlő funkciókat -- pl. törlés a bekezdés végéig -- ez a megjelenítő rutin nem hajtja végre. Az ehhez hasonló szolgáltatásokat a speciális programok együttese által képviselt, logikai eszközként nyilvántartott editor oldja meg.

A video karaktermegjelenítő rutinja csak a kódjaival meghatározott karakterek megjelenítésére szolgál.

Nézzük!

CC94 CD8FC9	CALL C98F	V1: az ismert lapozási eljárás
CC97 79	LD A,C	Nézzük a karaktert!
CC98 FE20	CP Z0	Ha 20H vagy nagyobb:
CC9A 300C	JR NC,CCA8	ugrás előre
CC9C FE0A	CP 0A	0AH (soremelés)?
CC9E CA31CD	JP Z,CD31	Ha igen: ugrás az ezt végrehajtó V26-hoz
CCA1 FE0D	CP 0D	0D (return)?
CCA3 CA2BCD	JP Z,CD2B	Ha igen: V25, a return végrehajtása
CCA6 AF	XOR A	A többi 20H alatti kóddal nem csinál semmit
CCA7 C9	RET	
CCA8 FEE0	CP E0	A kód E0H-nál kisebb?
CCAA 3E00	LD A,00	"Nincs hiba" jelzése, de most XOR A nem lehet, mert kell a jelzőbit!
CCAC D0	RET NC	Ha a kód 0E0H, vagy nagyobb: nem csinál semmit

Ezután már biztos, hogy a C regiszterben kiírható karakter kódja van.

CCAD D9	EXX	Komoly előkészítéseket végzünk
CCAE 11DCFF	LD DE,FFDC	Ez -24H fizikai y koordináta-csökkentést jelent, aminek a valóságos értelme: 9 sorral lejjebb!
CCB1 3A730B	LD A,(0B73)	üzemmód
CCB4 47	LD B,A	Kiszámítjuk, hogy ebben az esetben egy logikai pont hány fizikai pontból áll
CCB5 3E01	LD A,01	
CCB7 04	INC B	
CCB8 87	ADD A,A	
CCB9 10FD	DJNZ CCB8	A kapott értéket el is tároljuk
CCBB 32820E	LD (0B82),A	Mivel egy karaktert 8 logikai ponton jelenítünk meg, így a karakter jobb széle 7-szer ennyivel kerül jobbra
CCBE 6F	LD L,A	
CCBF 87	ADD A,A	
CCC0 85	ADD A,L	
CCC1 87	ADD A,A	
CCC2 85	ADD A,L	
CCC3 4F	LD C,A	BC=a relatív x elmozdulás a karakter jobb széléig
CCC4 AF	XOR A	
CCC5 CD7ACC	CALL CC7A	V24: a kiírandó karakter jobb alsó pontjának abszolút koordinátái
CCC8 CDBAC9	CALL C9BA	V4: az y ellenőrzése, ha az alsó sor alá menne: visszatér
CCCB D8	RET C	

CCCC CDB3C9	CALL C9B3	V3: az x ellenőrzése
CCCF 3006	JR NC,CCD7	Ha még van elég hely ebben a sorban: átugorja a következő 2 sort
CCD1 CD2BCD	CALL CD2B	V25: return
CCD4 CD31CD	CALL CD31	V26: soremelés

Most tehát már van helye a képernyőn a karakternek.

CCD7 CDD0C9	CALL C9D0	V6 szerint elkészíti a kiíráshoz szükséges adatokat, vonaltípus nincs!
-------------	-----------	--

A kiírás az utoljára megcímezett pontban kezdődik.

CCDA D9	EXX	
CCDB 1174C4	LD DE,C474	Az ábrázolható karakterek definícióit tartalmazó táblázat elé mutat 0140H-val, ez éppen a vezérlőkaraktereknek megfelelő 32*10=320 byte (dec.) eltolás
CCDE CB79	BIT 7,C	A kód 127-nél nagyobb?
CCE0 2803	JR Z,CCE5	ha igen, akkor definiált karakterről van szó, ennek a táblázata kell
CCE2 114007	LD DE,0740	
CCE5 CBB9	RES 7,C	Most már csak az eltolás értékét kell kiszámolni
CCE7 0600	LD B,00	
CCE9 60	LD H,B	
CCEA 69	LD L,C	Egy-egy kódnak 10-10 byte felel meg, ezért a kódot szorozni kell 10-zel
CCEB 29	ADD HL,HL	
CCEC 29	ADD HL,HL	
CCED 09	ADD HL,BC	
CCEE 29	ADD HL,HL	
CCEF 19	ADD HL,DE	Ezt a táblázat elejéhez adva, eljutunk a kért karakter 10 byte-jához
CCF0 060A	LD B,0A	
CCF2 3A500B	LD A,(0B50)	10-es ciklust szervezünk V-FLAG, karakterfelülírási mód (00...03)
CCF5 5F	LD E,A	Tároljuk
CCF6 2B	DEC HL	Még mindig előkészítés

Nos, ennyi minden kellett ahhoz, hogy eljussunk egy karakter tényleges kiírásának megkezdéséhez.

A másodlagos készletben HL és C adminisztrálja a videomemóriát -- és vezérli a tényleges megjelenítést - az aktuális regiszterkészlet pedig a karakter meghatározását jelentő byte-okat kezeli.

Kövessük tovább a működést!

CCF7 23	INC HL	
CCF8 4E	LD C, (HL)	A karakter következő sora
CCF9 D9	EXX	Egy sorban 8 pont van
CCFA 0608	LD B, 08	Először átlépi a következő
CCFC 1803	JR CD01	utasítást
CCFE CDD6CB	CALL CBD6	A V15, egyenesrajzoló segédrutin egyik ágát hívja: a pont léptetése jobbra
CD01 D9	EXX	
CD02 CB11	RL C	A sor következő pontja (INK vagy PAPER színű)
CD04 7B	LD A, E	A felülírási mód
CD05 D9	EXX	
CD06 CDB7CA	CALL CAB7	V7: minden feltételt figyelve kiteszi a pontot, ill. 8 pontból egy sort
CD09 10F3	DJNZ CCFE	
CD0B CBD3CB	CALL CBD3	Ez is a V15 egy része: a videomemóriában egy sorral lejjebb, utána pedig 7-szer léptetés balra, tehát a karakter következő sorának kezdőpontjába
CD0E 0607	LD B, 07	
CD10 CDC0CB	CALL CBC0	
CD13 10FB	DJNZ CD10	
CD15 D9	EXX	
CD16 10DF	DJNZ CCF7	Ezt ismételjük 10 soron át és kész!

Még át kell azonban állítani a következő kiíráshoz az aktuális paramétereket.

CD18 3A820B	LD A, (0B82)	Egy logikai pont mérete az aktuális üzemmódban
CD1B 87	ADD A, A	A 8 pontból álló karakter
CD1C 87	ADD A, A	szélessége ennek 8-szorosa
CD1D 87	ADD A, A	
CD1E 4F	LD C, A	
CD1F CD7ACC	CALL CC7A	V24: a következő kezdőpont abszolút koordinátája
CD22 CDB3C9	CALL C9B3	V3: még elfér a sorban?
CD25 3007	JR NC, CD2E	Ha igen: ugrás előre
CD27 0131CB	LD BC, CD31	Ha nem: a V31-ből saját szubrutint csinálunk úgy, hogy egy RET címet teszünk a verembe
CD2A C5	PUSH BC	

Ezután a rutin átcsorog az önállóan is használható kocsvissza (return) és soremelés (LF, line feed) szubrutinokra. Ez a PUSH BC mindenesetre elérte, hogy a return után a vezérlés a soremelés rutinra adódik vissza.

V25 Return rutin. Az aktuális kiírási pozíciót a sor
CD2B elejére állítja

CD2B 010000	LD	BC,0000	Fizikai x koordináta
CD2E C369CA	JP	CA69	V31: beállítja a pont megjelenítéséhez szükséges paramétereket és RET

V26 Soremelés rutin. Az aktuális kiírási pozíciót egy
CD31 karaktersorral (dec. 10 tv-sor) lejjebb állítja.
A képernyő alján nem határos

CD31 11B4FF	LD	DE,FFB4	Ez -4CH, tehát megnézi, hogy 19 tv-sorral lehet-e még lejjebb menni. Ez lenne a következő karaktersor utolsó tv-sora
CD34 CD7ACC	CALL	CC7A	
CD37 AF	XOR	A	
CD38 CDBAC9	CALL	C9BA	
CD3B D8	RET	C	Ha nem lehet: visszatér
CD3C 11D8FF	LD	DE,FFD8	-28H, azaz egy karaktersorral lejjebb
CD3F 010000	LD	BC,0000	A vízszintes pozíció nem változik
CD42 CD7ACC	CALL	CC7A	V24: kiszámítja az abszolút koordinátákat,
CD45 C365CA	JP	CA65	V31: előkészíti a pont megjelenítéséhez szükséges paramétereket és RET

A ROM-ban ezután a zárt alakzatok belsejét kifestő funkcióhívás rutinja következik.

A program maga nem túlságosan bonyolult, azonban teljes értelmezése meglehetősen hosszadalmas lenne. Ezzel a rutinnal méginkább úgy állunk, mint korábban az egyenesrajzolóval: kifejezetten felhasználói probléma, olyan szolgáltatás, amely nem tartozik a számítógép működésének legalapvetőbb funkciói közé.

Ha az Olvasót ez a rutin mégis nagyon érdekli, akkor az eddig már megadott, fontos segédrutinok alapján maga is sikerrel utána nézhet a működésnek.

Természetesen közöljük a funkciót megvalósító rutin teljes utasításlistáját, azt olyan funkcionális blokkokra is felbontva, amely az Olvasó saját, részletes visszafejtési próbálkozásait igyekszik megkönnyíteni.

Egy-két fontosabb részre pedig külön is felhívjuk a figyelmet.

VIDEO 10	PAINT	
0AH	Zárt alakzatok belsejének kifestése	
	in: -	
	out: -	
CD48 CD8FC9	CALL C98F	V1: lapozások kezelése
CD4B ED73880B	LD (0B88),SP	
CD4F FD21CFCA	LD IY,CACF	MODE 3
CD53 CD3FCC	CALL CC3F	V22: a megcímzett kezdőpont színkódja
CD56 CD10CC	CALL CC10	V21.1: a kezdőpont színű vonalbyte
CD59 328A0B	LD (0B8A),A	Elteszi
CD5C 5F	LD E,A	
CD5D CD0DCC	CALL CC0D	V21: INK vonalbyte
CD60 AB	XOR E	Ugyanaz?
CD61 210000	LD HL,0000	
CD64 C8	RET Z	Ha igen: nincs munka!
CD65 22910B	LD (0B91),HL	A kifestett pontok nyilvántartása
CD68 E5	PUSH HL	
CD69 2A7C0B	LD HL,(0B7C)	Fizikai x koordináta
CD6C 0604	LD B,04	
CD6E CDC6C9	CALL C9C6	V5.1: a soron belüli byte-pozíció
CD71 E5	PUSH HL	
CD72 CDDFC9	CALL C9DF	V6 része, rááll a kezdőpontra
CD75 3E40	LD A,40	Sorhossz.
CD77 D1	POP DE	Byte-pozíció
CD78 93	SUB E	Távolság a jobb oldali sorvégtől
CD79 47	LD B,A	
CD7A 228D0B	LD (0B8D),HL	Az aktuális VID cím
CD7D ED438B0B	LD (0B8B),BC	C=pontbitek, B=távolság jobbra
CD81 0601	LD B,01	
CD83 181E	JR CDA8	

Ezután kezdődik a tényleges munka.

CD85 E1	POP	HL	A letett pontok száma
CD86 7C	LD	A, H	Ha már nem lehetett új
CD87 B5	OR	L	pontot letenni:
CD88 C8	RET	Z	kész
CD89 C1	POP	BC	Távolság és pontbitek
CD8A 22910B	LD	(0B91), HL	
CD8D ED438B0B	LD	(0B8B), BC	
CD91 E1	POP	HL	Az aktuális határcím
CD92 114000	LD	DE, 0040	
CD95 19	ADD	HL, DE	Egy sorral lejjebb
CD96 228D0B	LD	(0B8D), HL	Az új kezdőcím
CD99 0602	LD	B, 02	
CD9B 1100BC	LD	DE, BC00	
CD9E ED52	SBC	HL, DE	Ha már a képernyő alá
CDA0 D263CE	JP	NC, CE63	menne: ugrás előre
CDA3 D9	EXX		
CDA4 CD92CE	CALL	CE92	Verem-ellenőrzés
CDA7 38DC	JR	C, CD85	Veszélyes helyzetben
			ugrás vissza
CDA9 2A910B	LD	HL, (0B91)	Lerakott pontok és
CDAC 7C	LD	A, H	jellemzők
CDAD CBBC	RES	7, H	A jellemzők törlése
CDAF CBB4	RES	6, H	után:
CDB1 228F0B	LD	(0B8F), HL	tárolás
CDB4 EB	EX	DE, HL	
CDB5 CB77	BIT	6, A	
CDB7 2809	JR	Z, CDC2	
CDB9 07	RLCA		
CDBA D9	EXX		
CDBB A8	XOR	B	
CDBC D9	EXX		
CDBD E601	AND	01	
CDBF C262CE	JP	NZ, CE62	
CDC2 2A8D0B	LD	HL, (0B8D)	VID-cím
CDC5 ED4B8B0B	LD	BC, (0B8B)	Távolság és pontbitek
CDC9 3A8A0B	LD	A, (0B8A)	Kezdő vonalbyte
CDCC AE	XOR	(HL)	Ha az aktuális pont más
CDCD A1	AND	C	színű:
CDCE 205F	JR	NZ, CE2F	ugrás előre
CDD0 3E40	LD	A, 40	
CDD2 90	SUB	B	
CDD3 3C	INC	A	A távolság a sor elejé-
CDD4 47	LD	B, A	től, byte-okban
CDD5 110000	LD	DE, 0000	Pontszámláló
CDD8 180B	JR	CDE5	
CDBA 3A8A0B	LD	A, (0B8A)	A kezdőponttól a soron
CDDD AE	XOR	(HL)	belül balra haladva
CDDE A1	AND	C	vonalat húz, ameddig egy
CDDF 200B	JR	NZ, CDEC	határvonalhoz nem ér

CDE1 CDCOCA	CALL CACO	
CDE4 13	INC DE	A ciklust elhagyva B-ben
CDE5 CB01	RLC C	a maradék távolság,
CDE7 30F1	JR NC,CDDA	C-ben az utolsó pontbit,
CDE9 2D	DEC L	DE=a letett pontok száma
CDEA 10EE	DJNZ CDDA	és HL lesz az utolsó VID cím
CDEC 3E40	LD A,40	
CDEE 90	SUB B	
CDEF 3C	INC A	A sor végétől való távolság
CDF0 47	LD B,A	
CDF1 CB09	RRC C	Egy ponttal jobbra lép,
CDF3 3002	JR NC,CDF7	
CDF5 2C	INC L	ha ehhez új byte kell,
CDF6 05	DEC B	adminisztrálja is
CDF7 E5	PUSH HL	VID-cím
CDF8 C5	PUSH BC	Távolság és pontbitek
CDF9 D5	PUSH DE	Pontszám
C DFA 2A8D0B	LD HL,(0B8D)	Aktuális VID kezdőcím
C DFD ED4B8B0B	LD BC,(0B8B)	Távolság és pontbitek
CE01 CD7ACE	CALL CE7A	Jobbra húz vonalat egy határpontig
CE04 E3	EX (SP),HL	
CE05 7C	LD A,H	Összeadja a pontszámokat és
CE06 B5	OR L	ha van baloldali pont is,
CE07 2802	JR Z,CE0B	akkor a H b7 bitjét 1-be állítja
CE09 CBFC	SET 7,H	
CE0B 19	ADD HL,DE	
CE0C E3	EX (SP),HL	
CE0D E5	PUSH HL	Az utolsó VID-cím
CE0E 2A8F0B	LD HL,(0B8F)	Korábbi pontszám
CE11 B7	OR A	
CE12 ED52	SBC HL,DE	
CE14 D1	POP DE	
CE15 E3	EX (SP),HL	Összes pontszám
CE16 F5	PUSH AF	A jelzőbitek tárolása
CE17 3E80	LD A,80	
CE19 CB7C	BIT 7,H	Volt baloldali pont is?
CE1B 2009	JR NZ,CE26	Ha igen: ugrás
CE1D 3809	JR C,CE28	Ez még a korábbi vizsgálatra vonatkozik
CE1F D9	EXX	
CE20 CB08	RRC B	A B 7. bitje a carry-be kerül,
CE22 CB00	RLC B	
CE24 D9	EXX	
CE25 1F	RRA	onnan pedig A-ban b7-be
CE26 AC	XOR H	
CE27 67	LD H,A	Ez lesz az érvényes
CE28 F1	POP AF	A korábbi jelzőbitek

CE29 E3	EX	(SP),HL	
CE2A 3836	JR	C,CE62	
CE2C 2834	JR	Z,CE62	
CE2E EB	EX	DE,HL	
CE2F CD92CE	CALL	CE92	Verem-ellenőrzés
CE32 DA85CD	JP	C,CD85	Veszélyes esetben ugrás vissza
CE35 7B	LD	A,E	
CE36 B7	OR	A	E=00?
CE37 2001	JR	NZ,CE3A	Ha nem: ugrás előre
CE39 15	DEC	D	
CE3A 3A8A0B	LD	A,(0B8A)	Kezdőpont színű vonalbyte
CE3D AE	XOR	(HL)	
CE3E A1	AND	C	Ha az aktuális pont ilyen: ugrás előre
CE3F 280B	JR	Z,CE4C	
CE41 1D	DEC	E	
CE42 281A	JR	Z,CE5E	
CE44 CB09	RRC	C	Jobbra lép és folytatja a vizsgálatot
CE46 30F2	JR	NC,CE3A	
CE48 2C	INC	L	
CE49 05	DEC	B	
CE4A 18EE	JR	CE3A	
CE4C 7B	LD	A,E	Az aktuális pont olyan, mint az első
CE4D B7	OR	A	
CE4E 2001	JR	NZ,CE51	
CE50 14	INC	D	
CE51 ED538F0B	LD	(0B8F),DE	
CE55 E5	PUSH	HL	
CE56 E3	EX	(SP),HL	
CE57 C5	PUSH	BC	
CE58 CD7ACE	CALL	CE7A	Az aktuális ponttól vonalat húz,
CE5B D5	PUSH	DE	
CE5C 18AF	JR	CE0D	és ugrás vissza
CE5E 15	DEC	D	
CE5F F244CE	JP	P,CE44	
CE62 D9	EXX		
CE63 05	DEC	B	Van még?
CE64 CA85CD	JP	Z,CD85	Ha nincs, ugrás vissza
CE67 D9	EXX		
CE68 2A8D0B	LD	HL,(0B8D)	
CE6B 1180FF	LD	DE,FF80	2 sorral feljebb
CE6E 19	ADD	HL,DE	
CE6F 228D0B	LD	(0B8D),HL	
CE72 CB7C	BIT	7,H	Ha nem lehet: ugrás vissza, egyébként
CE74 CA85CD	JP	Z,CD85	folytatja
CE77 C3A4CD	JP	CDA4	

```

CE7A 110000 LD DE,0000
CE7D 3A8A0B LD A,(0B8A)
CE80 AE XOR (HL)
CE81 A1 AND C
CE82 C0 RET NZ
CE83 13 INC DE
CE84 3A940B LD A,(0B94)
CE87 A1 AND C
CE88 AE XOR (HL)
CE89 77 LD (HL),A
CE8A CB09 RRC C
CE8C 30EF JR NC,CE7D
CE8E 2C INC L
CE8F 10EC DJNZ CE7D
CE91 C9 RET

```

Az aktuális ponttól vonalat húz jobbra, ameddig egy határvonalat nem talál. A ciklusból kilépő adatok megegyeznek a balra haladó vonalat húzó programrésznél leírtakkal

```

CE92 E5 PUSH HL
CE93 D5 PUSH DE
CE94 21EAF7 LD HL,FFEA
CE97 39 ADD HL,SP
CE98 ED5B170B LD DE,(0B17)
CE9C ED52 SBC HL,DE
CE9E D2C9CE JP NC,CEC9

CEA1 C5 PUSH BC
CEA2 D9 EXX
CEA3 C5 PUSH BC
CEA4 2A880B LD HL,(0B88)
CEA7 2B DEC HL
CEA8 2B DEC HL
CEA9 2B DEC HL
CEAA 54 LD D,H
CEAB 5D LD E,L
CEAC 4D LD C,L
CEAD 44 LD B,H
CEAE 210900 LD HL,0009
CEB1 39 ADD HL,SP
CEB2 B7 OR A
CEB3 ED42 SBC HL,BC
CEB5 60 LD H,B
CEB6 69 LD L,C
CEB7 2013 JR NZ,CECC
CEB9 010A00 LD BC,000A
CEBC EDB8 LDDR
CEBE EB EX DE,HL
CEBF F9 LD SP,HL
CEC0 33 INC SP
CEC1 B7 OR A
CEC2 EB EX DE,HL
CEC3 ED52 SBC HL,DE
CEC5 3F CCF

```

Verem-ellenőrzés

CEC6 C1	POP	BC
CEC7 D9	EXX	
CEC8 C1	POP	BC
CEC9 D1	POP	DE
CECA E1	POP	HL
CECB C9	RET	

Még mindig a PAINT rutin részeként, a megengedhetőnél jobban elmélyült verem manipulálásával foglalkozik a következő három segédrutin.

CECC CDFECE	CALL	CEFE
CECF 09	ADD	HL, BC
CED0 0100BC	LD	BC, BCOO
CED3 ED42	SBC	HL, BC
CED5 3006	JR	NC, CEDD
CED7 09	ADD	HL, BC
CED8 CD14CF	CALL	CF14
CEDB 280F	JR	Z, CEEC
CEDD CDFDCE	CALL	CEFD
CEE0 B7	OR	A
CEE1 ED42	SBC	HL, BC
CEE3 010080	LD	BC, 8000
CEE6 ED42	SBC	HL, BC
CEE8 09	ADD	HL, BC
CEE9 D414CF	CALL	NC, CF14
CEEC D9	EXX	
CEED 010600	LD	BC, 0006
CEF0 2005	JR	NZ, CEF7
CEF2 EDB8	LDDR	
CEF4 C3ACCE	JP	CEAC

CEF7 B7	OR	A
CEFS ED42	SBC	HL, BC
CEFA C3ACCE	JP	CEAC

CEFD D9	EXX	
CEFE E5	PUSH	HL
CEFF D9	EXX	
CF00 E1	POP	HL
CF01 56	LD	D, (HL)
CF02 2B	DEC	HL
CF03 5E	LD	E, (HL)
CF04 2B	DEC	HL
CF05 2B	DEC	HL
CF06 46	LD	B, (HL)
CF07 2B	DEC	HL


```

CF08 7E      LD   A, (HL)
CF09 E63F    AND  3F
CF0B 2B      DEC  HL
CF0C 6E      LD   L, (HL)
CF0D 67      LD   H, A
CF0E 78      LD   A, B
CF0F 014000  LD   BC, 0040
CF12 EB      EX  DE, HL
CF13 C9      RET

CF14 4F      LD   C, A
CF15 3A8A0B  LD   A, (0B8A)
CF18 47      LD   B, A
CF19 78      LD   A, B
CF1A AE      XOR  (HL)
CF1B A1      AND  C
CF1C C8      RET  Z
CF1D 1D      DEC  E
CF1E 2807    JR   Z, CF27
CF20 CB09    RRC  C
CF22 30F5    JR   NC, CF19
CF24 23      INC  HL
CF25 18F2    JR   CF19
CF27 15      DEC  D
CF28 F220CF  JP   P, CF20
CF2B C9      RET

```

Ezek a programrészek tartoztak tehát az alakzatok kifestésének kiszolgálásához. Nagyon bízom abban, hogy az ezek iránt különösen érdeklődő Olvasó kis fáradozással maga is túl tudja tenni magát a program megfejtésének nehézségein.

Folytatjuk a videorutinok feldolgozását, mégpedig a 11-es funkcióhívás megismerésével.

```

VIDEO 11      CH-DEF
0BH          Felhasználói karakterek definiálása
            in: DE=a karaktert definiáló 10 byte memóriacíme
              C=a karakterhez rendelt ASCII kód (80H...0DFH)
            out: A=00 nincs hiba
              F8 a kód nem a megengedett tartományban van

CF2C 79      LD   A, C          A megadott ASCII kód

```

CF2D FEE0	CP	EO	0DFH-nál nagyobb?
CF2F 3001	JR	NC,CF32	Ha igen: nem lehet!
CF31 B7	OR	A	80H-nál kisebb?
CF32 3EF8	LD	A,F8	Hibakód
CF34 F0	RET	F	Ha igen: az is hiba!

Ha a kód megfelelő, akkor -- mint tudjuk -- a karakter új képéhez nem kell mást tenni, mint a meghatározó 10 byte-ot bemásolni az U0 RAM felhasználói karakterek mátrixát tartalmazó, a megadott kódnak pontosan megfelelő területre. Ez történik a következő sorokban.

CF35 CBB9	RES	7,C	A kódból most már csak a 80H feletti sorszám kell
CF37 0600	LD	B,00	
CF39 69	LD	L,C	Mivel egy kód 10 byte-ot foglal el, ezért a sorszámot is 10-
CF3A 60	LD	H,B	zel kell szorozni ahhoz, hogy
CF3B 29	ADD	HL,HL	a táblázat elejéről a kódhoz
CF3C 29	ADD	HL,HL	tartozó címhez jussunk
CF3D 09	ADD	HL,BC	
CF3E 29	ADD	HL,HL	
CF3F 014007	LD	BC,0740	Ez a táblázat eleje, így
CF42 09	ADD	HL,BC	HL=a kódhoz tartozó mátrix
			kezdőcíme
CF43 EB	EX	DE,HL	A DE mutatta területről ide
CF44 010A00	LD	BC,000A	másoljuk a megadott 10 byte-
CF47 EDB0	LDIR		ot,
CF49 AF	XOR	A	"hibátlan" jelzés és
CF4A C9	RET		kész

Végül még egy hosszadalmasabb programmal, a karakterpozicionálásra vonatkozó funkcióhívás rutinjával ismerkedünk meg. Ez a szubrutin a képernyőt 24 fix karaktersorra osztja és híváskor az üzemmódnak is megfelelő valamelyik kezdő karakterpozícióra áll rá.

VIDEO 03	CH-POS	
03		Normál kiírási karakterpozíció beállítása
in:	C=a sor száma (1...24 dec.)	
	B=a soron belüli pozíció száma (1...16, 1...32	
	vagy 1...64 az üzemmódtól függően)	
out:	A=00 nincs hiba	
	F9 lehetetlen pozíció	

CF4B C5	PUSH	BC	A koordinátákat tárolja
CF4C 210000	LD	HL,0000	Fizikai pozíció kiinduló
			értéke

CF4F 05	DEC	B	Áttér a pozíció 0-val kezdődő
CF50 FA54CF	JP	M,CF54	számolására, de 0-nál kisebbet
CF53 68	LD	L,B	nem enged meg

Tehát ha B eredetileg 0 vagy negatív (!): L=00 lesz.

CF54 3A730B	LD	A,(0B73)	Üzem mód (00...02)
CF57 C604	ADD	A,04	Látszik, hogy a megadott
CF59 47	LD	B,A	pozíciót az üzemmódtól függően
CF5A 29	ADD	HL,HL	16-ta, 32-vel, 64-gyel
CF5B 10FD	DJNZ	CF5A	szorozzuk

Mivel egy karakter szélessége a képernyőn 8 logikai pont, így a kezdőpontot a sorszám 8-cal való szorzásával kapjuk. Mivel azonban egy logikai pontot a 2-es, 4-es és 16-os üzemmódban 2, 4, ill. 8 fizikai pont tesz ki, így a megadott karakterpozíció kezdőpontjának abszolút fizikai koordinátája valóban az előbbi szorzásokkal adódik.

CF5D E5	PUSH	HL	A megcímzett hely kezdőpontjának fizikai koordinátája
CF5E 11D8FF	LD	DE,FFD8	-28H (-40 dec.), ez éppen egy karakter 10 sorához tartozó fizikai y koordináta
CF61 21BF03	LD	HL,03BF	A képernyő felső sorának abszolút y koordinátája
CF64 41	LD	B,C	Függőlegesen is 0-tól kezdjük
CF65 05	DEC	B	a számlálást
CF66 FA6ECF	JP	M,CF6E	Ha pozíció 0, vagy negatív
CF69 2803	JR	Z,CF6E	lett, maradunk a felső sorban
CF6B 19	ADD	HL,DE	Egyébként y-t 40-enként
CF6C 10FD	DJNZ	CF6B	csökkentve kapjuk a megadott
CF6E EB	EX	DE,HL	karaktorsor kezdősorának fizikai y koordinátáját
CF6F C1	POP	BC	Fizikai x koordináta
CF70 D9	EXX		A másodlagos készletben
CF71 ED4B7C0B	LD	BC,(0B7C)	betölti a korábban címzett
CF75 ED5E7E0B	LD	DE,(0B7E)	pont fizikai koordinátáit
CF79 D9	EXX		
CF7A E1	POP	HL	Az eredetileg BC-ben megadott koordináták
CF7B 7D	LD	A,L	A karakter sorszáma
CF7C B7	OR	A	0?
CF7D 2004	JR	NZ,CF83	Ha nem: átlépi a következő 4
CF7F D9	EXX		sort

CF80 D5	PUSH DE	Ha 0-t adtak meg, akkor a régi
CF81 D9	EXX	y koordináta lesz ezután is
CF82 D1	POP DE	érvényes
CF83 7C	LD A,H	A megadott oszlopszám
CF84 B7	OR A	0?
CF85 2004	JR NZ,CF8B	Ha nem: ugrás előre
CF87 D9	EXX	
CF88 C5	PUSH BC	Ha a megadott oszlopszám 0,
CF89 D9	EXX	akkor a régi x lesz ezután is
CF8A C1	POP BC	érvényes
CF8B 3EF9	LD A,F9	Hibakód
CF8D CDB3C9	CALL C9B3	Ha az így kiszámolt
CF90 D8	RET C	koordináták kimutatnak a
CF91 CDBAC9	CALL C9BA	képernyőről: hiba
CF94 D8	RET C	Egyébként beiktatja az új
CF95 C365CA	JP CA65	fizikai pozíciót

És ezzel, kedves Olvasó, végére is értünk a képmegjelenítéssel kapcsolatos nem csekély jelentőségű, bonyolultságú és terjedelmű programrészeknek.

A video eszközmeghajtó rutinjainak bevezetéseként elmondtam, hogy ezek egyben a jelen fejezet legnagyobb összefüggő részét teszik ki.

Mivel mindezen most túl vagyunk, talán hasznos és megnyugtató érzés arra gondolni, hogy ama bizonyos gombnyomásig az eddigieknél már valamelyest kisebb fajsúlyú dolgok köthetik majd le figyelmünket!

Végül, mielőtt tovább lépnénk, érdemes még egy pillantást vetnünk az utolsó ROM-címekre: igen rövid idő múlva átlépjük a Cxxx címek tartományát! Ami azért mégis csak azt jelenti, hogy túl vagyunk a ROM teljes programjának első ötöd részén.

Lássuk, mi van ezután!

3.5.4 Az editor

Az editor a TV-Computer nagyon fontos funkciót betöltő logikai eszköze. Segítségével a képernyő teljes -- hasznos -- területén végezhetünk szövegszerkesztési műveleteket. A BASIC programok bevitelénél és javításánál számtalan praktikus szolgáltatással áll a felhasználó

rendelkezésre. Bár igazi szövegszerkesztőként közvetlenül csak korlátozottan használható, csak az emberi leleményességen múlik, hogy az alapfunkciókat hogyan bővítjük.

3.5.4.1 Az editor ugrótábla

A ROM editor funkciókat megvalósító része is a funkcióhívások ugrótáblájával indul.

ROM	Tartalom	Jelentés
CF98	05	A hívható editor funkciók száma
CF99	CFA3	0. funkció (ED-INT)
CF9B	D052	1. funkció (ED-CHIN/OUT)
CF9D	D041	2. funkció (ED-BKIN/OUT)
CF9F	D01D	3. funkció (CUPOS)
CFA1	D013	4. funkció (CUFIX)

3.5.4.2 A vezérlőrutinok

Amikor az operációs rendszer megszakításkezelő alprogramjába az editor be van kapcsolva, akkor annak kiszolgálása a kurzor előállítását, illetve villogtatását eredményezi.

Tudjuk, hogy az IT rutin az egyes eszközöket a 0. funkció hívásával szolgálja ki, ennek is ismerjük már a mechanizmusát.

Az editor kurzorrutinja a következő.

EDITOR 0	ED-INT		
20, A0	A kurzor villogtatása		
in:	-		
out:	-		
CFA3 3E50	LD A,50	U0-U1-VID-SYS lapozást állít	
CFA5 D302	OUT (02),A	be	
CFA7 21480E	LD HL,0E48	Számláló a kurzor	
		villogtatás időzítéséhez	
CFAA ED4B490E	LD BC,(0E49)	A kurzor pozíciója	
CFAE 34	INC (HL)	Növeli a számlálót	

CFAF 3E94	LD	A, 94	
CFB1 96	SUB	(HL)	Ha már elérte a 94H-t,
CFB2 281C	JR	Z, CFDO	ugrás: a kurzort törli
CFB4 FE80	CP	80	Amíg a számláló nem 14H,
CFB6 C0	RET	NZ	visszatérés
CFB7 77	LD	(HL), A	Ha elérte: átírja 80H-ra és
			hozzákezd a kurzor
			megjelenítéséhez
CFB8 3A660B	LD	A, (0E66)	LOCK-KEY, a billentyűzet
CFBB 167F	LD	D, 7F	LOCK állapotát jelzi
CFBD B7	OR	A	
CFBE 280C	JR	Z, CFCC	00 esetén alapállapot,
CFC0 169E	LD	D, 9E	b=1 a CTRL, b1=1 SHIFT, b3=1
CFC2 0F	RRCA		az ALT-LOCK állapotot jelzi.
CFC3 3807	JR	C, CFCC	Ezeknek megfelelően a D-be
CFC5 169F	LD	D, 9F	az
CFC7 0F	RRCA		alap (7F), inverz C (9E),
CFC8 3802	JR	C, CFCC	inverz S (9F) vagy inverz A
CFCA 168F	LD	D, 8F	(8F) kódját teszi és az
CFCC 7A	LD	A, D	adott kurzor pozíción
CFCD C320D4	JP	D420	kurzorként megjeleníti
			Ha a számláló 94H lett:
CFDO 77	LD	(HL), A	visszaállítja az eredeti
CFD1 C391D4	JP	D491	karaktert

Az eddigiekből látszik, hogy a rutin a 0E48H-s számlálón egyszerűen nyilvántartja és időzíti a kurzor mind be-, mind pedig kikapcsolt állapotát: 00...14H között az eredeti karakter, 80H...94H között a kurzor látszik a képernyőn, az utóbbi egyben a billentyűzet pillanatnyi állapotát is jelzi.

A szubrutin a 0D420H vagy a 0D491H címekre ugorva fejezi be a munkáját. Ezekkel a programrészekkel később megismerkedünk. Feladatuk a kurzor megjelenítése idejére az eredeti képernyőtartalom tárolása, a kurzor kirakása, ill. ugyanezek visszaállítása.

Az editor inicializáló rutinja következik. Az alaphelyzetet eredményező egyszerű beállításokat minden CLS-hívás elvégzi.

EDITOR INIC			Az editor munkaterületének és belső változóinak
CFD4H			alaphelyzetbe állítása
CFD4 21480E	LD	HL, 0E48	Először is törli a
CFD7 0620	LD	B, 20	0E48...0E67H területet,
CFD9 AF	XOR	A	amely a belső változókat

CFDA 77	LD (HL),A	és a képernyő soraira
CFDB 23	INC HL	vonatkozó információkat
CFDC 10FC	DJNZ CFDA	tartalmazza
CFDE 210101	LD HL,0101	A kurzorpozíciót az elejére
CFE1 22490E	LD (OE49),HL	állítja
CFE4 210001	LD HL,0100	
CFE7 110101	LD DE,0101	A képernyő ASCII kódú
CFEA 01FF05	LD BC,05FF	másolatát betűközökkel törli
CFED 3620	LD (HL),20	(20H kód)
CFEF EDB0	LDIR	

A karakterek megjelenítését az editor a videorutinoktól függetlenül, önállóan végzi. Az egyes üzemmódokban használható karaktermegjelenítő rutinok kezdőcímét is az inicializáló rutin választja ki és helyezi el az editor munkaterületén. Feljegyzi továbbá az aktuális üzemmódban a képernyő egy sorában elférő karakterek számát, valamint, hogy egy karakter hány byte-ot foglal el a videomemóriában. Mindezeket egy táblázatból másolja ki a saját munkaterületére, s a továbbiakban, a tényleges működés során ezekkel a részkérdésekkel már nem kell foglalkoznia. Ez a megoldás lényegesen gyorsabb működést eredményez.

Az inicializáló rutin hátralévő része ezt a táblázatkezelést valósítja meg.

CFF1 3A730B	LD A,(0B73)	Az üzemmód: 00, 01, 02
CFF4 87	ADD A,A	Négygel szorozzuk, mert a
CFF5 87	ADD A,A	táblázat elemei négy byte-
		ból állnak
CFF6 2107D0	LD HL,D007	A táblázat eleje
CFF9 4F	LD C,A	
CFFA 09	ADD HL,BC	Az üzemmódhoz tartozó adatok
		helye
CFFB 11680E	LD DE,0E68	Ide tesszük az ugrási
CFFE 3EC3	LD A,C3	utasítás műveleti kódját,
D000 12	LD (DE),A	mögé pedig a megfelelő
D001 13	INC DE	szubrutin kezdőcímét, a
D002 0E04	LD C,04	sorhosszt és a
D004 EDB0	LDIR	karakterméretet
D006 C9	RET	

A felhasznált táblázat a következő:

ROM- cím	A rutin címe	A sor hossza	Egy karak- ter mérete	üzemmód kód	jelentés
D007	D3AD	40	01	00	2-es
D00B	D3BC	20	02	01	4-es
D00F	D3D9	10	04	02	16-os

A ROM következő részében a 04-es funkcióhívás rutinja helyezkedik el.

```
EDITOR 04      CU-FIX
24H, A4H      Az aktuális kurzor-pozíció megjegyzése
               in: -
               out: -
```

```
D013 2A490E    LD    HL, (0E49)    Az aktuális pozíciót
D016 224E0E    LD    (0E4E), HL      átmásolja, majd e
D019 3E80      LD    A, 80           szolgáltatás kérését
D01B 181C      JR    D039         adminisztrálja a 0E4DH címre
                                   írt 80H-val
```

Az aktuális pozíció megjegyzése azzal jár, hogy az ezt követő karakter- és blokkbeolvasási funkciók a karakterek beolvasását a sor megjegyzett pozícióján kezdik. Előtte tehát olyan járulékos információk helyezhetők el, amelyek a feldolgozás szempontjából figyelmen kívül hagyhatók.

```
EDITOR 03      CU-POS
23H, A3H      Az editor kurzorát a megadott karakterpozícióra
                                   állítja
               in: B=a soron belüli pozíció (oszlop) 1...16,
                   1...32, vagy 1...64 üzemmód szerint
                   C=a sorszám: 1...24
               out: A=00 nincs hiba
                   F6 lehetetlen pozíció
```

```
D01D 2A490E    LD    HL, (0E49)    Az aktuális kurzorpozíció
D020 78        LD    A, E        A megadott új oszlopszám
D021 B7        OR    A        0?
D022 2807      JR    Z, D02E      Ha igen: nem változtat
D024 3A6B0E    LD    A, (0E6B)    Az érvényes sorhossz
D027 B8        CP    B        A megadott oszlop-pozíció
                                   nagyobb?
D028 3814      JR    C, D03E      Ha igen: hiba,
D02A 60        LD    H, B        egyébként elfogadja
D02B 79        LD    A, C        A megadott sorszám
D02C B7        OR    A        0?
D02D 2806      JR    Z, D035      Ha igen: nem változtat
D02F 3E18      LD    A, 18         A maximális sorszám
D031 B9        CP    C        Ha a megadott ennél is
D032 380A      JR    C, D03E      nagyobb: hiba
```


D034 69	LD	L,C	Egyébként elfogadja és az új
D035 22490E	LD	(0E49),HL	pozíciót beiktatja az erre
D038 AF	XOR	A	vonatkozó jelzéssel együtt
D039 324D0E	LD	(0E4D),A	
D03C AF	XOR	A	Hiba nincs
D03D C9	RET		
D03E 3EF6	LD	A,F6	Ha rossz értékeket adtak meg,
D040 C9	RET		a hibakóddal tér vissza

Ezután ismét egy nagyobb lélegzetvételi munkához látunk: az editor egyik legfontosabb funkciója a karakteres, ill. blokkos beolvasás és kiküldés.

Alaphelyzetben, elő beolvasáskor az editor a billentyűzetről bekért karaktereket tárolja a saját ASCII képernyőjén, egyben megjeleníti a tv-képernyőn is, s az összes szerkesztési szolgáltatással a rendelkezésünkre áll. Ezt a munkát az ESC, CTRL+ESC, vagy a RETURN lenyomásakor fejezi be. Ezután kezdi el a tulajdonképpeni karakterbeolvasást azzal, hogy a hívó programnak átadja a hívás kezdete óta írt "bekezdés" első karakterét (RETURN) esetén.

Az ESC lenyomása esetén csak annak kódját adja vissza és a bekezdés pillanatnyilag érvénytelen lesz, míg a CTRL+ESC lenyomása a szokásos STOP állapotot eredményezi.

Az editorral kapcsolatban egy bekezdésnek tekintünk a képernyőn olyan szövegrészt, amely két nem teljesen teleírt sor között helyezkedik el. Egy sor tele van, ha utolsó karaktere is foglalt.

A sorok foglaltságát a sorjellemzők mutatják. Értékük minden pillanatban a soron belüli, írt karakterek száma, de ha a sor teljesen tele van -- ez az üzemmódtól is függ -- akkor a sorjellemző b7 bitje 1.

Ha bekezdés érvényes volt (a végén RETURN), akkor ismételt input hívásokra a bekezdés egymást követő karaktereit adja vissza, legutoljára a lezáró RETURN kódját is. Újabb input hívás esetén az előbbiek ismétlődnek.

Az output funkciók egyszerűbben zajlanak le. Ekkor az editor a karaktereket nem a billentyűzetről olvassa be és nem átadja azokat a hívó rutinnak, hanem éppenséggel a CPU C regiszterében kapja az őt aktiváló programtól. A karakterek kezelése egyébként ugyanúgy történik, mint ha a billentyűzetről olvasta volna be. A karaktereket az editor a saját ASCII képernyőjén is tárolja, tehát pl. input hívásokkal később feldolgozhatók. Az így kapott karaktereket az editor ahhoz a bekezdéshez tartozónak tekinti,

amelyben a kurzor éppen van. Ezen belül a bekezdés korábban beírt karaktereit felülírja, de más bekezdéseket nem ront el: azokat szükség esetén a képernyőn -- és saját ASCII képernyőjén is -- elmozgatja (új, üres sorokat szúr be).

Az összes szerkesztési műveletre igaz (átírás, beszúrás, kiejtés stb.), hogy ezek egy bekezdésen belül zajlanak és a többi bekezdésre mindaddig nincsenek káros hatással, ameddig azok a képernyőről esetleg le nem szorulnak.

A blokkos input és output műveletekben a feldolgozásra kerülő karakterek teljes számát határozhatjuk meg. Input esetén az editorból beolvasott karakterek a memória megadott címekre tárolódnak.

Az editor szerkesztési parancsait, a megfelelő funkcióbillentyűket és a parancsok hatását a velük kapcsolatos programrészek ismertetésénél adjuk meg.

Először a funkcióhívások vezérlőrutinjaival foglalkozunk.

EDITOR 02	BK-IN, BK-OUT	
A2H, 22H	Karakterblokk beolvasása vagy kiküldése. Az A2H funkciókód esetén blokkbeolvasás, míg a 22H kóddal blokk output valósul meg	
	in: DE=memória kezdőcím, ahová -- vagy ahonnan -- a karaktereket mozgatni kell	
	BC=a karakterek száma	
	out: A=00 hibátlan	
	FA a megengedett legnagyobb memóriacím átlépése	
D041 D9	EXX	Előállítja és tárolja az INK
D042 CD49D4	CALL D449	(0E95) és PAPER színű (0E96)
D045 D9	EXX	vonalybyte-ot
D046 21E7D0	LD HL, D0E7	Az output rutin címe
D049 FZ6DC5	JF F, C56D	OUT-CHARS, vezérlőrutin

Az univerzális blokk-output vezérlőprogram -- mint láttuk korábban -- a legmagasabb memóriacím folyamatos ellenőrzésével irányítja a folyamatot. Tudjuk azonban azt is, hogy sajnálatos programozási hiba folytán éppen ezt a funkciót nem tátja el. Sőt, amikor be kellene avatkoznia, akkor az egész rendszer eltéréését okozza.

Ha nem output funkció volt, akkor a program itt folytatódik:

D04C 2158D0	LD	HL,D058	Az input rutin címe
D04F C38FC5	JP	C58F	IN-CHARS, vezérlőrutin

Ezzel ugyanaz a helyzet, mint az output vezérlővel.

Most következik a munka lényegi része: bár még mindig csak a vezérlés szintjén.

EDITOR 01	ED-CHIN, ED-CHOUT
A1H, 21H	Karakter beolvasása, karakter kiküldése. Az A1H funkciókód esetén egy karakter beolvasását végzi el -- a leírtak szerint --, míg a 21H kóddal egy karakter output valósul meg

in: C=a beolvasott, vagy a kiküldendő karakter kódja
out: A=00 nem lépett fel hiba

D052 CD49D4	CALL	D449	Előállítja és tárolja az INK (0E95) és PAPER színű (0E96) vonalbyte-ot
D055 F2B7D0	JP	P,D0B7	Output hívás esetén ugrás a megfelelő belépési pontra

Itt a karakteres input vezérlése folytatódik.

D058 3E50	LD	A,50	U0-U1-VID-SYS konfigurációban
D05A 320300	LD	(0003),A	belapozzuk a videomemóriát
D05D D302	OUT	(02),A	
D05F 3A4D0E	LD	A,(0E4D)	Az input típusa
D062 0F	RRCA		Ismételt hívás?
D063 DAFAD0	JP	C,D0FA	Ha igen: ugrás előre

Láttuk már, hogy a 0E4D belső változó 80H értéke a beolvasás kezdetének helyét meghatározó "kurzorpozíció megjegyzésre" utal, 00 esetén új beolvasást, 01 értékkel pedig ismételt hívásokat jelent.

Utóbbi esetén a rutinnak nem kell a billentyűzetről adatok begépelésére várni, a korábban beírt karaktereket a saját ASCII pufferéből küldi a hívó rutin felé.

Ha új beolvasás van, akkor a program itt folytatódik:

```
D066 CD9CD3      CALL   D39C      Az aktuális sorjellemező és
                                     címe
```

A sorok állapotát az editor a 0E50...0E67H terület egy-egy byte-ján tartja nyilván. Értéke szabad sor esetén 00, a teljesen teleírt sorokat b7=1 bejegyzéssel látja el. A részben megírt sorokban b7=0 és a többi biten az adott sorban található karakterek számát tartja nyilván a rendszer.

Az előbbi D3D9H-s szubrutin a kurzor sorához tartozó sorjellemezőt teszi a HL-be, A a sorban levő karakterek számát tartalmazza b7=0-val, azonban a carry-ben jelzi a sor foglaltságát.

A program következő része a megjegyzett kurzorpozíció érvényességét vizsgálja meg. Ha a tárolt pozíció az aktuális pozíció után van, akkor érvényteleníti.

```
D069 ED4B490E    LD     BC,(0E49)      Az aktuális kurzorpozíció
D06D C5          PUSH  BC
D06E ED5B4E0E    LD     DE,(0E4E)      A megjegyzett pozíció
D072 79          LD     A,C
D073 BB          CP     E              Ha a megjegyzett sorszám
D074 3806        JR     C,D07C        nagyobb: törli
D076 2009        JR     NZ,D081       Ha kisebb: tovább kell nézni

D078 78          LD     A,B
D079 BA          CP     D              Ha ugyanabban a sorban
D07A 300F        JR     NC,D08B       van a megjegyzett
                                     kurzorpozíció, de nem az
                                     aktuális hely után: meghagyja

D07C CD38D0      CALL  D038           Egyébként a pozíció
D07F 180A        JR     D08B           megjegyzését törli és
                                     ugrás a szerkesztő hurokba

D081 0D          DEC    C              Megnézi az előző sort
D082 28F8        JR     Z,D07C        Ha az első sor volt:
                                     érvénytelenítés
D084 2B          DEC    HL            Az előző sorjellemező
D085 CB7E        BIT    7,(HL)
D087 1601        LD     B,01
D089 20E7        JR     NZ,D072       A sor elejére áll, s ha az
                                     aktuális sor még foglalt:
                                     ugrás vissza egyébként kezdi
                                     a szerkesztést

D08B C1          POP   BC            A kurzor aktuális pozíciója
```

D08C CD77D4	CALL D477	Saját munkaterületére másolva kimentti a videomemóriából a kurzor helyén álló karaktert,
D08F 3E13	LD A,13	a kurzor időzítését azonnali váltásra állítva a CU-INT rutint bekapcsolja az IT-be és vár egy billentyűre
D091 32480E	LD (0E48),A	
D094 21100B	LD HL,0E10	
D097 CB96	RES 2,(HL)	
D099 F791	RST 30: 91	
D09B CBD6	SET 2,(HL)	Ezután tiltja a kurzort
D09D F5	PUSH AF	A hibakódot és a beolvasott billentyű kódját a veremben tárolja
D09E C5	PUSH BC	
D09F ED4B490E	LD BC,(0E49)	Ismét az aktuális pozíció
DOA3 CD91D4	CALL D491	Visszaállítja az eredeti karaktert és ha a hibakód nem 00 volt (pl CTRL+ESC): visszatér
DOA6 C1	POP BC	
DOA7 F1	POP AF	
DOA8 C0	RET NZ	

Ezután megkezdődhet a billentyűzetről beolvasott karakterek feldolgozása.

DOA9 79	LD A,C	A beolvasott kód
DOAA FE1B	CP 1B	ESC?
DOAC 288A	JR Z,D038	Ha igen: input alaphelyzetet állít be (0E4D) és C=1BH-val visszatér
DOAE FE0D	CP 0D	RETURN?
DOB0 2839	JR Z,DOEB	Ha igen: ugrás az ezt feldolgozó részhez
DOB2 CD24D1	CALL D124	A többi karakter feldolgozása
DOB5 18AF	JR D066	Majd ugrás vissza, s ez így megy tovább, egészen addig, amíg a kilépést eredményező 3 billentyű egyikét le nem nyomják.

Ez a vezérlőrutin -- amint majd látni fogjuk -- a 0D124H szubrutin révén meglehetősen nagy területeket mozgat meg. Végül is ennek a munkája az egész editor működésének egyik leglényegesebb része.

Most azonban még a karakteres output funkció rutinja előtt állunk. Ha a funkciókód 21H, akkor a vezérlést a következő utasítássorozat kapja meg:

DOB7 3E50	LD	A,50	Először belapozzuk a
DOB9 320300	LD	(0003),A	videomemóriát
DOBC D302	OUT	(02),A	
DOBE 2138D0	LD	HL,D038	Az egyszerűség kedvéért RET-
DOC1 E5	PUSH	HL	tel fogunk a D038-ra ugrani, JP
			helyett!
DOC2 79	LD	A,C	A kiküldendő karakter
DOC3 FE0D	CP	0D	RETURN?
DOC5 2006	JR	NZ,DOCD	Ha nem: ugrás tovább

A RETURN feldolgozása:

DOC7 3E01	LD	A,01	
DOC9 324A0E	LD	(0E4A),A	A kurzor a sor elejére és kész!
DOCC C9	RET		
DOCD FE0A	CP	0A	LF? (soremelés)
DOCF 2015	JR	NZ,DOE6	Ha nem: ugrás tovább

A soremelés feldolgozása:

DOD1 ED4B490E	LD	BC,(0E49)	A kurzor pozíciója
DOD5 79	LD	A,C	A sorszám
DOD6 CD9FD3	CALL	D39F	Foglalt a sor?
DOD9 0C	INC	C	(a következő sorra áll)
DODA 38F9	JR	C,DOD5	Ha igen: tovább, ha nem
DODC 0D	DEC	C	visszaáll a szabad sorra (a
			bekezdés utolsó sora)
DODD C5	PUSH	BC	
DODE CD63D3	CALL	D363	Majd az új sor elejére, de ezen
DOE1 F1	POP	AF	belül az eredeti oszlop-
DOEZ 324A0E	LD	(0E4A),A	pozícióra áll és kész.
DOE5 C9	RET		
DOE6 CD24D1	CALL	D124	Ha a küldött karakter nem 0D és
			nem 0A, akkor külön dolgozza
			fel
DOE9 AF	XOR	A	
DOEA C9	RET		Utána kész

Ezután a vezérlő funkciók kivitelezésének nagyszabású munkálatai kezdődnek.

Elsőként nézzük, mi történik, amikor a billentyűzetről való adatbevitel végén lenyomjuk a lezáró RETURN-t. Erre a vezérlést végző szubrutin kilép a karakterbekérő és szerkesztő ciklusból, s megkezdődik a beolvasott karakterek felküldése a hívó programnak.

A RETURN az első karakter "feladását" is elvégzi. Figyelembe kell azonban venni, hogy volt-e érvényes kurzorpozíció-megjegyzés, mert ha igen, akkor azon a helyen kell kezdeni és nem a bekezdés elején!

E1 RETURN feldolgozása a billentyűzetről történő
D0EB karakterbeolvasás végén

DOEB 3A4D0E	LD A, (0E4D)	Volt pozíció-megjegyzés?
DOEE 07	RLCA	Ha igen, carry=1
DOEF ED4B4E0E	LD BC, (0E4E)	A megjegyzett pozíció
DOF3 D470D3	CALL NC, D370	Ha nincs előírva, hogy hol kezdeni, akkor a bekezdés elejére áll
DOF6 ED43490E	LD (0E49), BC	Az érvényes pozíció

Ennyi volt a RETURN ág önálló munkája: a feldolgozandó első karakterre állt. Ezután ezt a karaktert még ténylegesen be is kell küldeni, s ez némi további adminisztrációval is jár. Viszont ez a munka már azonos az ismételt input-hívások kiszolgálásával. A RETURN rutin is ezzel folytatódik.

E3 Ismételt input kérések kezelése
D0FA

D0FA ED4B490E	LD BC, (0E49)	Az aktuális kurzorpozíció
DOFE 79	LD A, C	Nézzük ezt a sort!
DOFF CD9FD3	CALL D39F	A-ban az utolsó írt pozíció.
D102 B8	CP B	Ha a kurzor pozíciója nagyobb, akkor a végére értünk
D103 3817	JR C, D11C	
D105 CD84D3	CALL D384	Egyébként vesszük a következő karakter címét az ASCII pufferben
D108 04	INC B	A következő pozíció
D109 3A6B0E	LD A, (0E6B)	
D10C B8	CP B	Ha még nem nagyobb, mint a sorhossz: rendben,
D10D 3003	JR NC, D112	egyébként a következő sor elejére állunk
D10F 0601	LD B, 01	
D111 0C	INC C	
D112 ED43490E	LD (0E49), BC	Ez lesz a következő hely
D116 4E	LD C, (HL)	Beolvassuk a karaktert,
D117 3E01	LD A, 01	0E4DH-n beállítjuk az "ismételt input" jelzést és kész
D119 C339D0	JF D039	

E4 Ismételt input lezárása a bekezdés végén:
D11C egy RETURN kód küldése és input alaphelyzet

D11C	CD63D3	CALL	D363	A kurzor a következő sor
D11F	0E0D	LD	C,0D	elejére áll, majd C=0D és
D121	C338D0	JP	D038	beállítja az alaphelyzetet

Az editor legfontosabb vezérlőrutinjához érkeztünk. Elvégzi a 20...DFH kódú karakterek kiírását és tárolását az ASCII pufferben, ennél is fontosabb azonban, hogy kezeli az összes vezérlőfunkciót.

A különböző szerkesztési feladatokhoz tartozó rutinok elérése a megszokott ugrótábla kezelési technikával történik.

Először a vezérlő programrészt ismerjük meg.

E5 A beolvasott karakterek feldolgozása (a rutin a
D124H 20...DFH kódú karakterek kiírását és a
 szerkesztési funkciók végrehajtását vezérli
in: A=feldolgozandó kód

D124	FE20	CP	20	Vezérlőkarakter?
D126	38ZF	JR	C,D157	Ha igen: ugrás az ugrótábla kezelőhöz
D128	FEE0	CP	E0	Ha a kód DF-nél nagyobb,
D12A	302B	JR	NC,D157	akkor is oda ugrik

Itt a kiírható karakterek feldolgozása következik.

D12C	08	EX	AF,AF	Előállítja az aktuális
D12D	CD84D3	CALL	D384	kurzorpozícióhoz tartozó
D130	08	EX	AF,AF	ASCII-puffer címet és
D131	77	LD	(HL),A	leteszi a karaktert
D132	ED4B490E	LD	BC,(0E49)	Az aktuális kurzorpozíció
D136	C5	PUSH	BC	alapján a képernyőn is
D137	CD20D4	CALL	D420	megjeleníti
D13A	C1	POF	BC	

Ezután rá kell még állni a következő pozícióra.

D13B	79	LD	A,C	Megnézzük az aktuális
D13C	CD9FD3	CALL	D39F	sorjellemezőt. Ha már korábban
D13F	B8	CP	B	is volt itt karakter, akkor ezt
D140	3001	JR	NC,D143	most nem kell adminisztrálni,

D142 70	LD (HL),B	egyébként ez lesz az új sorjellemző
D143 04	INC B	Jobbra lépünk, s
D144 3A6B0E	LD A,(0E6B)	ha még nem léptünk túl a sor végén: rendben,
D147 B8	CP B	mehet tovább
D148 3008	JR NC,D152	

Ha azonban B értéke nagyobb lett, mint a 0E6BH-n tárolt sorhossz, akkor új sort kell kezdeni.

D14A 0601	LD B,01	Az első oszlop-pozíció,
D14C 0C	INC C	a következő sor
D14D CB7E	BIT 7,(HL)	Ha az eredeti sor eddig még nem volt foglalt,
D14F CCCBD2	CALL Z,D2CB	lebonyolítja az új sor kezdését
D152 ED43490E	LD (0E49),BC	Az új kurzorpozíciót elteszi
D156 C9	RET	és kész

Majd nemsokára látni fogjuk, hogy egy új sor kezdése mennyi munkával jár (most csak gondoljunk pl. a sor alatti bekezdések esetleg szükségessé váló elmozgatására).

Ennyi volt a kiírható karakterek feldolgozása. A vezérlőrutin másik, itt következő ága a szerkesztési funkciók felismerését és kezelését végzi.

E5 A vezérlőkarakterek azonosítása és a kért funkciók felé a vezérlés átadása
D157H in: A=a vezérlőkód

D157 2170D1	LD HL,D170	A beépített funkciók azonosító táblázata
D15A 060B	LD B,0B	11-féle lehet
D15C BE	CP (HL)	Ez az?
D15D 23	INC HL	
D15E 5E	LD E,(HL)	DE-be beolvassa a végrehajtó rutin kezdőcímét, majd
D15F 23	INC HL	
D160 56	LD D,(HL)	
D161 23	INC HL	
D162 2803	JR Z,D167	a címmel ugrás előre
D164 10F6	DJNZ D15C	Ha a kód nem ez volt: folytatja a keresést
D166 C9	RET	Ha egyikkel sem egyezett a kód, munka nélkül tér vissza

Az azonosított vezérlőkód feldolgozása:

D167 ED4B490E	LD BC,(0E49)	A kurzor pozíciója
---------------	--------------	--------------------

D16B 3A6B0E	LD	A, (OE6B)	A sorhossz
D16E EB	EX	DE, HL	HL-ben a rutin címe és
D16F E9	JP	HL	máris mehet!

Nézzük a lehetséges funkciókat és belépési pontokat.

Az editor szerkesztési funkcióinak azonosító és ugrótáblája:

ROM- cím	Tartalom	Kód	Rutin- cím	Funkció
D170	13 91D1	13	D191	LEFT kurzor balra
D173	04 98D1	04	D198	RIGHT kurzor jobbra
D176	05 95D1	05	D195	UP kurzor fel
D179	18 9ED1	18	D19E	DOWN kurzor le
D17C	16 05D2	16	D205	INS karakterbeszúrás
D17F	07 87D2	07	D287	DC törlés a kurzornál
D182	08 78D2	08	D278	DEL törlés a CU előtt
D185	19 FDD1	19	D1FD	DL sortörlés
D188	0E F7D1	0E	D1F7	IL sorbeszúrás
D18B	0B A8D1	0B	D1A8	CEL törlés a bekezdés végéig
D18E	09 CDD1	09	D1CD	TAB új tabulátor pozícióra

Sorra vesszük az egyes funkciókhoz tartozó rutinokat.

Az első négy funkció tulajdonképpen egyetlen rutin, 4 belépési ponttal. A kurzor helyének megváltoztatása mindössze abból áll, hogy a megfelelő ellenőrzések elvégzése után átírjuk az aktuális pozíciót tároló belső változókat.

Íme:

LEFT	A kurzor léptetése balra
D191	

D191 05	DEC B	Csökkentjük az oszlop pozíciót, s ha még nem lett 00: kész is
D192 200F	JR NZ, D1A8	
D194 47	LD B, A	Ha már a sor eleje volt, akkor először is ennek a sornak a végére megyünk

A sor végén természetesen még egy "kurzor fel" funkciót kell lebonyolítani, így ezzel a feladattal az előző rutin "átcsoroghat" a következőre.

UP D195H	A kurzor léptetése felfelé	
D195 0D D196 180A	DEC C JR D1A2	Csökkenti a sorszámot és kész is, az ellenőrzést lentebb könnyen elvégezhetjük
RIGHT D198H	A kurzor léptetése jobbra	
D198 04 D199 B8 D19A 3007	INC B CP B JR NC,D1A3	Növeljük az oszlop pozíciót, s ha még nem léptünk túl a sor végén: kész is
D19C 0601	LD B,01	Egyébként visszamegyünk a sor elejére

Ekkor még egy "kurzor le" funkciót kell végrehajtani, így a rutin "átcsoroghat" a következőre.

DOWN D19EH	A kurzor léptetése egy sorral lejjebb	
D19E 0C D19F 79 D1A0 FE19 D1A2 C8	INC C LD A,C CP 19 RET Z	Növeljük a sorszámot Ha ezzel az utolsó sor alá mennénk, lehetetlen funkcióként értékelve nem teszünk semmit
D1A3 ED43490E D1A7 C9	LD (OE49),BC RET	Egyébként pedig beírjuk az új pozíció adatait

Valamivel nagyobb fajsúlyú problémák megoldása következik:

CEL D1A8	Törlés a kurzortól a bekezdés végéig Ezt a funkciót a CTRL+K billentyűkombináció váltja ki	
D1A8 C5 D1A9 CDADD4	PUSH BC CALL D4AD	Tároljuk a pozíciót Töröljük a képernyőn a kurzor sorát
D1AC C1 D1AD CD84D3	POP BC CALL D384	A kurzor címe az ASCII pufferben

D1B0 05	DEC B	A pozícióból és a sorhosszból
D1B1 3A6B0E	LD A, (OE6B)	előállítjuk a letörölt
D1B4 90	SUB B	sorrész hosszát, ennek
D1B5 3620	LD (HL), 20	alapján a megfelelő részt az
D1B7 23	INC HL	ASCII "képernyőn" is töröljük
D1B8 3D	DEC A	
D1B9 20FA	JR NZ, D1B5	
D1BE CD9CD3	CALL D39C	HL=a sorjellemző címe,
D1BE 70	LD (HL), B	beírjuk az új értéket
D1BF 0C	INC C	A következő sor
D1C0 23	INC HL	és a sorjellemző címe
D1C1 D0	RET NC	Ha a sor nem volt foglalt,
		tehát a bekezdés utolsó sora
		volt: kész
D1C2 46	LD B, (HL)	Egyébként nézzük az új sort is
D1C3 79	LD A, C	A vizsgált sor száma
D1C4 D9	EXX	
D1C5 CD1ED3	CALL D31E	Itt már elég egy sorkiejtés
D1C8 D9	EXX	
D1C9 CB00	RLC B	A sorjellemző b7 bitje
		mutatja, hogy ez a sor foglalt
D1CB 18F4	JR D1C1	volt-e. Ugrás vissza

Ezzel kész, hiszen ha a bekezdés utolsó sorát is kiejtettük, akkor ezzel a kért funkciót megvalósítottuk.

A közben felhasznált szubrutinokkal lesz még dolgunk a továbbiakban.

TAB	A kurzort a következő tabulátor-pozícióra állítja	
D1CDH	(CTRL+I)	
D1CD CD9CD3	CALL D39C	A sorjellemzővel kezdjük és az
D1D0 08	EX AF, AF	eredményt tároljuk
D1D1 78	LD A, B	Az eredeti pozícióból
D1D2 3D	DEC A	áttérünk a 0-val kezdődő
		számolásra, így a
D1D3 E6F8	AND F8	b2...b0 biteket nullázva az
D1D5 C609	ADD A, 09	előző tabulátor pozíciót
D1D7 47	LD B, A	kapjuk, melyhez 9-et adva
		elvileg kész
D1D8 3A6B0E	LD A, (OE6B)	Meg kell azonban nézni, hogy
D1DB B8	CP B	elfér-e még a sorban? Ha igen:
D1DC 300A	JR NC, D1E8	ugrás
D1DE F680	OR 80	Ha azonban túlmenne a soron,
D1E0 77	LD (HL), A	akkor beállítjuk a foglalt
		jelzést

D1E1 0601	LD B,01	A következő sor elejére állunk
D1E3 0C	INC C	
D1E4 08	EX AF,AF	Nézzük az eredeti sort
D1E5 D4CBD2	CALL NC,D2CB	Ha a bekezdés utolsó sora volt, akkor egy új sort kell felszabadítani
D1E8 79	LD A,C	
D1E9 CD9FD3	CALL D39F	Vesszük a sorjellemzőt
D1EC 05	DEC B	Ha a kiválasztott pozíció után
D1ED B8	CP B	ebben a sorban már nincs
D1EE 3001	JR NC,D1F1	korábban írt karakter, akkor
		ez lesz az új sorjellemző
D1F0 70	LD (HL),B	
D1F1 04	INC B	Végül az így kikövetkeztetett
D1F2 ED43490E	LD (0E49),BC	koordinátákat beiktatjuk
D1F6 C9	RET	és kész

A sorbeszúrás és sortörlés két -- formailag nagyon rövid -- szubrutinja következik.

IL Sorbeszúrás
D1F7H

D1F7 CD70D3	CALL D370	A kurzort a bekezdés elejére állítjuk és ugrik a sorbeszúró rutinra
D1FA C3D6D2	JP D2D6	

DL Sortörlés
D1FDH

D1FD CD70D3	CALL D370	A kurzort a bekezdés elejére állítja
D200 CD9CD3	CALL D39C	Előkészíti a sorjellemzőt és
D203 18BD	JR D1C2	ugrás vissza

Ehhez csak annyit teszünk hozzá, hogy a 0D1C2H címre való beugrás a sor átírását eredményezi az alatta levő sorokkal, ami valóban a kívánt funkció.

Ezután egy -- az eddigieknél hosszabb utasítássorozattal megvalósítható -- funkció következik, a karakterbeszúrást lebonyolító program.

INS Karakterbeszúrás
D205H

D205 CD9CD3	CALL D39C	Előállítja a sorjellemzót. Ha a kurzor a bekezdés végén áll: értelmetlen
D208 E8	CP B	
D209 D8	RET C	
D20A 1E01	LD E,01	Számláljuk a sorokat a bekezdés végéig. Először ugrás 2 sorral lejjebb
D20C 1802	JR D210	A következő sorjellemző címe
D20E 23	INC HL	
D20F 1C	INC E	Számlálás
D210 7E	LD A,(HL)	A sorjellemző
D211 07	RLCA	A sor tele van?
D212 0C	INC C	A következő sorra áll
D213 38F9	JR C,D20E	Ha tele volt: tovább!

Amikor idáig jut, megtalálja a bekezdés végét. Az E értéke mutatja, hogy hány soron át kell az INS-t végezni, C pedig már a bekezdés utáni sorra mutat.

D215 34	INC (HL)	Az új sorjellemző
D216 3A6B0E	LD A,(OE6B)	A sorhossz
D219 BE	CP (HL)	A sor vége lesz?
D21A 2017	JR NZ,D233	Ha még nem: ugrás előre

Ha itt folytatódik, ez azt jelenti, hogy a sor végére értünk.

D21C 7B	LD A,E	Hány soros INS
D21D 81	ADD A,C	+ a következő sor száma
D21E FE31	CP 31	

Ez akkor lehet 31H, ha C=19H és E=18H, vagyis a bekezdés a teljes képernyőt betölti és a kurzor az első sorban van.

D220 2004	JR NZ,D226	Ha nem így van: ugrás előre
D222 35	DEC (HL)	Ha az utolsó sor utolsó karaktere lenne: visszaáll,
D223 37	SCF	ezt carry-ben jelzi és ugrás előre
D224 180E	JR D234	

Itt folytatódik a programfutás akkor, ha nem a teljes képernyőt betöltő bekezdés utolsó sorának végén vagyunk. Ekkor beiktathatunk egy új sort anélkül, hogy a bekezdésnek akár egyetlen karaktere is elveszne.

D226 D5	PUSH DE	Az INS sorok száma
---------	---------	--------------------

D227 C5	PUSH BC	A bekezdés utáni sor
D228 CDCBD2	CALL D2CE	Egy új sor beszúrása
D22B F1	POP AF	A=az eredeti pozíció,
D22C D1	POP DE	E=az INS sorok száma
D22D 47	LD B,A	
D22E 79	LD A,C	Az új sor száma, majd a
D22F 3D	DEC A	bekezdés utolsó sora és a
D230 CD9FD3	CALL D39F	sorjellemző
D233 B7	OR A	

Itt tehát már minden elő van készítve. HL a bekezdés utolsó sora sorjellemzőjének címére, C a következő sorra mutat. Ha az INS miatt az utolsó sor is betelne, akkor létrehoztunk egy új sort. Legsúlyosabb esetben, a teljes képernyőt betöltő bekezdés esetén a carry 1 értékkel jelzi a gondot.

D234 08	EX AF,AF	Eltesszük ezt a jelzést
D235 B7	OR A	
D236 0D	DEC C	A bekezdés utolsó sora
D237 1D	DEC E	Az utolsó INS-sor?
D238 D5	PUSH DE	Tároljuk
D239 C5	PUSH BC	a pozíciót is,
D23A E5	PUSH HL	a sorjellemző címét is,
D23B F5	PUSH AF	és a választ is (F-ben)
D23C 2802	JR Z,D240	Az utolsó INS sorban már a
		kurzor pozícióját vesszük,
D23E 0601	LD B,01	a többi sorban az elejét
D240 C5	PUSH BC	A soron belüli INS kezd.
D241 CD92D5	CALL D592	Elmozgatás a képernyőn
D244 C1	POP BC	
D245 116B0E	LD DE,0E6B	
D248 1A	LD A,(DE)	A sorhossz
D249 67	LD H,A	A sor utolsó pozíciója
D24A 69	LD L,C	A sor száma
D24B CD87D3	CALL D387	HL=ennek ASCII címe
D24E E5	PUSH HL	Tároljuk
D24F 08	EX AF,AF	Most nézzük a jelzőbitet
D250 3002	JR NC,D254	ha nem a "lap" alján vagyunk:
		ugrás, ha viszont igen: itt az
		utolsó karakter elveszik
D252 2B	DEC HL	
D253 04	INC B	
D254 1A	LD A,(DE)	Sorhossz
D255 90	SUB B	A=a mozgatandó byte-szám
D256 4F	LD C,A	
D257 0600	LD B,00	BC=hossz
D259 54	LD D,H	
D25A 5D	LD E,L	DE=végcím
D25B 2B	DEC HL	HL=kezdőcím

D25C 2802	JR	Z,D260	Ha a hossz 0, a következő sort átugorja
D25E EDB8	LDDR		Egyébként elmozgatás
D260 E1	POP	HL	A sor végének ASCII címe
D261 01C0FF	LD	BC,FFC0	Ez -40H
D264 09	ADD	HL,BC	Az előző sor vége
D265 46	LD	B,(HL)	Innen a karaktert -- hacsak már nem a kurzor sorában voltunk -- átmásoljuk az utoljára elmozgatott karakter helyére. A kurzor helyére betűköz kerül
D266 F1	POP	AF	
D267 2002	JR	NZ,D26B	
D269 0620	LD	B,20	
D26B 78	LD	A,B	
D26C 12	LD	(DE),A	
D26D E1	POP	HL	A veremből kiszedjük a sorjellemző címét,
D26E C1	POP	BC	az aktuális sorszámot,
D26F D1	POP	DE	a még hátralévő INS sorok számát, áttérünk az előző sorjellemzőre, s
D270 2B	DEC	HL	
D271 20C1	JR	NZ,D234	ha még van INS sor, ugrás vissza
D273 ED43490E	LD	(OE49),BC	Egyébként kész
D277 C9	RET		

A következő két funkció logikailag könnyen egybefoglalható. Az első esetben a kurzor is balra mozog, s az előtte álló karaktereket törli, miközben maga után mozgatja a bekezdés összes mögötte álló karakterét. A másik esetben a kurzor áll, s csak a bekezdésben mögötte álló karakterek mozognak balra, így egy-egy elmozdulásnál a kurzortól jobbra eső karakter törlődik.

Ha először a kurzort mozgatjuk el balra, törölve ezzel a kurzortól eredetileg balra eső karaktert, akkor utána már a kétféle feladat ugyanazt jelenti. A bekezdés kurzor mögötti részét kell balra elmozgatni.

Látható, hogy logikailag az első pontban belépve az ún. DEL funkció, a második ponttól pedig a DEL-CHAR (DC) funkció valósul meg.

Így szerveződik a következő két szubrutin is.

DEL		A kurzortól balra eső karakter törlése	
D278H			
D278 05	DEC	B	A kurzor balra lép

D279	2008	JR	NZ,D283	Ha még nem a sor elején voltunk, akkor a kurzor új pozíciója ezzel kész
D27B	0D	DEC	C	Ha a sor elején voltunk, akkor az előző sorba lépünk. Ez azonban az első sorban hatástalan
D27C	C8	RET	Z	
D27D	79	LD	A,C	A sorjellemzők
D27E	CD9FD3	CALL	D39F	Ha az előző sor már nem tartozik ehhez a bekezdéshez: nem lehet!
D281	D0	RET	NC	
D282	47	LD	B,A	
D283	ED43490E	LD	(0E49),BC	Az előző sor utolsó helye lesz az új kurzorpozíció

Ennyi volt tehát a DEL rutin önálló része. Ezután már csak egyszerűen végre kell hajtani azt, ami DEL-CHAR (DC) rutin feladata.

DEL-CHAR D287H	A kurzortól jobbra eső karakter törlése a bekezdés végéig tartó rész balra mozgatásával
-------------------	---

D287	CD9CD3	CALL	D39C	Nézzük a sorjellemzőt!
D28A	88	CP	B	Ha a kurzor a bekezdés végén áll: nincs tennivaló
D28B	88	RET	C	
D28C	08	EX	AF,AF	Eltesszük a sorjellemzőt
D28D	7E	LD	A,(HL)	
D28E	E67F	AND	7F	Kiszámítjuk a sor végétől való távolságot
D290	90	SUB	B	
D291	3830	JR	C,D2C3	Ha negatív: ugrás előre

Ez lehet negatív akkor, ha a bekezdés éppen egy sor végén fejeződik be. Ekkor a hozzá tartozó sorjellemző még foglaltat jelez, ugyanakkor a következő sor sorjellemzője már 0.

D293	E5	PUSH	HL	A sorjellemző címe
D294	F5	PUSH	AF	A távolság
D295	D5	PUSH	DE	Később a sor végének címe az ASCII képernyőn
D296	C5	PUSH	BC	A kezdő törlési pozíció
D297	CD42D5	CALL	D542	Végrehajtjuk az elmozgatást a képernyőn
D29A	C1	POP	BC	
D29A	B1	POP	DE	
D29C	60	LD	H,B	HL=a kezdő törlési pozíció
D29D	69	LD	L,C	

D29E CD87D3	CALL D387	Ennek ASCII képernyőcíme
D2A1 08	EX AF,AF	
D2A2 7E	LD A,(HL)	Az első törlendő byte
D2A3 3001	JR NC,D2A6	

Először nincs carry, de a későbbiekben ez azt jelzi, hogy korábban már egy előző sort a végéig elmozgattunk, így az új sor elején álló karaktert oda kell letenni.

D2A5 12	LD (DE),A	
D2A6 54	LD D,H	DE=az elmozgatás végcíme a sor elején
D2A7 5D	LD E,L	
D2A8 23	INC HL	HL=a sor 2. karakterének címe, kezdőcím
D2A9 F1	POP AF	A=a távolság a sor végétől
D2AA C5	PUSH BC	
D2AB 2805	JR Z,D2B2	Ha ez 0: ugrás előre
D2AD 4F	LD C,A	
D2AE 0600	LD B,00	BC=hossz
D2B0 EDB0	LDIR	Elmozgatás
D2B2 3E20	LD A,20	
D2B4 12	LD (DE),A	A végére üres helyköz
D2B5 C1	POP BC	
D2B6 0C	INC C	A következő sor
D2B7 0601	LD B,01	Elejére állunk
D2B9 E1	POP HL	A sorjellemző címe
D2BA CB7E	BIT 7,(HL)	Tele volt a sor?
D2BC 23	INC HL	
D2BD 37	SCF	Ha igen: előkészítjük a további munkát és ugrás vissza!
D2BE 20CC	JR NZ,D2BC	
D2C0 2B	DEC HL	Ha pedig nem volt tele, akkor visszaállunk erre,
D2C1 35	DEC (HL)	átállítjuk a sorjellemzőt
D2C2 C9	RET	és kész!

Most már csak azt a problémát kell lekezelnünk amikor a bekezdés éppen egy sor végén fejeződött be. Ilyenkor az utolsóként vizsgált szabad sorban egyetlen karakter sincs, sőt a törlés miatt már az előző sor vége is üresen maradt. Most a befejezés:

D2C3 2B	DEC HL	Visszaállunk az előző sorra,
D2C4 35	DEC (HL)	beállítjuk annak sorjellemzőjét,
D2C5 CBBE	RES 7,(HL)	megszüntetjük a foglaltságát
D2C7 79	LD A,C	és kiejtjük a szabaddá vált sort!
D2C8 C31ED3	JP D31E	

A karakterbeszúrás és a karaktertörlés rutinjaival az editor nagyon fontos -- és tegyük hozzá nem túl könnyű -- részein túl vagyunk.

Következik az új sor beiktatását lebonyolító szubrutin.

E18 Egy új sor beiktatása a szövegbe
D2CBH

in: HL=az aktuális sorjellemző címe
C=a következő sor sorszáma (ahol az üres sor kell)

D2CB CBFE	SET	7, (HL)	A sort lefoglaljuk, majd
D2CD 79	LD	A, C	nézzük a következő sort
D2CE FE19	CP	19	A 25. lenne?
D2D0 283F	JR	Z, D311	Ha igen: új sor csak a képernyősorok elmozgatásával lehetséges (roll)
D2D2 23	INC	HL	Ha még nem: nézzük a
D2D3 7E	LD	A, (HL)	sorjellemzőt
D2D4 B7	OR	A	Ha üres sor, nem kell semmit
D2D5 C8	RET	Z	tenni: kész

Ott tartunk, hogy még nem vagyunk a képernyő alján, azonban a következő sor nem üres. Ilyenkor kell végrehajtani a sorbeszúrást, az alatta levő sorok lefelé mozgatásával. Az editor úgy tekinti, hogy az éppen kezelt bekezdés a képernyőn a legfontosabb!

D2D6 C5	PUSH	BC	
D2D7 CD09D5	CALL	D509	Végrehajtja a sorbeszúrást a képernyőn
D2DA C1	POP	BC	
D2DB C5	PUSH	BC	
D2DC 3E19	LD	A, 19	
D2DE 91	SUB	C	Kiszámítjuk, hogy a C. sortól az utolsóig hány sort kell lemozgatni, s az utolsótól kezdve először végrehajtjuk a sorjellemzők elmozgatását
D2DF 21660E	LD	HL, 0E66	
D2E2 11670E	LD	DE, 0E67	
D2E5 D5	PUSH	DE	
D2E6 0600	LD	B, 00	
D2E8 4F	LD	C, A	
D2E9 EDB8	LDDR		
D2EB 13	INC	DE	
D2EC EB	EX	DE, HL	A beszúrt sor sorjellemzője 0: üres
D2ED 71	LD	(HL), C	
D2EE E1	POP	HL	A képernyő utolsó sorát szabadnak nyilvánítja
D2EF CBBE	RES	7, (HL)	

D2F1 1F	RRA	
D2F2 CB19	RR	C
D2F4 1F	RRA	
D2F5 CB19	RR	C
D2F7 47	LD	B,A
D2F8 21BF06	LD	HL,06BF
D2FB 11FF06	LD	DE,06FF
D2FE EDB8	LDDR	
D300 0640	LD	B,40
D302 3E20	LD	A,20
D304 13	INC	DE
D305 12	LD	(DE),A
D306 10FC	DJNZ	D304
D308 C1	POP	BC
D309 214E0E	LD	HL,0E4E
D30C 7E	LD	A,(HL)
D30D 91	SUB	C
D30E D8	RET	C
D30F 34	INC	(HL)
D310 C9	RET	

Az elmozgatott sorok számát 40H-val szorozva megkapjuk az elmozgatandó byte-ok számát, s ezzel elvégezzük az eltolást az ASCII képernyőn is

A felszabadított sor helyét 20H karakterekkel (betűköz) töltjük fel: a sor törlése

Ha volt CU pozíció megjegyzés: annak sorszáma Ha az elmozgatott sorok előtt van: kész, ha pedig alatta, akkor ezt is növelve helyreállítja a megjegyzett sor számát

Azt hiszem, jól érzékelhető a kedves Olvasó számára is, hogy a korrekt munka érdekében mennyi mindenre gondolni kell, s mennyi résztvevővel, ill. adminisztrációval jár még egy ilyen, aránylag egyszerű feladat elvégzése is.

Továbbmenve: a szerkesztési munka során sokszor válik szükségessé a képernyősorok felfelé mozgatása. Most ezzel a rutinnal ismerkedtünk meg. Ennek logikai mechanizmusa az, hogy definiálunk egy sorkiejtési funkciót, amely abban áll, hogy a megadott sort az alatta levő sorral írunk át, s ezt folytatjuk egészen a képernyő utolsó soráig.

Ezt a rutint használva a képernyősorok mozgatását tetszőleges sortól kezdhetjük, s pl. a teljes képernyő eltolása ebben a felfogásban az első sor kiejtését jelenti.

E19 A képernyő minden sorát felfelé mozgatja
D311

D311 3E01	LD	A,01	Az első sor kijelölése
D313 CD1ED3	CALL	D31E	Sorkiejtés
D316 011801	LD	BC,0118	A kurzor az utolsó sor
D319 ED43490E	LD	(0E49),BC	elejére áll
D31D C9	RET		és kész

Végül, utolsó teljes funkciót megvalósító rutinként az itt is felhasznált sorkiejtést ismerjük meg. Tapasztaltuk, hogy ezt a szubrutint az editor más funkcióját végrehajtó szubrutinok is többször hívták. Talán mondanom sem kell, hogy a felhasznált saját programjaiból is hívhatja. Természetesen ehhez biztosítani kell azt a struktúrát, amelyben a program nyújtotta szolgáltatások ténylegesen és praktikusán kihasználhatók.

E20	Sorkiejtés	
D31E	A megadott sortól kezdődően a képernyő aljáig minden sort eggyel felfelé mozgat	
	in: A=a kiejtendő sor száma	
D31E F5	PUSH AF	A sorszámot tárolja
D31F 4F	LD C,A	Megjegyzett kurzor pozíció
D320 214E0E	LD HL,0E4E	esetén ellenőrizzük, hogy a
D323 96	SUB (HL)	megjegyzett sor melyik
		tartományban van
D324 CC39D0	CALL Z,D039	Ha éppen ez a sor:
		érvénytelenítjük
D327 3001	JR NC,D32A	Ha feljebb van: rendben!
D329 35	DEC (HL)	Ha lejjebb van, a bejegyzést
		korrigáljuk
D32A CDE0D4	CALL D4E0	Sorkiejtés a képernyőn
D32D C1	POP BC	
D32E 78	LD A,B	A sorszám
D32F CD9FD3	CALL D39F	A sorjellemző címe,
D332 E5	PUSH HL	tároljuk
D333 68	LD L,B	
D334 2601	LD H,01	A kiejtendő sor elejének
D336 CD87D3	CALL D387	ASCII puffercíme
D339 EB	EX DE,HL	DE=a sor eleje: célcím
D33A 214000	LD HL,0040	
D33D 19	ADD HL,DE	HL=a következő sor eleje
		(kezdőcím)
D33E 3E18	LD A,18	
D340 90	SUB B	A mozgatandó sorok száma
D341 F5	PUSH AF	
D342 280B	JR Z,D34F	Ha eleve az utolsó sor: ugrás
D344 0E00	LD C,00	
D346 1F	RRA	A sorok számát szorozzuk 40H-
D347 CB19	RR C	val, így megkapjuk a
D349 1F	RRA	mozgatandó sorok számát
D34A CB19	RR C	
D34C 47	LD B,A	
D34D EDB0	LDIR	Sorkiejtés az ASCII-ban

D34F 0640	LD	B,40	
D351 2B	DEC	HL	Az utolsó sort feltölti
D352 3620	LD	(HL),20	20H (betűköz) karakterekkel:
D354 10FB	DJNZ	D351	sortörlés
D356 F1	POP	AF	A sorok száma
D357 4F	LD	C,A	
D358 E1	POP	HL	A kiejtett sor
D359 54	LD	B,H	sorjellemzőjének címe,
D35A 5D	LD	E,L	DE=célcím,
D35B 23	INC	HL	HL=kezdőcím
D35C 2802	JR	Z,D360	Ha az utolsó sor volt, most
			is ugrás
D35E EDB0	LDIR		Elmozgatjuk a sorjellemzőket
D360 AF	XOR	A	
D361 12	LD	(DE),A	Az utolsó sor szabad
D362 C9	RET		Kész

Ezt a szubrutin kategóriát két egyszerű, de sokszor hívott segédrutin zárja:

E21 A kurzort a következő sor elejére állítja
D363

D363 79	LD	A,C	A sorszám
D364 FE18	CP	18	Ha az utolsó sor: ugrás a
D366 28A9	JR	Z,D311	sorok elmozgatásához
D368 0C	INC	C	Egyébként veszi következő
D369 0601	LD	B,01	sor elejét,
D36B ED43490E	LD	(0E49),BC	"beiktatja"
D36F C9	RET		és kész.

A másik rövid szubrutin pedig a kurzort visszaállítja az aktuális bekezdés elejére.

E22 A kurzort a bekezdés elejére állítja
D370H

D370 ED4B490E	LD	BC,(0E49)	Az aktuális kurzorpozíció
D374 B7	OR	A	
D375 0D	DEC	C	Az előző sor
D376 79	LD	A,C	Ha még a képernyőn van,
D377 C49FD3	CALL	NZ,D39F	vesszük a sorjellemzőt
D37A 38F8	JR	C,D374	Ha még ez a sor is foglalt,
			tovább nézzük

D37C 0C	INC C	Ha az előző sor már üres, akkor ez a bekezdés első sora volt
D37D 0601	LD B,01	A sor elejére áll
D37F ED43490E	LD (OE49),BC	
D383 C9	RET	és kész

3.5.4.3 A segédrutinok

Ezután -- ugyan még hosszú ideig az editorban maradva -- az eddigieknél már alacsonyabb szintű rutinok következnek. Ebből a jellegből adódóan nem egy-egy komplex feladat megoldását adják, hanem a maguk szerény módján, bizonyos részfeladatok ellátásával segítik az eddig megismert "nagy" rutinok munkáját. Legtöbbjükkel -- ill. hívásukkal -- majd minden funkcionál találoztunk.

Most nézzük egyenként, mit és hogyan végeznek ezek az apróbb-nagyobb programok.

E23 D384	A kurzor koordinátáiból előállítja a hozzá tartozó ASCII puffercímet	
D384 2A490E	LD HL,(OE49)	A kurzor pozíciója
D387 25	DEC H	Áttér a 0-val kezdődő számolásra
D388 2D	DEC L	
D389 7C	LD A,H	A kurzor x koordinátája
D38A 87	ADD A,A	
D38B 87	ADD A,A	A=az x 4-szerese
D38C 65	LD H,L	A sorszám
D38D CB1C	RR H	

Ha most H-t egy 2 byte-os adat felső részének tekintjük, akkor benne a sorszám 80H-szorosa (*128) áll.

D38F 1F	RRA	H és A együtt: sor*80H + 2*oszlop
D390 CB1C	RR H	
D392 1F	RRA	Végül:
D393 6F	LD L,A	HL=sor*40H + oszlop

Ez már a puffer elejéhez viszonyított relatív cím, tehát már csak hozzá kell adni annak kezdőcímét, 0100H-t:

D394 7C	LD A,H	
D395 C601	ADD A,01	A felső byte növelésével
D397 67	LD H,A	0100H hozzáadása

D398 224B0E	LD	(0E4B),HL	Végül a kapott címet a
D39B C9	RET		megfelelő változóba írjuk

Látszik, hogy az editor ASCII képernyőjének 0100H kezdőcíme itt mennyire praktikus kihasználható volt: csak a felső byte-ot kellett eggyel növelni. (Ez persze egyszerűbben, egy közös INC H-val is elintézhető lett volna.)

E24 Sorjellemző előállítás
D39C

D39C 3A490E	LD	A,(0E49)	Az aktuális sorszám
-------------	----	----------	---------------------

Sokszor nem a D39CH belépési ponton, hanem a következő soron hívtuk ezt a rutint. Ekkor A-ban a vizsgálandó sorszámot kellett elhelyezni a hívás előtt.

D39F C64F	ADD	A,4F	A 24 sorjellemző címe:
D3A1 6F	LD	L,A	0E50...0E67H. Így 4F-hez a
D3A2 3E0E	LD	A,0E	sorszámot adva megkapjuk a
D3A4 CE00	ADC	A,00	cím alsó byte-ját,
D3A6 67	LD	H,A	a felső pedig 0EH
D3A7 7E	LD	A,(HL)	A kiolvasott sorjellemzőt
D3A8 07	RLCA		először ciklikusan balra
			forgatjuk, a
D3A9 B7	OR	A	carry-t töröljük, majd az A-
D3AA CB1F	RR	A	t a carry-n át jobbra
D3AC C9	RET		forgatva minden bit a helyére
			kerül. Kivéve a b7, mert ez
			0 lett, de a bit eredeti
			értéke a carry-ben van

Figyeljük meg a 3 soros megoldás szépségét!

Végül tehát HL a sorjellemző címére mutat. A carry bit jelzi, ha a sor teljesen tele van, tehát ez a tény a hívó programok számára azonnal vizsgálható. A-ban pedig a sor utolsó karakterének oszlop-
pozíciója maradt, a kizárólag jelző funkciót ellátó 7. bit törlődött.

3.5.4.4 A videomemória rutinok

Ezután három még alacsonyabb szintű rutin következik. Említettük, hogy az editor maga végzi a karakterek megjelenítését.

A háromféle üzemmódnak megfelelően három külön program gondoskodik a karakterek gyors kiírásáról. Itt sokkal kevesebb feltételt kell vizsgálni, ezért az összes tennivaló egyszerűbben megszervezhető, mint a megfelelő videorutinok esetében azt láttuk.

Az editor mindig a fix karakterpozíciókkal dolgozik. Ez már maga is jelentős egyszerűsödés: könnyebb a videomemória címek kézben tartása.

Másik jelentős körülmény, hogy az editor nem foglalkozik pl a karakterfelülírás kérdésével. Minden -- a képernyőn esetleg korábbról megmaradt -- grafikus és szöveges információt teljesen felülír.

E25 Karaktermegjelenítés 2-es üzemmódban
D3AD

in: B=PAPER színű vonalbyte
 C=INK színű vonalbyte
 HL=VID kezdőcím
 HL'=a karaktermátrix címe
 B'=a karakter tv-sorainak száma (0AH)

D3AD	114000	LD	DE,0040	Sorhossz a képernyőn
D3B0	D9	EXX		
D3B1	7E	LD	A,(HL)	A karakter egy sorának
D3B2	D9	EXX		bitképe
D3B3	A1	AND	C	
D3B4	A8	XOR	B	
D3B5	77	LD	(HL),A	Megjelenítés
D3B6	19	ADD	HL,DE	A következő sor VID címe
D3B7	D9	EXX		
D3B8	23	INC	HL	A karakter következő sora és
D3B9	10F6	DJNZ	D3B1	így tovább
D3BB	C9	RET		

E26 Karaktermegjelenítés 4-es üzemmódban
D3BCH

in: ugyanaz, mint 2-es üzemmódban (E25)

D3BC	D9	EXX		
D3BD	7E	LD	A,(HL)	A karakter egy sora
D3BE	D9	EXX		
D3BF	57	LD	D,A	
D3C0	0F	RRCA		Láthatóan egykönnyen nem
D3C1	0F	RRCA		követhető bitmanipulációk
D3C2	0F	RRCA		egész sorát bonyolítjuk itt
D3C3	0F	RRCA		le, amelyeknek részletes
D3C4	AA	XOR	D	értékelése oldalakat

D3C5 A1	AND C	venne igénybe. A lényeg
D3C6 5F	LD E,A	azonban az, hogy általuk a
D3C7 E60F	AND OF	karakter egy sorának 8 pontját
D3C9 AA	XOR D	igen rövid eljárással ketté
D3CA A1	AND C	osztjuk, s a kívánt formában
D3CB A8	XOR B	a VID két, egymást követő
D3CC 77	LD (HL),A	byte-jába töltjük. Javaslom,
D3CD 2C	INC L	az Olvasó esetleg elégítse ki
D3CE AB	XOR E	önállóan kíváncsiságát!
D3CF 77	LD (HL),A	
D3D0 113F00	LD DE,003F	
D3D3 19	ADD HL,DE	A VID következő sora
D3D4 D9	EXX	
D3D5 23	INC HL	A karakter következő sora és
D3D6 10E5	DJNZ D3BD	ismétlés 10 soron át
D3D8 C9	RET	

E27 Karaktermegjelenítés 16-os üzemmódban
D3D9H

in: ugyanaz, mint 2-es üzemmódban

Itt még hosszabb sorozatát találjuk a bitműveleteknek. Most végiggondoljuk a működést. Tudjuk, hogy ebben az üzemmódban a videomemória egy byte-ján két pontot tudunk ábrázolni. A bal oldali pontot ezen belül a b7-b5-b3-b1 bitek, a jobb oldalt a b6-b4-b2-b0 jelölik ki.

A karaktermátrix elemei viszont a megjelenítendő karakter egy sorát alkotó 8 pontot tárolnak. Amelyik bit értéke 1, a neki megfelelő pontot a képernyőn tintaszínnel ki kell gyújtani.

A C regiszterben rendelkezésünkre áll a tintaszínű vonalbyte, legyen ez most bitenként feltüntetve: 11111111, tehát két egymás melletti fehér színű pont.

A B regiszterben a papírszínű vonalbyte van, ez legyen most 00000000, tehát két fekete pont.

Kezdjük el:

D3D9 D9	EXX	
D3DA 7E	LD A, (HL)	A karakter egy sorának bitképe
D3DB D9	EXX	
D3DC 17	RLA	

Ezzel minden bit balra lépett, a legszélső 7. bit a carry-be került. Ez mutatja, hogy milyen az első pont.

D3DD 5F	LD E,A	Megőrizzük
D3DE 9F	SBC A,A	

Most A-ból önmagán kívül még a carry-t is kivontuk, így ha az első bit 0 volt, akkor most A=00, míg ha a kiléptetett pont bitje 1 volt, akkor most A=FF. Tehát A minden bitje olyan lett, mint a kiléptetett bit volt.

D3DF 57	LD	D,A	Az 1. pont szerinti 8 bitet tesszük D-be
D3E0 CB13	RL	E	Kiléptetjük a 2. pont bitjét
D3E2 9F	SBC	A,A	Úgy, mint az előbb: most is minden bit azonos a kiléptetetttel

Tehát az 1. pont szerinti bitek a D-ben, a 2. pont bitjével azonos bitek A-ban vannak.

```
D3E3 AA      XOR  D
```

Ha a két pont azonos, akkor D és A minden bitje is, tehát az eredmény: A=00. Különböző pontok esetén pedig az egyik regiszter: 000, a másik: FF, ezért XOR után A=FF.

Tehát most:

- D = az 1. pont bitjei
- A = 00 , ha a 2 pont azonos
- A = FF , ha a 2 pont különböző

D3E4 E655	AND	55	A=00: azonos pontok, A=55: különbözők
D3E6 AA	XOR	D	

Ha azonos pontok voltak, akkor most A minden bitje az első pontéval azonos, tehát 00, vagy FF. Ha azonban különbözők voltak, akkor az XOR után

-- ha az 1. pont bitje volt 1, úgy most az A: 10101010,
azaz A=0AAH lesz,
-- ha pedig a 2. pont volt egy, akkor az A: 01010101, azaz
A=55H marad.

A 8 bitet együtt nézve jól látszik, hogy a 2 pontot kijelölő biteknek megfelelően álltak be az A hozzájuk tartozó 4-4 bitjei is. A megjelenítési elvnek megfelelően, ha az A-t így a videomemóriába raknánk, akkor a karakter ábrázolt sora 2 bal szélső bitjeinek megfelelően:

- 00 esetén: két fekete
- 01 esetén: 1 fekete, 1 fehér
- 10 esetén: 1 fehér, 1 fekete
- 11 esetén: két fehér

pont jelenne meg a képernyőn.

Hogy ezek valójában milyen színnel tünnek fel, azt még természetesen a megadott INK és PAPER szín is befolyásolja.

D3E7	A1	AND	C	Az INK vonalbyte
D3E8	A8	XOR	B	A PAPER vonalbyte

Ezek a műveletek az előbb kiadódott biteket még úgy változtatják meg, hogy végeredményül a meghatározott színek jelenjenek meg a képernyőn -- de azt hiszem, számunkra most nem ez volt az igazán érdekes.

D3E9	77	LD	(HL),A	A két pont megjelent!
------	----	----	--------	-----------------------

A többi 6 pont sorsát ugyanígy lehet nyomon kísérni, az utasítások is ezután pontosan ugyanígy ismétlődnek. Természetesen nem ismétljük meg az előbbieket még 3-szor.

D3EA	2C	INC	L	Továbblépünk a videomemóriában
------	----	-----	---	--------------------------------

D3EB	CB13	RL	E
D3ED	9F	SBC	A,A
D3EE	57	LD	D,A
D3EF	CB13	RL	E
D3F1	9F	SBC	A,A
D3F2	AA	XOR	D
D3F3	E655	AND	55
D3F5	AA	XOR	D
D3F6	A1	AND	C
D3F7	A8	XOR	B
D3F8	77	LD	(HL),A

A 3. és 4. pont

D3F9	2C	INC	L	A következő VID-cím
------	----	-----	---	---------------------

D3FA	CB13	RL	E
D3FC	9F	SBC	A,A
D3FD	57	LD	D,A
D3FE	CB13	RL	E
D400	9F	SBC	A,A
D401	AA	XOR	D
D402	E655	AND	55
D404	AA	XOR	D
D405	A1	AND	C
D406	A8	XOR	B
D407	77	LD	(HL),A

Az 5. és 6. pont

D408	2C	INC	L	Az utolsó VID-cím
------	----	-----	---	-------------------

D409	CB13	RL	E
D40B	9F	SBC	A,A
D40C	57	LD	D,A
D40D	CB13	RL	E
D40F	9F	SBC	A,A
D410	AA	XOR	D

D411	E655	AND	55	
D413	AA	XOR	D	
D414	A1	AND	C	
D415	A8	XOR	B	
D416	77	LD	(HL),A	A 7. és 8. pont
D417	113D00	LD	DE,003D	Most csak ennyi a növekmény a következő sorhoz a videomemóriában
D41A	19	ADD	HL,DE	A következő sor kezdő VID-címe
D41B	D9	EXX		
D41C	23	INC	HL	A karaktermátrix következő sora
D41D	10BB	DJNZ	D3DA	Ez ismétlődik 10-szer és
D41F	C9	RET		utána kész

Nem volt minden célzás nélkül való, hogy az előbbi rutin 10 sorát másfél oldalon át elemeztük!

Egyrészt példát láttunk arra, hogy a legrejtettebb, az utasítások külső megjelenéséből nem nyilvánvaló működés is -- kellő türelemmel és elszántsággal -- nyomon követhető.

Másrészt az is meggyőzően látszik, hogy bár könyvünk célja a ROM programjának megismerése -- ez nem jelentheti minden esetben a működés minden részletre kiterjedő ismertetését. A több ezer soros program esetében ez tízezres oldalszámú könyvet eredményezne.

Így tehát meg kell elégednünk azzal a kompromisszummal, hogy az egész működés szempontjából fontos részeket igyekszünk kellő részletességgel megbeszélni, más esetekben viszont az Olvasó nagyfokú önálló aktivitására is szükség lesz.

E28 Karaktermegjelenítés az ASCII kód alapján
D420H

in: A=karakterkód
BC=a kurzor aktuális pozíciója

Ez egy igen jól használható kis szubrutin, könnyen paraméterezhető és bármely programból hívva tűrhetően gyors működést biztosít.

D420	08	EX	AF,AF	Tároljuk a kódot, amíg
D421	0D59D4	CALL	D459	előállítjuk a videomemória
D424	E5	PUSH	HL	címét
D425	08	EX	AF,AF	
D426	4F	LD	C,A	A kód megszorozva 10-zel,
D427	0600	LD	B,00	előállítjuk a hozzá tartozó 10
D429	69	LD	L,C	byte-os

D42A 60	LD H,B	karaktermátrixhoz való
D42B 29	ADD HL,HL	eltolást -- a megfelelő
D42C 29	ADD HL,HL	táblázatban
D42D 09	ADD HL,BC	Ez az eljárás pontosan
D42E 29	ADD HL,HL	megfelel annak, amit a
D42F CB79	BIT 7,C	videomeghajtónál láttunk
D431 0174C4	LD BC,C474	
D434 2803	JR Z,D439	Végül HL a kívánt
D436 014002	LD BC,0240	karaktermátrix első byte-jára
D439 09	ADD HL,BC	mutat
D43A 060A	LD B,0A	Ciklust szervezünk a karakter
		10 sorának megjelenítésére,
D43C D9	EXX	
D43D 3A960E	LD A,(0E96)	majd B-be és C-be betöltjük a
D440 47	LD B,A	PAPER és az INK színű
D441 3A950E	LD A,(0E95)	vonalybyte-okat
D444 4F	LD C,A	
D445 E1	POP HL	Elővesszük a VID-címet
D446 C3680E	JP 0E68	és ugrás!

Tudjuk, hogy hová. Az editor inicializáló rutinja minden esetben elhelyezi ezen a címen azt az ugró utasítást, amely az üzemmódnak megfelelő megjelenítő rutinnak adja át a vezérlést (ez valamelyik az előbb tárgyalt három közül). Az ezzel kapcsolatos részletek a 3.5.4.2 szakaszban megtalálhatók.

E28 A PAPER- és INK-színű vonalybyte-ok előállítása
D449H

A video eszközmeghajtó két szubrutinját hívja. Emellett már a megjelenítést is előkészíti azzal, hogy az INK vonalybyte-ot a PAPER byte-tal XOR kapcsolatba hozva teszi le a megfelelő belső munkaváltozóba. Ennek hasznát a megjelenítő rutinokban való egyszerű kezelhetőségben látjuk.

D449 08	EX AF,AF	A másodlagos A-val dolgozik
D44A CD05CC	CALL CC05	Előállítja a PAPER színű
D44D 32960E	LD (0E96),A	vonalybyte-ot, s
		tárolja.
D450 CD0BCC	CALL CC0B	Majd az INK színűt, de ennek
D453 A8	XOR B	tárolása előtt a PAPER byte-
D454 32950E	LD (0E95),A	tal XOR műveletet végez
D457 08	EX AF,AF	
D458 C9	RET	

Még egy ilyen nagyon elemi, előkészítő szubrutin van hátra. Ez a kurzor pozíciójának megfelelő kezdő videomemória címét állítja elő.

E30 VID-cím előállítása kurzorpozícióból
D459H

in: BC-ben a kurzor aktuális pozíciója

D459 05	DEC	B	A kurzor x koordinátája, 0-tól számozva
D45A 3A730B	LD	A, (0B73)	Ezt az üzemmódnak
D45D B7	OR	A	megfelelően (0,1,2),
D45E 2805	JR	Z, D465	megszorozva 1-gyel, 2-vel
D460 CB20	SLA	B	vagy 4-gyel: kapjuk a soron
D462 3D	DEC	A	belüli byte-sorszámot a
D463 20FB	JR	NZ, D460	videomemóriában. A sorszám: B-ben
D465 0D	DEC	C	A kurzor y koordinátáját is
D466 79	LD	A, C	0-tól számozzuk. Mivel egy
D467 87	ADD	A, A	karaktorsor 10 tv-sorból
D468 87	ADD	A, A	áll, ezért a sorszámot
D469 81	ADD	A, C	megszorozzuk 10-zel
D46A 87	ADD	A, A	
D46B 67	LD	H, A	Ez a H-ba téve 256-tal való
D46C AF	XOR	A	szorzást jelent
D46D CB3C	SRL	H	Egyelőre A alkotja az alsó
D46F 1F	RRA		byte-ot
			Osztunk 2-vel

Most tehát a H és A regiszterben együtt a sorszám 10-szerese áll, megszorozva még 128-cal: $H, A = 128 * (y * 10)$ (decimális értékek).

D470 37 SCF

Figyeljük meg, hogy ez az SCF egymagában megoldja a kiszámított relatív címhez a videomemória kezdőcímének, 8000H-nak a hozzáadását.

D471 CB1C RR H
D473 1F RRA

Most ui. nem csak a H és A újabb 2-vel való osztását értük el, hanem a H 7. bitjébe belépett a carry-be írt 1, ami önmagában 8000H-t jelent (H a kétbyte-os cím felső byte-ja). Tehát $H, A = 8000H + (sor * 0AH) * 40H$.

D474 B0 OR B Az alsó bitjeibe beírjuk a soron belüli pozíciót

D475 6F LD L,A

Ezután már $HL = 8000H + (sor*0AH)*40H + \text{oszlop-pozíció}$, azaz megvan a keresett videomemória cím.

D476 C9 RET

Ezután ismét olyan rutinok következnek, amelyek egy-egy komplex funkciót látnak el, az editor alapvető szerkesztő rutinjaihoz kapcsolódva, azokat kiszolgálva, a videomemóriában végzik el az aktuális feladatokat.

Helyes paraméterezés esetén ezek is hívhatók a felhasználói programokból, s átfogva a képernyőmanipulációs feladatok nagy részét, hatékonyan segíthetik a felhasználó munkáját.

E31 A kurzor helyén álló karakter mentése
D477H

Ez a rutin a kurzor pozíciójának megfelelő hely tartalmát a képernyőről (videomemóriából) a saját munkaterületére menti. Az editor kurzor rutinja használja.

Érdekes megoldás, hogy nem a kurzor helyén álló karakter kódját, hanem magát a videomemória-részt másoljuk ki és állítjuk vissza a kurzor villogtatása közben, ami azzal is jár, hogy a kurzor nem rontja el azt a grafikus képtartalmat sem, ahová éppen kerül.

D477 CD59D4	CALL D459	Előállítja HL-ben a kezdő videomemória címet
D47A 116DOE	LD DE,OE6D	A mentési puffer eleje
D47D 3A6COE	LD A,(OE6C)	1 karakter szélessége,
D480 4F	LD C,A	egymás mellett ennyi byte-ot kell kimásolni a VID-ből,
D481 060A	LD B,0A	soronként
D483 C5	PUSH BC	10 sorból áll a karakter
D484 E5	PUSH HL	
D485 0600	LD B,00	A kezdő VID-cím
D487 EDB0	LD1R	Egy sor átmásolása
D489 E1	POP HL	
D48A OE40	LD C,40	A következő sor kezdőcímére állunk
D48C 09	ADD HL,BC	
D48D C1	POP BC	
D48E 10F3	DJNZ D483	és ezt ismételtetjük
D490 C9	RET	

A következő szubrutin dolga az előzőnek éppen a fordítottja.

E32 D491H	A kurzor pozíciójának megfelelő terület képernyőtartalmának visszaállítása	
D491 CD59D4	CALL D459	A kezdő videomemória cím
D494 116D0E	LD DE,0E6D	A kimentett terület pufférének kezdőcíme
D497 3A6C0E	LD A,(0E6C)	Egy karakter szélessége, ennyi byte-ot kell visszatenni egymás mellé a videomemóriába, tehát 10 ilyen egymás alatti sor tesz ki egy karaktert
D49A 4F	LD C,A	
D49B 060A	LD B,0A	A ciklusfej
D49D EB	EX DE,HL	
D49E C5	PUSH BC	
D49F D5	PUSH DE	
D4A0 0600	LD E,00	Egy sor visszaállítása
D4A2 EDB0	LDIR	
D4A4 EB	EX DE,HL	
D4A5 E1	POP HL	
D4A6 0E40	LD C,40	A következő sor kezdő videomemória címe
D4A8 09	ADD HL,BC	
D4A9 C1	POP BC	
D4AA 10F1	DJNZ D49D	Ez megy 10 soron át és kész
D4AC C9	RET	

A következő szubrutin az editor karakter- és sorbeszúrás, ill. karakter- és sortörlés funkcióit kísérik a képernyőn.

E33 D4ADH	Törli a kurzor sorát	
D4AD CD59D4	CALL D459	A kurzorhoz tartozó VID cím HL-ben
D4B0 3A960E	LD A,(0E96)	PAPER színű vonalbyte, tároljuk
D4B3 08	EX AF,AF	
D4B4 3E3F	LD A,3F	Kivonással és a felső bitek törlésével előállítjuk a sor végétől való távolságot
D4B6 95	SUB L	
D4B7 E63F	AND 3F	

D4B9 2818	JR	Z, D4D3	Ha a kurzor a sor végén áll, akkor ugrás a következő szubrutinra
D4BB 4F	LD	C, A	A távolság: a törlendő byte- szám
D4BC 7D	LD	A, L	
D4BD 060A	LD	B, 0A	
D4BF E63F	AND	3F	Az A alsó 6 bitjén az eredeti pozíció, a felső két bitbe az aktuális sorszámhoz tartozó érték
D4C1 B5	OR	L	
D4C2 6F	LD	L, A	A mindenkori kezdő cím a sorban
D4C3 08	EX	AF, AF	A=PAPER színű vonalbyte
D4C4 77	LD	(HL), A	Egy byte törlése
D4C5 54	LD	D, H	HL=kezdőcím,
D4C6 5D	LD	E, L	DE=HL+1, célcím,
D4C7 1C	INC	E	
D4C8 C5	PUSH	BC	
D4C9 0600	LD	B, 00	
D4CB EDB0	LDIR		az egész sorban elvégzi a törlést
D4CD C1	POP	BC	
D4CE EB	EX	DE, HL	A következő sor eleje
D4CF 08	EX	AF, AF	A-ban ismét a pozíció
D4D0 10ED	DJNZ	D4BF	Ismétlés 10 soron át
D4D2 C9	RET		

E34 Törli a megadott VID kezdőcímű karakter helyét a
D4D3H képernyőn
in: HL=a kezdő videomemória cím

D4D3 114000	LD	DE, 0040	Sorhossz a VID-ben
D4D6 060A	LD	B, 0A	A karakter 10 soros
D4D8 3A960E	LD	A, (0E96)	A PAPER színű vonalbyte
D4DB 77	LD	(HL), A	Ezzel elvégzi a törlést,
D4DC 19	ADD	HL, DE	veszi a következő sort,
D4DD 10FC	DJNZ	D4DB	ismétlés 10-szer és
D4DF C9	RET		kész

E35 Sorkiejtés a képernyőn
D4E0H
in: C=a sorszám (karakter sor, tehát 1...24)

D4E0 79	LD	A, C	A megadott sorszám
D4E1 FE18	CP	18	Az alsó sor?

D4E3 2814	JR	Z, D4F9	Ha igen: csak ezt kell törölni, ugrás előre
D4E5 0601	LD	B, 01	A sor elejére állunk és vesszük ennek VID-címét
D4E7 CD59D4	CALL	D459	Ez lesz a célcím DE-ben
D4EA EB	EX	DE, HL	HL=az utolsó karaktorsor első tv-sorának eleje,
D4EB 2180B9	LD	HL, B980	
D4EE B7	OR	A	
D4EF ED52	SBC	HL, DE	a mozgatandó byte-hossz
D4F1 44	LD	B, H	
D4F2 4D	LD	C, L	
D4F3 218002	LD	HL, 0280	A megadott alatti karaktorsor első tv-sora
D4F6 19	ADD	HL, DE	
D4F7 EDB0	LDIR		Az egészet egyszerre elmozgatja

Most már csak az alsó karaktersort kell törölni.

D4F9 2180B9	LD	HL, B980	Az első tv-sor elejétől,
D4FC 1181B9	LD	DE, B981	egyenként minden byte-ba
D4FF 017F02	LD	BC, 027F	ilyen hosszán
D502 3A960E	LD	A, (0E96)	betöltjük a PAPER színű
D505 77	LD	(HL), A	vonalybyte-ot
D506 EDB0	LDIR		
D508 C9	RET		

A fenti rutin tehát egyszerű sorreltolást hajt végre a képernyőn, felfelé.

Ezzel szemben lefelé mozgatja a képernyősorokat -- sorbeszúrással -- a következő:

E36 Sorbeszúrással a képernyőn
D509H in: C=sorszám (1...24)

D509 3E18	LD	A, 18	Az utolsó sor?
D50B 91	SUB	C	Ha igen: csak ezt kell törölni, ugrás vissza
D50C 28EB	JR	Z, D4F9	Ez tehát a mozgatandó sorok száma
D50E 4F	LD	C, A	
D50F 87	ADD	A, A	
D510 87	ADD	A, A	Először megszorozzuk 10-zel,
D511 81	ADD	A, C	hiszen egy karaktorsor 10 tv-
D512 87	ADD	A, A	sorból áll

D513 47	LD B,A	Azután a szokásos módon 256-
D51F 11FFBB	LD DE,BBFF	tal szorozva, majd egymás után
D515 CB38	SRL B	kétszer 2-vel osztva (ami
D517 1F	RRA	egyenértékű a 64-gyel való
D518 CB38	SRL B	szorzással) megkapjuk az
D51A 1F	RRA	összes mozgató byte számát
D51B 4F	LD C,A	
D51C 217FB9	LD HL,B97F	Az utolsó előtti és
D51F 11FFBB	LD DE,BBFF	az utolsó karaktersor utolsó
D522 EDB8	LDDR	tv-sorának vége, s az
D524 23	INC HL	elmozgatás
D525 54	LD D,H	Végül a szabaddá tett sor
D526 5D	LD E,L	elejére állítjuk HL-t, mellé
D527 13	INC DE	DE-t, s egy visszaugrással
D528 18D5	JR D4FF	töröljük a karaktersort

E37 Egy karakter elmozgatása a képernyőn

D52AH

in: HL=a karakter kezdőcíme a videomemóriában
DE=a végcím a VID-ben

D52A 3A60E	LD A,(0E6C)	A karakter szélessége byte-
		okban (01...04)
D52D 4F	LD C,A	
D52E 060A	LD B,0A	Egy karakter: 10 tv-sor
D530 C5	PUSH BC	
D531 E5	PUSH HL	A kezdőcím,
D532 D5	PUSH DE	célcím és
D533 0600	LD B,00	BC=karakterszélesség alapján
D535 EDB0	LDIR	elmozgatás
D537 0E40	LD C,40	
D539 E1	POP HL	
D53A 09	ADD HL,BC	A következő célcím
D53B EB	EX DE,HL	DE-be,
D53C E1	POP HL	majd:
D53D 09	ADD HL,BC	a következő kezdőcímre HL-t
		állítjuk, vesszük a ciklus- és
D53E C1	POP BC	hosszparamétereket és ezt
D53F 10EF	DJNZ D530	ismételjük 10 tv-soron át
D541 C9	RET	

Ez a szubrutin is jelentős gyakorisággal használható a felhasználói programokban és természetesen nem csak karakterek mozgatására. Az első 3 sort elhagyva, tehát a 0D52FH vagy 0D530H címen belépve a B és a C regiszter is tetszőleges értékkel betölthető, ami már fölöttébb változó méretű területek mozgatását teszi lehetővé!

E38 Egy karaktorsor mozgatása balra, megadott
D542H pozíciótól

in: A=a mozgatandó karakterek száma, Z=1: ha 00
B=a kezdő pozíció a sorban
C=a sorszám
HL=a sorjellemező címe

D542	281E	JR	Z, D562	Ha semmit nem kell mozgatni: ugrás előre
D544	E5	PUSH	HL	Egyelőre tároljuk az
D545	C5	PUSH	BC	adatokat
D546	5F	LD	E, A	
D547	3A730B	LD	A, (0B73)	A mozgatandó karakterek
D54A	B7	OR	A	számából és az üzemmódból,
D54B	2805	JR	Z, D552	megfelelő szorzások révén
D54D	CB23	SLA	E	(*1, *2, *4)
D54F	3D	DEC	A	megkapjuk a mozgatandó
D550	20FB	JR	NZ, D54D	byte-ok számát
D552	CD59D4	CALL	D459	A sor- és oszlop-pozíció
D555	4B	LD	C, E	alapján HL-be a kezdő
D556	54	LD	D, H	videomemória cím,
D557	3A6C0E	LD	A, (0E6C)	C-be a mozgatandó hossz,
D55A	85	ADD	A, L	DE-be pedig a HL-től egy
D55B	5F	LD	E, A	karakter szélességével
D55C	EB	EX	DE, HL	nagyobb memóriacím kerül,
D55D	CD2ED5	CALL	D52E	ami tehát a kiválasztott
D560	C1	POP	BC	karakterrel jobbról
D561	E1	POP	HL	szomszédos karakter kezdő
D562	46	LD	B, (HL)	VID-címe
D563	CB78	BIT	7, B	Így beállítva a
D565	2811	JR	Z, D578	paramétereket, meghívjuk az
				előbbi átmozgató rutint
				Ezután vesszük az eredeti
				pozíciót és a sorjellemezőt
				A sor tele volt?
				Ha nem: csak az utolsó
				helyet kell törölni, ugrás
				a következő rutinra

Ha a sor foglalt volt (tele volt), akkor a következő sor első karakterét át kell hozni ennek a sornak a felszabadított végére.

D567	C5	PUSH	BC	
D568	CB88	RES	7, B	Vesszük az utolsó pozíció
D56A	CD59D4	CALL	D459	videomemória címét, ezt DE-
D56D	EB	EX	DE, HL	ben helyezük el

D56E C1	POP	BC	Ezután előállítjuk
D56F 0C	INC	C	a következő sor
D570 0601	LD	B, 01	első karakterének
D572 CD59D4	CALL	D459	VID címét HL-ben, s végül
D575 C32AD5	JP	D52A	ezekkel végrehajtjuk az előző szubrutint, amely a kívánt karakter átmozgatást hajtja végre

Egyszerű szubrutin következik ezután: egyetlen karakter helyét törli a képernyőn.

E39 Karakter törlése a képernyőn
D578H

in: B=a karaktorsoron belüli pozíció
C=a sorszám

D578 CD59D4	CALL	D459	Vesszük az adott pozícióhoz tartozó videomemória címet
D57B 0E0A	LD	C, 0A	10 soros ciklus lesz
D57D 114000	LD	DE, 0040	A sorhossz a VID-ben
D580 3A6C0E	LD	A, (0E6C)	Egy karakter szélessége, byte-ban (1, 2, 4)
D583 47	LD	B, A	Ennyi egymás melletti byte-ot kell törölni
D584 3A960E	LD	A, (0E96)	A PAPER színű vonalbyte
D587 E5	PUSH	HL	
D588 77	LD	(HL), A	Törlés, a karakter
D589 23	INC	HL	szélességének megfelelő
D58A 10FC	DJNZ	D588	hosszon
D58C E1	POP	HL	A kezdő VID-címből a következő
D58D 19	ADD	HL, DE	soron belüli kezdőcím
D58E 0D	DEC	C	
D58F 20EF	JR	NZ, D580	Ha még nincs kész a 10 sor: ugrás vissza,
D591 C9	RET		egyébként kész

És ezzel, kedves Olvasó elérkeztünk az editor utolsó rutinjához. Az előzőekhez hasonlóan ez is közvetlenül a videomemóriában dolgozva támogatja az editor munkáját, de természetesen minden leleményes felhasználó munkáját is.

E40 A karaktorsor mozgatása jobbra, megadott soron
D592H belüli pozíciótól kezdve

in: Z=1, ha a megadott pozíciót törölni kell,
egyébként annak helyére az előző sor utolsó
karaktere kerül

B=a soron belüli kezdőpozíció

C=a sorszám

HL=a sorjellemző címe

D592 F5	PUSH AF	A Z flaget tároljuk
D593 7E	LD A, (HL)	Nézzük a sorjellemzőt,
D594 E67F	AND 7F	de a 7. bit nem kell (foglaltság)
D596 90	SUB B	A megadott pozíció az utolsó írt karakter a sorban?
D597 2835	JR Z, D5CE	Ha igen: ugrás előre, mozgatás nem lesz
D599 5F	LD E, A	A mozgatandó karakterek
D59A 3A730E	LD A, (0E73)	számát az üzemmódnak
D59D B7	OR A	megfelelően szorozva
D59E 2805	JR Z, D5A5	(*1, *2, *4)
D5A0 CB23	SLA E	megkapjuk a mozgatandó
D5A2 3D	DEC A	byte-ok számát
D5A3 20FB	JR NZ, D5A0	
D5A5 E5	PUSH HL	
D5A6 C5	PUSH BC	
D5A7 46	LD B, (HL)	A sorjellemző alapján
D5A8 CB88	RES 7, B	előállítjuk a soron belüli
D5AA CD59D4	CALL D459	utolsó karakter kezdő VID- címét (HL),
D5AD 2B	DEC HL	
D5AE 4B	LD C, E	BC-ben a mozgatandó byte-ok
D5AF 0600	LD B, 00	számát,
D5B1 EB	EX DE, HL	a karakterek szélessége
D5B2 3A6C0E	LD A, (0E6C)	alapján pedig a következő
D5B5 6F	LD L, A	karakter kezdő videomemória
D5B6 60	LD H, B	címét
D5B7 19	ADD HL, DE	
D5B8 EB	EX DE, HL	
D5B9 060A	LD B, 0A	A karaktorsor 10 tv-sorból
D5BB C5	PUSH BC	áll
D5BC E5	PUSH HL	HL=a kezdőcím
D5BD D5	PUSH DE	DE=a célcím
D5BE 0600	LD B, 00	BC=a hossz, ezekkel egy sor
D5C0 EDB8	LDDR	elmozgatása
D5C2 0E40	LD C, 40	
D5C4 E1	POP HL	A célcím a következő tv-
D5C5 09	ADD HL, BC	sorban

D5C6 EB	EX	DE,HL	A címet áttöltjük DE-be, HL-
D5C7 E1	POP	HL	be pedig a következő sorbeli
D5C8 09	ADD	HL,BC	kezdőcím kerül
D5C9 C1	POP	BC	Ezt ismételjük 10 tv-soron át
D5CA 10EF	DJNZ	D5BB	

Eddig tehát már megtörtént a soron belüli elmozgatás. Azt kell még tisztázni, hogy mi történjen az eredeti pozícióval: a régi tartalom nyilvánvalóan ott nem maradhat (az elmozgatás után tőle jobbra ugyanaz a karakter van).

D5CC C1	POP	BC	Vesszük tehát az eredeti
D5CD E1	POP	HL	pozíciót, sorjellemzőt és a
D5CE F1	POP	AF	hozott Z jelzőbitet
D5CF 28A7	JR	Z,D578	Ha Z=1, akkor az eredeti
			helyet töröljük
D5D1 EB	EX	DE,HL	Egyébként az aktuális
D5D2 C5	PUSH	BC	sorjellemző DE-be kerül és HL-
D5D3 CD59D4	CALL	D459	ben kiszámítjuk az eredeti
			hely kezdő VID-címét
D5D6 C1	POP	BC	
D5D7 EB	EX	DE,HL	Ezután DE-ben az aktuális VID-
			cím
D5D8 0D	DEC	C	Ráállunk az előző sorra és a
D5D9 2B	DEC	HL	sorjellemzőjére, majd ennek
D5DA 46	LD	B,(HL)	alapján kiszámítjuk a sor
D5DB CBB8	RES	7,B	utolsó karakterének kezdő
D5DD CD59D4	CALL	D459	VID-címét
D5E0 C32AD5	JP	D52A	Az E37 szubrutinnal fejezzük
			be, mely most az előző sor
			végéről másol át ide egy
			karaktert

Megjegyezzük, hogy ebben az utolsó esetben az eredeti pozíció éppen az aktuális sor első karaktere (az INS rutinban), tehát az előző sor végének áthozatala teljesen indokolt.

Ezzel kész! Ennyit tud az editor.

Pontosabban ezeket a rutinokat felhasználva lehet esetleg a TVC editoránál többet szövegkezelő programokat is fejleszteni. Ne felejtsük el, hogy a TV-Computer editora elsősorban a BASIC parancsok és programok -- tehát egyáltalán ennek a BASIC-orientált környezetnek és az ebben adódó szituációknak -- kiszolgálására készült.

3.5.5 A billentyűzet

Új és legalább ekkora jelentőségű részhez érkeztünk. A következő pár oldalon a billentyűzetkezelő rutinokkal ismerkedünk meg.

Azt hiszem, az Olvasó, aki eddig igyekezett követni a könyv eddigi gondolatmenetét, nagyon jól kezdi érezni, hogy "mire megy ki a játék".

Lassan észrevehetően összegyűlik minden ahhoz, hogy a számítógépet magasabb szinten használni tudjuk. Mögöttünk van máris egy sereg olyan fontos funkciót ellátó szubrutin, amelyekre való egyszerű hivatkozásokkal nagyon alapvető, nagyon komplex és nagyon fontos problémák intézhetők el ezután, egyetlen sorban.

A lehetőségeink azonban még mindig nem teljeseek.

3.5.5.1 A billentyűzetkezelés technikája

A billentyűzetet kiszolgáló különböző szintű rutinokhoz szintén egy ugrótáblán keresztül juthatunk el. Az itt elhelyezett címeken természetesen átfogóbb jellegű feladatok megoldása folyik, amelyek megvalósítását néhány apróbb szubrutin támogatja - úgy mint eddig.

A billentyűzetkezelő rutinok azonban az eddigieknél jóval egyszerűbb logikai struktúrát mutatnak. Amíg a video vagy az editor összességében nagyon sokféle feladatnak kellett, hogy megfeleljen, s a szerteágazó, összetett funkciókat több tucat egyenrangú vagy egymásnak alárendelt programrész együttműködése valósíthatta meg, addig a billentyűzet esetében egyetlen fő funkció köré csoportosul az összes részfeladat.

A megszakításkezelő alprogram megismerésekor azt mondtuk, hogy az a gép legjobban futtatott programja. De azt is láttuk, hogy az egyes eszközök kiszolgálásában megkülönböztetett jelentősége van a billentyűzetnek. A megszakításkezelő program minden alkalommal lefuttatja a 0. funkciószerű billentyűzetrutint, amely

- letapogatja a teljes billentyűzetet,
- előállítja a lenyomott gombok kódját,
- vezérli a tartósan lenyomott gombok funkcióismétlését, s
- még néhány egyéb, járulékos funkciót is ellát.

Ám a billentyűzettel kapcsolatban valójában ez az egyetlen, ténylegesen gyakorlati munkával is járó nagy feladat. Látni fogjuk, hogy a többi ide tartozó szubrutin vagy ezt az egyet szolgálja ki, vagy pedig egyszerű adminisztratív jellegű munkát végez.

A nagyon egyszerű logikai struktúra természetesen a tényleges felépítményre is hatással van: ez az egyetlen szubrutin teszi ki a billentyűzettel kapcsolatos ROM-terület jelentős részét. Nem beszélve itt azokról a táblázatokról, amelyekből a billentyűzet különböző LOCK állapotaiban a kódok kiolvasása történik -- hiszen ezek a ROM elég nagy területét foglalják el ugyan, de nem programok.

3.5.5.2 Az ugrótábla

A billentyűzethez szóló funkcióhívások ugrótáblája:

ROM	Tartalom	Jelentés
D5E3	04	A hívható funkciók száma
D5E4	D62D	0. funkció (KBD-INT)
D5E6	D618	1. funkció (KBD-CHIN)
D5E8	D62C	2. funkció (NOP)
D5EA	D612	3. funkció (KBD-STAT)

3.5.5.3 A billentyűzet rutinjai

Az első, egyszerű kis program az inicializálás része. Bekapcsoláskor, vagy a RESET gomb megnyomása után a billentyűzet munkaterületének alaphelyzetbe állítására szolgál. Utána pedig már a funkcióhívások lebonyolítása következik.

KEY-INIC A billentyűzet inicializálása
D5ECH

D5EC 3E1E	LD	A,1E	A DELAY-KEY, a funkció
D5EE 32650B	LD	(0B65),A	ismétlés késleltetésének
			alapértéke
D5F1 3E03	LD	A,03	RATE-KEY, a tartósan
D5F3 32670B	LD	(0B67),A	lenyomott billentyű
			funkcióismétlésének
D5F6 AF	XOR	A	gyakorisága
D5F7 32660B	LD	(0B66),A	LOCK-KEY, alaphelyzet

D5FA 21510B	LD HL,0B51	A PICTURE és az OLDPIC
D5FD 11520B	LD DE,0B52	10-10 byte-os
D600 011300	LD BC,0013	billentyűzetmátrixok
D603 77	LD (HL),A	nullázása
D604 EDB0	LDIR	(nincs lenyomott gomb)
D606 21E50B	LD HL,0BE5	Végül a 0BE5...0BEEH
D609 11E60B	LD DE,0BE6	munkaterület nullázása
D60C 0E09	LD C,09	
D60E 77	LD (HL),A	
D60F EDB0	LDIR	
D611 C9	RET	Az inicializálás kész

BILL. 03 KBD-STAT (keyboard status)
93H A billentyűzet állapotát adja vissza, arról
 informál, hogy van-e lenyomva valamilyen
 billentyű

in: -

out: C=00 nincs lenyomott gomb
 FF van, a karakter kódja beolvasható
 A=00 hibátlan működés jelzése

D612 3AE50B	LD A,(0BE5)	Ez a változó éppen a kérdezett állapotot mutatja
D615 4F	LD C,A	Értékét C-be tesszük,
D616 AF	XOR A	"rendben" jelzés és
D617 C9	RET	kész

Csak pár sorral hosszabb a következő.

BILL 01 KBD-CHIN (keyboard character input)
91H Vár egy billentyű lenyomására, majd visszaadja
 annak kódját

in: -

out: C=a lenyomott billentyű ASCII kódja
 A=00 rendben
 F5 CTRL+ESC együttes lenyomása

D618 21E50B	LD HL,0BE5	A billentyűk állapota
D61B 7E	LD A,(HL)	Van lenyomott billentyű?
D61C B7	OR A	(00: nincs)
D61D 28FC	JR Z,D61B	Ha nincs: ugrás vissza!

Ez nem végtelen hurok, a program nem ragad itt le, ui. a megszakítások alatt futó billentyűzetkezelő program (0.funkció) átírja ezt a változót, ha valamilyen gombot lenyomva talál.

D61F 3AE90B	LD	A, (0BE9)	Ha van lenyomott billentyű, akkor annak kódját a rendszer ide rakja le
D622 4F	LD	C,A	Betölti C-be és kész, csak még a CTRL+ESC-t kell megnézni
D623 3600	LD	(HL), 00	A kiolvasást mindenesetre nyugtázzuk
D625 3A160B	LD	A, (0B16)	STOP-FLAG, itt történik a CTRL+ESC adminisztrálása
D628 B7	OR	A	00?
D629 C8	RET	Z	Ha igen: ezzel visszatér
D62A 3EF5	LD	A, F5	Egyébként beállítja a hibakódot és ezt adja vissza
D62C C9	RET		

Látható, hogy ezeknek a funkcióhívásoknak a működése teljes egészében a fő munkarutinra épül.

A billentyűzettel kapcsolatban nincs a blokk input-output műveletnek megfelelő funkció. Az ugrótáblába beírt névből is látszik (NOP), hogy ehhez semmiféle tevékenység nem fűződik. A táblába ugyanakkor be kellett kerülnie, mert logikailag a 3-as funkciónak kell ellátnia az állapotfigyelést (a 2-es a blokkműveletekre van fenntartva). Ezért a 2-es funkcióhoz valamit rendelni kellett. Ez jelen esetben a 0D62CH címen levő egyetlen RET utasítás.

Most kemény munkához látunk!

Előttünk a 0. funkcióhívás rutinja. Tehát az, amelyre minden megszakítás közben sor kerül, s amelynek működéséről a felhasználónak általános esetben tudomása sem kell, hogy legyen.

Természetesen tudjuk, hogy az operációs rendszer IT kezelő programja ezen kívül még számos egyéb fontos feladatot is ellát. Köztük azonban a billentyűzet kiszolgálása a gép és az ember közötti kapcsolattartás legalapvetőbb eszköze.

Nagy elemző munkára készülünk fel!

BILL 00	KBD-INT
90H	A billentyűzet leolvasása

D62D 3A660B	LD	A, (0B66)	LOCK-KEY, a LOCK állapot
D630 E60B	AND	0B	00001011, tehát csak a b3, b1, b0 bit kell

A 00 érték alapállapotot jelent, míg b0=1 CTRL, b1=1 a SHIFT, b3=1 pedig az ALT-LOCK állapotot jelzi.

D632 47	LD	B,A	Ez a három egyszerű utasítás
D633 ED44	NEG		helyrehozza az esetleg
D635 A0	AND	B	ezeektől eltérő bejegyzéseket

Hatásuk eredménye az, hogy a 8 biten a jobb széléről balra haladva csak az első 1 értékű bit marad meg, a többi 0 lesz, a LOCK-állapot tehát teljesen egyértelművé válik.

D636 32660B	LD	(0B66),A	Ezt teszi vissza
D639 CDA5D7	CALL	D7A5	Leolvassa a billentyűzetet és az eredményt beírja a PICTURE mátrixba,
			IY ennek elejére mutat
D63C 21E70B	LD	HL,0BE7	Itt külön nyilvántartja a LOCK billentyű állapotát
D63F FDCE066E	BIT	5,(IY+06)	A mátrix 6. sorának 5. eleme: LOCK
D643 280D	JR	Z,D652	Ha nincs lenyomva: ugrás

A következő sorokban kitüntetett figyelemmel kezeljük a LOCK billentyű jelentését, mivel ez a további munkánkat, a kód előállítását teljesen meghatározza.

D645 7E	LD	A,(HL)	Ezelőtt is le volt nyomva a LOCK?
D646 B7	OR	A	Ha igen, akkor addig nem csinál semmit, amíg fel
D647 C0	RET	NZ	nem engedik!

Ez egy nagyon lényeges kérdés. Mivel a kódok előállítása ezen múlik, a felhasználó szándékát egyértelművé kell tenni. Ha egyszer a LOCK-ot lenyomjuk, akkor ez változtatást jelent, s ez a beállítási állapot addig tart, amíg a felhasználó fel nem engedi a LOCK nyomógombot. De addig minden más feldolgozás szünetel! A 0BE7H változó FF értéke jelzi, hogy LOCK-beállítási folyamat van.

Ne felejtsük el, hogy ezt a programot a számítógép átlagosan másodpercenként 50-szer futja át, tehát valószínű, hogy egyszer lenyomva a LOCK-ot, a gép azt hosszú időn át lenyomva találja -- az ő belső időszámítása szerint.

D648 35	DEC	(HL)	Ha viszont korábban nem volt lenyomva, akkor xxxxxxxxxxxx (olvashatatlan sor)
---------	-----	------	--

D649 3E01	LD A,01	Beolvassa a PICTURE-ból
D64B CD90D7	CALL D790	a LOCK-kal együtt lenyomott
D64E 32660B	LD (0BE6),A	funkcióbillentyűket és
D651 C0	RET NZ	tárolja LOCK-KEY-ben, amint
		azt már leírtuk
		Ha ez nem alapállapot, akkor
		vissza is tér

Ilyenkor tehát a leolvasás egyetlen eredménye az új LOCK állapot beállítása. Ha azonban a LOCK-ot magában nyomták le, akkor továbbmegy, s a LOCK "beállítás alatt" jelzést is érvényteleníti.

D652 70	LD (HL),B	B=00, nem LOCK beállítás
---------	-----------	--------------------------

Ezután kezdődik el a beolvasott billentyűmátrix értékelése.

D653 3E04	LD A,04	Megvizsgálja a betűváltó
D655 CD90D7	CALL D790	billentyűk állapotát, s az
D658 32E80B	LD (0BE8),A	eredményt tárolja

Az eredmény jelentése:

A=08	ALT
A=04	CTRL
A=02	SHIFT
A=00	egyik sem

A pillanatnyi állapot ismerete azért fontos, mert a betűváltó billentyűknek az egyes LOCK állapotokban visszaváltó hatásuk van, amit majd figyelembe kell venni.

D65B AF	XOR A	Z=0, ez inputként kell a
D65C CDC7D6	CALL D6C7	következő rutinhíváshoz
		Előállítja az új billentyű
		kódját

Ez a szubrutin -- amint majd később látni fogjuk:

- E=FF-et ad vissza, ha nem volt új billentyű lenyomva a korábbi állapothoz képest, vagy CTRL+P volt,
- HL=0BE6H a CTRL+P lenyomását mutatja, míg
- elfogadott új billentyű esetén C-ben már a kódot adja vissza

D65F 1C	INC E	Van visszaadható kód?
D660 2810	JR Z,D672	Ha nincs: ugrás előre!

Az az ág, ahová ugrani kell, a további feldolgozás jelentős részét képezi, hiszen a mondott okokból csak ritkán van új kód, legtöbbször a leolvasás eredménye egy korábbi állapot ismétlése.

Az új kód átadása a következőképpen bonyolódik le:

D662 79	LD	A,C	Leteszi az előállított
D663 32E90B	LD	(0BE9),A	billentyű kódját,
D666 3EFF	LD	A,FF	jelzi, hogy az új kód
D668 32E50B	LD	(0BE5),A	kiolvasható,
D66B 3A650B	LD	A,(0B65)	a DELAY-KEY késleltetési
D66E 32EA0B	LD	(0BEA),A	paraméter alapértékével
D671 C9	RET		inicializálja a saját
			számlálóját és visszatér

Az új billentyűkód kiadása után a tartósan lenyomott billentyű funkciókésleltetése természetesen újratezdódik.

Nézzük azonban a gyakoribb esetet, amikor nem egy új billentyű lenyomásának feldolgozásáról van szó.

D672 7E	LD	A,(HL)	CTRL+P állapot van?
D673 E7	OR	A	(FF:igen)
D674 20E6	JR	NZ,D65C	Ha igen: ugrás vissza!

A 0D65CH címről meghívott szubrutin most a billentyűzet ismételt leolvasásával végzi el a munkát. Amíg valamelyik billentyűt le nem nyomták, addig a program az eddig leírt beolvasási-értékelési hurokban marad.

Ez a CTRL+P lenyomásával létrehozott HOLD (feltartás, berekesztés) állapot lényege. Ilyenkor a megszakított programhoz nem jut vissza a vezérlés, tehát látszólag a gépben megállt az élet!

Ha nem CTRL+P állapot van, akkor a program az ismétlődések feldolgozásával folytatódik.

D676 21EA0B	LD	HL,0BEA	Ez a DELAY-KEY,
			alapkésleltetésének
D679 CDC0D6	CALL	D6C0	megfelelő, belső számláló

Ez a rutin a következő jelzéseket adja vissza:

Z=1: a számláló értéke 00

Z=0: a számláló még csökkentése után sem lett 00

CY=1: a számláló most - az aktuális csökkentése után - lett 00

D67C 23	INC	HL	Rááll a RATE-KEY számlálójára
			(ismétlési ütem)
D67D D4C0D6	CALL	NC,D6C0	Ha az alapkésleltetés nem
			most telt le, erre is meghívja
			az előbit

D680 D0	RET NC	Ha az egyik számláló sem lett 00: visszatér
---------	--------	---

Ilyenkor tehát nem ad vissza a program kiolvasható kódot. Nézzük, mi van, ha az ismétlési idő letelt.

D681 3AEE0B	LD A, (0BEE)	Ez csak az új billentyűhöz tartozó bitet tárolja,
D684 4F	LD C, A	
D685 ED5BEC0B	LD DE, (0BEC)	ez pedig a megfelelő sor címe az OLD-PIC-ben
D689 21640B	LD HL, 0B64	Az OLD-PIC utolsó sora
D68C ED52	SBC HL, DE	

Mivel most carry=1, ezért nem az utolsó (9.), hanem a megelőző (8.) sorra kérdezzük rá: ez a beépített és az elülső csatlakozóba dugott botkormány sora.

D68E EB	EX DE, HL	HL=aktuális OLD-PIC cím
D68F 280A	JR Z, D69E	Ha itt volt az utoljára érvényesített kód: ugrás
D691 B7	OR A	Most carry=0
D692 EB	EX DE, HL	
D693 21640B	LD HL, 0B64	
D696 ED52	SBC HL, DE	Az utolsó sorban volt? (hátsó botkormány)
D698 EB	EX DE, HL	Ha itt sem, akkor nem a botkormányról van szó:
D699 2009	JR NZ, D6A4	ugrás előre

Ha a program itt maradt, akkor tehát az derült ki, hogy egy botkormány funkcióismétléséről van szó. Ez esetben megnézzük, hogy egyidejűleg másik irány is be van-e kapcsolva, mert ezeket váltakozva kell ismételtetni!

D69B 1666	LD D, 66	b6, b5 és b2, b1 az iránybitek a 8-9. sorban
D69D A2	AND D	Az utolsó funkció ezek közül volt valamelyik?
D69E 2804	JR Z, D6A4	Ha nem: ugrás előre!
D6A0 AA	XOR D	Ha igen, akkor ezzel az eddig aktív bit 0 lesz, viszont ha van másik iránybit is kijelölve, az 1 marad
D6A1 A6	AND (HL)	
D6A2 2011	JR NZ, D6E5	Ha tényleg van másik irány is: ugrás előre

A következő sorokra tehát akkor kerül a vezérlés, ha
 -- a botkormányon csak egy irány van kijelölve,
 -- nem irány, hanem a tűz gomb van megnyomva,
 -- nem a botkormányok sorairól van szó
 Ilyenkor közönséges funkcióismétlést engedünk meg.

D6A4 79	LD	A,C	Vesszük az aktív bitet
D6A5 A6	AND	(HL)	A hozzá tartozó OLD-PIC címen még szerepel ez a bit?
D6A6 200D	JR	NZ,D6B5	Ha igen: ugrás előre

Ha az OLD-PIC-ben az bit már nem szerepel, akkor töröljük az egészet és egyelőre nem adunk vissza kódot.

D6A8 215B0B	LD	HL,0B5B	OLD-PIC kezdőcím
D6AB 060A	LD	B,0A	10 byte-ból áll,
D6AD 77	LD	(HL),A	Töröljük valamennyi
D6AE 23	INC	HL	mátrixelemet,
D6AF 10FC	DJNZ	D6AD	
D6B1 32EBOB	LD	(0BEB),A	a RATE-számlálóba is 00-t
D6B4 C9	RET		írunk és kész

Most már csak az az eset maradt, amikor valóban ismételni kell egy korábban lenyomott billentyűt. Ennek az a technikája, hogy átmenetileg töröljük az OLD-PIC-ben a billentyű lenyomását jelző bitet, megegyeszer átadjuk a vezérlést a leolvasó rutinnak, s ha még mindig le van nyomva a kérdéses billentyű, akkor a rendszer már újként értékeli és visszaadja a kódját. A botkormányok iránybitjeire nézve ez úgy zajlik le, hogy az aktív iránybitet már előbb nulláztuk, ezért a következő sorokban éppen az eddig nem aktív bit válik nullává. Így ismételt leolvasásnál ez lesz az érvényesülő, új kódot előállító bit.

Ha első olvasásra ez bonyolultnak tűnik is, kérem a kedves Olvasót, hogy szükség esetén a leírtakat vesse össze az előbbi utasításokkal, s egy-egy feltételezett szituációt végigkövetve könnyen érthetővé válik a működés. A program nagyon tömör, kevés sorban -- sokat intéz!

D6B5 2F	CPL		
D6B6 A6	AND	(HL)	Az aktív bitet nullázzuk az
D6B7 77	LD	(HL),A	OLD-PIC-ben
D6B8 3A670B	LD	A,(0B67)	A RATE-KEY alapértékével
D6BB 32EBOB	LD	(0BEB),A	inicializáljuk a belső
			számlálót és
D6BE 189E	JR	D65B	ugrás vissza, újra beolvassuk
			a billentyűzetet

Ennyi volt tehát a billentyűzet leolvasását végző vezérlőprogram. Ennek nagyon fontos kiszolgáló szubrutinjára csak utaltunk a megfelelő (D65CH c.) sorban: a beolvasott billentyűzetképből a kód előállítás csinos kis feladat.

Rövidesen rátérünk erre, hiszen csak ennek működését megismerve válhat teljesen egyértelművé az előbbi vezérlőrutin is.

Előbb azonban a késleltetések állapotát visszaadó egyszerű szubrutinon fussunk át:

```

B1          A késleltetések kezelése
D6C0H

          in:  HL=a számláló memóriacíme

D6C0 7E          LD   A,(HL)          A számláló értéke
D6C1 B7          OR   A              Ha már eleve 00, akkor
D6C2 C8          RET  Z              Z=1 és visszatér
D6C3 35          DEC  (HL)          Ha még nem 00: csökkenti
D6C4 C0          RET  NZ            Ha még mindig nem 00:
                                   Z=0-val és CY=0-val RET
D6C5 37          SCF                Ha pedig éppen most lett
D6C6 C9          RET                nulla, akkor Z=1 és CY=1

```

Itt is nagyon sok -- a működésnél rendkívül hasznosan felhasználható -- információt állítunk elő, 7 sorban!

Ezután hozzáfogunk a nagy segédrutin, a lenyomott billentyű kódját előállító és néhány más, járulékos funkciót ellátó rutin ismertetéséhez. E szubrutin hosszabb, mint a vezérlőprogram volt, mégis könnyebb lesz követni, mert kevesebb a más helyekre való hivatkozás. Viszont itt is készüljünk fel a CPU bitkezelő utasításainak igen aktív használatára. A szubrutin szerves részét képezik azok a táblázatok, amelyekből a billentyűk kódjait olvassuk ki, ezekre azonban csak kicsit később kerítünk sort. Nézzük a programot!

```

B2          A billentyűzetmátrix feldolgozása
D6C7H

          in:  Z=0, ha újra el kell készíteni a mátrixot,
              IY a PICTURE mátrix kezdőcíme mutat
          out: HL=0BE6H a CTRL+P állapotot jelző byte címe
              E=FF, ha nincs új billentyű lenyomva vagy "HOLD"
                  állapot van (CTRL+P)
              C=a kiadható billentyűkód

```

D6C7	C4A5D7	CALL	NZ, D7A5	Szükség esetén újra leolvassa a billentyűzetet
D6CA	FDCB069E	RES	3, (IY+06)	Törli a PICTURE mátrixban a SHIFT-nek, a CTRL-nek, az ALT-nak és a LOCK-nak megfelelő biteket
D6CE	FDCB07A6	RES	4, (IY+07)	
D6D2	FDCB0786	RES	0, (IY+07)	
D6D6	FDCB06AE	RES	5, (IY+06)	
D6DA	21E60B	LD	HL, 0BE6	HOLD-cím
D6DD	110A00	LD	DE, 000A	A mátrix 10 soros
D6E0	D9	EXX		
D6E1	21650B	LD	HL, 0B65	Előkészítés a két mátrix vizsgálatához. A DE és HL -- szinkronban -- a PICTURE és az OLD-PIC aktuális elemeire mutat
D6E4	115B0B	LD	DE, 0B5B	
D6E7	1B	DEC	DE	
D6E8	2B	DEC	HL	
D6E9	D9	EXX		
D6EA	1D	DEC	E	Van még sor?
D6EB	F8	RET	M	Ha innen tér vissza, akkor a két mátrix teljesen azonos: nincs új billentyű lenyomva
D6EC	D9	EXX		
D6ED	1A	LD	A, (DE)	Egy sor az utolsó leolvasásnál
D6EE	47	LD	B, A	
D6EF	AE	XOR	(HL)	Ez pedig az OLD-PIC-ből

Ezzel a régi és az új beolvasásnál egyező bitek értéke 0 lett.

D6F0	2F	CPL		Most már: 1
D6F1	A6	AND	(HL)	

Ezután pedig csakis a mindkét helyen 1 értékű bitek maradtak meg.

D6F2	77	LD	(HL), A	Egyelőre ezt tárolja
D6F3	A8	XOR	B	Az új beolvasás értéke

Most viszont a mindkét helyen azonos bitek törlődnek, így csak az új bitek értéke lesz 1.

D6F4	28F1	JR	Z, D6E7	Ha egy új sincs: tovább nézi a mátrixot
D6F6	47	LD	B, A	
D6F7	ED44	NEG		

Ennek hatására a legkisebb helyiértékű bit változatlan marad, a többi viszont átbillen, így a következő:

D6F9	A0	AND	B
------	----	-----	---

hatására már csak egyetlen egy, a legkisebb helyiértékű új bit marad meg. Ezt az egyet vesszük figyelembe.

D6FA 47	LD	B,A	Az aktív bitet tároljuk, sőt
D6FB AE	XOR	(HL)	beírjuk az OLD-PIC-be is --
D6FC 77	LD	(HL),A	legközelebb ez lesz a régi
D6FD 78	LD	A,B	Az új, aktív bitet tartalmazó
D6FE 22EC0B	LD	(OBEC),HL	értéket pedig a címével együtt
D701 32EE0B	LD	(OBEE),A	tároljuk

Következik az elfogadott bitnek megfelelő új kód előállítás.

A kódokat a különböző billentyűzetállapotok szerint 4 táblázatból olvassuk ki. E táblázatokat a billentyűmátrix oszlopai szerinti sorrendben adjuk meg. Egy-egy oszlop 10 billentyűt képvisel és tudjuk, hogy 8 oszlop van.

Az aktív bit alapján ki tudjuk választani, hogy az adott táblázat melyik 10 byte-os területére esik a keresett kód. Az E' regiszter mutatja, hogy ebben az oszlopban hányadik elemről van szó. Így a táblázat megfelelő eleméhez jutunk.

D704 3E50	LD	A,50	8*10=80 (dec). Egy oszlop 10
D706 D60A	SUB	0A	elemből áll
D708 CB38	SRL	B	Itt az aktív bit?
D70A 30FA	JR	NC,D706	Ha nem, tovább megyünk

Tehát a táblázat utolsó oszlopának elejétől kezdve tízesével visszafelé haladunk addig az oszlopig, amelyet az aktív bit kijelöl.

D70C D9	EXX		Ehhez hozzáadjuk a sorszámot,
D70D 83	ADD	A,E	s így máris megvan, hogy a
			táblázat melyik eleméről van
			szó

Most már azt kell eldönteni, hogy melyik táblázatot kell használni. Ez a beállított LOCK-állapotoktól és a pillanatnyilag lenyomott betűváltó billentyűktől függ. Az előkészítés eddigi módszere alapján elég könnyű lesz a megfelelő táblázathoz jutni.

D70E 5F	LD	E,A	DE=eltolás a kódhoz
D70F 3AE80B	LD	A,(OBES)	A betűváltó billentyűk
D712 47	LD	B,A	pillanatnyi állapota
D713 3A660B	LD	A,(OB66)	A LOCK-állapot
D716 AB	XOR	B	Itt a kulcs a táblázatok
D717 47	LD	E,A	kiválasztásához

Tekintsük át B értékét a különböző lehetőségek szerint!

		Pillanatnyi állapot				
		-	SHIFT	CTRL	ALT	
LOCK		00	02	04	08	(0BE8H)
(0B66H)						
	-	00	00	02	04	08
	CAPS	01	01	03	05	09
	SHIFT	02	02	00	06	0A
	ALT	08	08	0A	0C	00

Jól látszik, hogy ezzel a kiosztással sikerült megvalósítani, hogy a SHIFT és az ALT visszaváltó hatású, de a CTRL billentyűhöz kötődő funkciók bármilyen állapotban érvényesülhetnek.

D718 E5	PUSH HL	Ezt felhasználva előállítjuk
D719 C047D7	CALL D747	C-ben a kódot
D71C E1	POP HL	
D71D 3A680B	LD A, (0B68)	A HOLD mód engedélyezett?
D720 3C	INC A	
D721 C8	RET Z	Ha nem: kész!

Ezután a rutin CTRL+P kezelését végzi el, mivel a felhasználó az érvényesítését nem tiltotta meg.

D722 7E	LD A, (HL)	Megnézzük, hogy eddig HOLD
D723 B7	OR A	állapot volt-e
D724 79	LD A, C	A-ba tesszük az új kódot, s
D725 2815	JR Z, D73C	ha eddig nem HOLD volt: ugrás
		előre

Ha a program itt marad, az azt jelenti, hogy eddig HOLD állapot volt, ami most feloldódik, de tisztázni kell, hogy hogyan.

D727 3A110E	LD A, (0B11)	A 03-as port másolata
D72A E6F0	AND F0	A felső biteket meghagyva, az
D72C F607	OR 07	alsó biteken a billentyűzet
D72E D303	OUT (03), A	7. sorát jelöljük ki és
D730 DB58	IN A, (58)	olvassuk be, de csak a b4-b3
D732 E618	AND 18	bit érdekei: a CTRL és ESC

D734 280A	JR Z, D740	Ha ebben a pillanatban ezek
		vannak lenyomva, ugrás előre

D736 AF	XOR A	Egyébként az eddigi STOP
D737 32160B	LD (0B16), A	jelzést -- ha volt --
D73A 1804	JR D740	töröljük és ugrás!

D73C FE10	CP	10	Ha eddig nem volt HOLD állapot, megnézzük, hogy most CTRL+P-t nyomtak-e?
D73E 2803	JR	Z,D743	Ha igen: ugrás előre
D740 3600 D742 C9	LD RET	(HL),00	Ha eddig HOLD állapot volt; töröljük és kész
D743 35 D744 1EFF D746 C9	DEC LD RET	(HL) E,FF	Ha pedig most nyomtak CTRL+P-t, akkor beállítjuk a HOLD jelzést

Mi tagadás, ezt valamivel egyszerűbben is meg lehetett volna oldani -- bár így is jó.

A program tehát visszaadja az új billentyű(k) kódját, emellett elintézi a CTRL+P lenyomásával kapcsolatos intézni valókat is. A C, E és HL regiszterekben további felhasználásra alkalmas adatokat ad át az őt hívó programnak.

Még mindig hátra van azonban a kódokat ténylegesen előállító szubrutin vizsgálata.

B3 A billentyűkód táblázatok kezelése
D747H

in: DE=eltolás a táblázat elejéhez
B=kulcs a táblázatok kiválasztásához:

		Lenyomott			
		-	SHIFT	CTRL	ALT
LOCK	-	00	02	04	08
mód	CAPS	01	03	05	09
	SHIFT	02	00	06	0A
	ALT	03	0A	0C	00

out: C = billentyűkód

D747 2184D7 LD HL,D784

A rutin minden esetben a 0D784H címen elhelyezett utasítássorozattal fejeződik be. Hogy ne kelljen minden részágról megannyi JP-vel odaugrani, a címet a verembe tesszük és közönséges RET-tel tesszük elérhetővé -- tehát egy belső szubrutint hozunk létre.

D74A E5 PUSH HL Ezután a RET-tel ugorhatunk a 0D784H címre!

D74B 215FD8 LD HL,D85F A CTRL-lel együtt lenyomott billentyűkódok táblázata

Ha a CTRL le van nyomva, akkor az összes kódnak függetlennek kell lennie a LOCK állapottól. Ez könnyen elérhető, hiszen a CTRL pillanatnyilag lenyomott állapotát a B kulcs 2. bitje mindentől függetlenül jelzi (ellenőrizzük).

D74E CB50 BIT 2,B Ha a CTRL le van nyomva, ez a
D750 C0 RET NZ táblázat kell

D751 CB40 BIT 0,B CTRL állapot?
D753 281F JR Z,D774 Ha nem: ugrás előre!

Ha a program itt maradt, akkor CTRL állapot van. Ekkor nem kell minden billentyű funkcióját átváltani.

D755 21BFD7 LD HL,D7BF Az alaptáblázat, ebből az
D758 19 ADD HL,DE eltolás után kiolvasható a kód
D759 7E LD A,(HL)
D75A FE61 CP 61 Nagybetűk esetén váltani
D75C 3816 JR C,D774 kell: ugrás előre
D75E FE7B CP 7B Kisbetűk esetén váltani kell:
D760 3808 JR C,D76A ugrás előre
D762 FE90 CP 90 Ha ékezetes nagybetű, váltani
D764 380E JR C,D774 kell: ugrás
D766 FE99 CP 99 Ha nem ékezetes kisbetű:
D768 300A JR NC,D774 ugrás előre

D76A 210FD8 LD HL,D80F A SHIFT-LOCK táblázat
D76D CB48 BIT 1,B Ha a SHIFT nincs lenyomva, ez
D76F C8 RET Z a táblázat kell, ha azonban
D770 21BFD7 LD HL,D7BF SHIFT is van, az alaptáblázat
D773 C9 RET kell a betűkhöz

D774 210FD8 LD HL,D80F SHIFT-LOCK táblázat
D777 CB48 BIT 1,B Ha a SHIFT le van nyomva, ez
D779 C0 RET NZ a táblázat kell
D77A 21AFD8 LD HL,D8AF ALT-LOCK táblázat
D77D CB58 BIT 3,B ALT állapotban pedig ez a
D77F C0 RET NZ táblázat kell

Ha pedig ezek egyike sem, akkor valóban az alaptáblázat kell. Mivel most már simán ráfuthatunk a RET-ekben hivatkozott címre, a veremből eldobhatjuk:

D780 E1 POP HL
D781 21BFD7 LD HL,D7BF Az alaptáblázat
D784 19 ADD HL,DE Hozzáadjuk az eltolást,
D785 4E LD C,(HL) beolvassuk a kódot

D786 79	LD	A,C	
D787 FEFF	CP	FF	A CTRL+ESC kódja?
D789 C0	RET	NZ	Ha nem: készen vagyunk
D78A 32160B	LD	(OB16),A	Egyébként pedig beállítjuk a
D78D 0E1B	LD	C,1B	STOP változót és az ESC kódját
D78F C9	RET		adjuk át

Azt hiszem, kedves Olvasó, ezeket is érdemes -- szükség esetén -- úgy követni, hogy néhány speciális karakterrel végigjárjuk a program működését. Különben csak maga az elv a fontos, az, amit a B regiszterbeli kulccsal kapcsolatban korábban elmondtunk, s az, ahogyan az egyes állapotokban a táblázatok használatában eligazodunk.

A következő egyszerű szubrutint a beolvasást vezérlő program használja: a betűváltó billentyűk állapotát vizsgálja meg és ad vissza ezekre jellemző értéket.

B5 A CTRL, SHIFT és ALT billentyűk állapota
D790H

in: A=a CTRL-hez rendelt érték
out: A=08 ALT lenyomott állapota
 02 SHIFT lenyomva
 00 egyik sincs lenyomva
 változatlan CTRL van lenyomva

D790 FDCB0766	BIT	4, (IY+07)	CTRL?
D794 C0	RET	NZ	Ha igen, visszatér
D795 3E02	LD	A,02	
D797 FDCB065E	BIT	3, (IY+06)	SHIFT?
D79B C0	RET	NZ	Ha igen: A=02-vel,
D79C FDCB0746	BIT	0, (IY+07)	
D7A0 3E08	LD	A,08	ALT esetén A=08-cal,
D7A2 C0	RET	NZ	
D7A3 AF	XOR	A	ha egyik sincs lenyomva:
D7A4 C9	RET		A=00-val visszatér

Végül a billentyűzetkezelő rutinok ismertetését a fizikai leolvasást végző programrészsel fejezzük be. Ez sorra kiadja a billentyűzet fizikai sorcímeit, majd beolvassa az egyes sorok tartalmát, s az eredményeket elhelyezi a PICTURE táblázatban.

Ez az egy rutin tehát, amely közvetlenül a billentyűzetet mint fizikai eszközt kezeli.

B6
D7A5H

A billentyűzetmátrix fizikai leolvasása

D7A5 3A110B	LD	A, (0B11)	A 03-as port kópiája
D7A8 E6F0	AND	FO	A felső 4 bitet változatlanul
D7AA 4F	LD	C,A	kell hagyni, az alsó biteken a billentyűzet sorait választjuk ki
D7AB 21510B	LD	HL, 0B51	A PICTURE eleje
D7AE E5	PUSH	HL	
D7AF FDE1	POP	IY	IY-ba is
D7B1 060A	LD	B, 0A	10 sor lesz
D7B3 79	LD	A, C	A sorszámokat 00...09
D7B4 D303	OUT	(03), A	sorrendben kiküldve
D7B6 DB58	IN	A, (58)	kiválasztjuk a billentyűzet
D7B8 2F	CPL		egy-egy sorát, a megfelelő
D7B9 77	LD	(HL), A	billentyűk állapotát pedig -
D7BA 0C	INC	C	- a biteket megfordítva --
D7BB 23	INC	HL	sorra lerakjuk a PICTURE
D7BC 10F5	DJNZ	D7B3	táblába
D7BE C9	RET		

3.5.5.4 Kódtáblázatok

Ezzel lényegileg a billentyűzettel foglalkozó szoftver végére értünk, csak a hozzá tartozó kódtáblázatok ismertetése maradt hátra.

A következő oldalon megadjuk ezeket, feltüntetve a ROM-címeiket, az egyes oszlopokhoz tartozó 10-10 elem szerinti csoportosításban a kódokat, s alatta a hozzá tartozó karaktereket.

Minden táblázat 8 x 10 elemből áll, a megadott csoportosításban a b7-b6-...-b0 sorrendben haladva. Minden egyes 10-es csoport az oszlopok eredetileg is növekvő sorszámai sorrendjében tünteti fel a billentyűkódokat.

ROM-cím	Sorszám										bit	Alap
	00	01	02	03	04	05	06	07	08	09		
D7BF	34	37	72	75	66	6A	76	6D	2A	2A		b7
D7C9	31	94	71	70	61	91	79	2D	13	F3		b6
D7D3	92	93	40	96	3C	98	2A	20	04	E4		b5
D7DD	36	2A	7A	5B	68	0D	6E	2A	01	E1		b4
D7E7	30	97	3B	95	5C	90	2A	1B	06	E6		b3
D7F1	32	39	77	6F	73	6C	78	2E	18	F8		b2
D7FB	33	38	65	69	64	6B	63	2C	05	E5		b1
D805	35	5E	74	5D	67	08	62	2A	16	43		b0

ROM-cím	Sorszám										bit	SHIFT LOCK
	00	01	02	03	04	05	06	07	08	09		
D80F	21	3D	52	55	46	4A	56	4D	2A	2A		b7
D819	27	84	51	50	41	81	59	5F	13	F3		b6
D823	82	83	60	86	3E	88	2A	20	04	E4		b5
D82D	2F	23	5A	7B	48	0D	4E	2A	01	E1		b4
D837	26	87	24	85	7C	80	2A	1B	06	E6		b3
D841	22	29	57	4F	53	4C	58	3A	18	F8		b2
D84B	2B	28	45	49	44	4B	43	3F	05	E5		b1
D855	25	7E	54	7D	47	07	42	2A	16	49		b0

ROM-cím	Sorszám										bit	CTRL LOCK
	00	01	02	03	04	05	06	07	08	09		
D85F	8B	9C	12	15	06	0A	16	0D	2A	2A		b7
D869	99	DC	11	10	01	D9	19	1F	13	F3		b6
D873	DA	DB	00	DE	3C	98	2A	20	04	E4		b5
D87D	8C	8E	1A	1B	08	0D	0E	2A	01	E1		b4
D887	89	DF	3B	DD	1C	CF	2A	FF	06	E6		b3
D891	8A	9D	17	0F	13	0C	18	2E	18	F8		b2
D89B	9A	8D	05	09	04	0B	03	2C	05	E5		b1
D8A5	9B	1E	14	1D	07	08	02	2A	16	53		b0

ROM-cím	Sorszám										bit	ALT LOCK
	00	01	02	03	04	05	06	07	08	09		
D8AF	A4	A7	C2	C5	B6	BA	C6	BD	2A	2A		b7
D8B9	A1	D4	C1	C0	B1	D1	C9	AD	13	F3		b6
D8C3	D2	D3	E0	D6	AC	D8	2A	20	04	E4		b5
D8CD	A6	AA	CA	CB	B8	0D	BE	2A	01	E1		b4
D8D7	A0	D7	AB	D5	CC	D0	2A	1B	06	E6		b3
D8E1	A2	A9	C7	BF	C3	BC	C8	AE	18	F8		b2
D8EB	A3	A8	B5	B9	B4	BB	B3	AF	05	E5		b1
D8F5	A5	CE	C4	CD	B7	08	B2	2A	16	4C		b0

Ennyi volt tehát, a ROM-nak a billentyűzet kezelésével foglalkozó része.

A számítógép kincsestárához még négy olyan terület tartozik, amely számunkra pillanatnyilag ismeretlen.

Eddig megismertünk néhány kernel funkciót, foglalkoztunk a képmegjelenítés rutinjaival, behatóan tanulmányoztuk a számítógép editorának működését. Az előbb pedig a gép billentyűzetkezelő programjait ismertük meg.

Ami még hiányzik: a nyomtató, a hang és a magnetofon működtetését megvalósító programok.

Az igazsághoz azonban hozzá tartozik, hogy bár a bővítőkártyákkal a géphez kapcsolt külső eszközöket általában külső, saját programok működtetik, a soros vonalat vezérlő szoftver -- kivételesen -- a számítógép saját ROM-jában helyezkedik el.

A felsorolt, még hátralévő eszközök közül a nyomtató és a hangképzés vezértörutinjai itt, a ROM főrészében található. A magnetofon és a soros vonal rutinjai pedig az EXT ROM-ban vannak. Ez utóbbiakkal természetesen ott foglalkozunk részletesen.

3.5.6 A nyomtató

Az Olvasó rövidesen láthatja, hogy a nyomtató program szintű vezérlése mennyire egyszerű.

A számítógép oldaláról nézve a munka maga is az, hiszen a nyomtató feladata mindössze az, hogy a kiküldött karaktereket papíron megjelenítse.

3.5.6.1 A vezérlés folyamata

Intelligens eszközzől lévén szó, a megjelenítés részleteivel a számítógépnek nem kell törődnie. Egyszerű parancskiadásról van szó: közöljük, hogy mit kell kiírni, s megvárjuk, hogy a parancs teljesítéséről visszajelzés jöjjön.

Ennek az eléggé egyoldalú párbeszédnek a hardveres részleteiről már beszéltünk. A nyomtatónak szánt tényleges

információt, parancsot a 01-es portra kell kiküldeni. A feladat elvégzésére való felszólítást úgy oldjuk meg, hogy a 06-os port b7 bitjét rövid időre 1-es szintről 0-ra állítjuk, majd vissza (STROBE jel). A nyomtató figyeli ezt a vonalat, így amikor ilyen 1-0-1 átmenetet észlel, akkor a kapott adatot feldolgozza. A feladat teljesítését pedig egy másik vonalon jelenti vissza a számítógépnek (ACK jel), ezt az 59H-s port b7 bitjén tudja a CPU beolvasni.

A feladat tehát túlságosan is egyszerű -- az eddigiekhez képest -- s lényegileg nem jelent mást, mint hogy az előbb szavakban leírt folyamatot a mikroprocesszor utasításaival kell megfogalmazni.

Igazodunk persze a funkcióhívások többi eszköznél is érvényesített filozófiájához: itt is biztosítjuk a karakteres és blokkos adatkivitel lehetőségét, a funkcióhívások vezérlőrutinjának a nyomtatáéhoz való hozzáférését pedig megfelelő ugrótáblával oldjuk meg.

3.5.6.2 A nyomtató ugrótábla

ROM	Tartalom	Jelentés
D8FF	03	A hívható funkciók száma
D900	D929	0. funkcióhoz tartozó rutin címe
D902	D90C	1. funkció (PR-CHOUT)
D904	D906	2. funkció (PR-BKOUT)

A 0. funkció itt sem takar ténylegesen ellátandó tevékenységet. Tudjuk, hogy jelenlétét a formailag egységes táblázatkezelési technika teszi szükségessé.

3.5.6.3 A nyomtató rutinok

PRINTER 02	PR-BKOUT (printer block-output)
42H	Karakterblokk kiküldése a nyomtatára
in:	DE=a szöveg kezdőcíme a memóriában (ASCII) BC=a szöveget alkotó karakterek száma
out:	A=00 hibátlan működés F5 CTRL+ESC-t nyomtak FA a legmagasabb használható RAM cím átlépése

D906 210CD9	LD	HL,D90C	A kiküldést megvalósító rutin címével
D909 C36DC5	JP	C56D	ugrás a HI-MEM ellenőrzéssel dolgozó szubrutinra

A legmagasabb memóriacím átlépését figyelmes paraméterezéssel nekünk kell megakadályozni. Mint tudjuk, a 0C56DH-n kezdődő vezérlőprogram a HIMEM átlépése esetén sajnálatos eltérést okoz!

PRINTER 01 PR-CHOUT (printer character-output)
41H Egy karakter kiküldése a nyomtatóra

in: C=a karakter kódja
out: A=00 rendben
 F5 CTRL+ESC-t nyomtak

D90C 3A160E	LD	A, (0B16)	STOP-FLAG, ha a billentyűzeten CTRL+ESC lenyomása történt, benne FF van
D90F 3C	INC	A	Ezt ellenőrizzük. Ha valóban CTRL+ESC volt, akkor A=F5-tel visszatérés
D910 3EF5	LD	A,F5	
D912 C8	RET	Z	
D913 DB59	IN	A, (59)	Beolvassuk az ACK jelet, s ha a nyomtató még nem kész a következő adat fogadására: várakozunk!
D915 07	RLCA		
D916 30F4	JR	NC, D90C	
D918 F3	DI		A portok kezelésével járó adatkiküldés idejére tiltjuk a megszakítást
D919 79	LD	A,C	A karaktert a nyomtató adatvezetékeire tesszük, majd a 06-os port kópiáját használva -- a többi bit változtatása nélkül -- a b7 bitbe 0-t, majd 1-t írunk: ez a STROBE jel
D91A D301	OUT	(01),A	
D91C 3A130E	LD	A, (0B13)	
D91F E67F	AND	7F	
D921 D306	OUT	(06),A	
D923 F680	OR	80	
D925 D306	OUT	(06),A	
D927 FB	EI		Engedélyezzük a megszakításokat, és "rendben" jelzéssel visszatérés
D928 AF	XOR	A	
D929 C9	RET		

Kész!

Ennyi volt a nyomtató-vezérlő program.

A 0. funkcióhívás egyébként -- amint az az ugrótáblából is kiderül -- e rutin végén álló RET utasítás.

3.5.7 Hangképzés

Áttérünk a hanggal kapcsolatos funkcióhívások vizsgálatára. Ez sem lesz lényegesen szerteágazóbb, mint a nyomtatókezelés volt, hiszen a funkciók száma itt is szűkös. Le kell kezelni a felhasználó hangképzésre vonatkozó kéréseit, azután gondoskodni kell annak lebonyolításáról, ill. a vele kapcsolatos adminisztrációk elvégzéséről.

Itt azért van egy kis munka -- főleg az utóbbi téren -- hiszen tudjuk, hogy ugyanaz a frekvenciaosztó a számítógépben többféle funkció ellátására is szolgál. Csak egy példa: biztos, hogy minden hangképzés átállítja a frekvenciaosztót úgy, hogy az a soros vonal órajeleként már nem használható. Ezt a körülményt jelezni kell.

3.5.7.1 A hangfunkciók ugrótáblája

ROM	Tartalom	Jelentés
D92A	04	A hívható funkciók száma
D92B	D933	0. funkció (SOUND-INT)
D92D	D960	1. funkció (NOP)
D92F	D960	2. funkció (NOP)
D931	D961	3. funkció (TONE-SET)

Nyilvánvaló, hogy az üres funkciójú hívások magyarázata ugyanaz, mint eddig: az egységes táblázatkezelés, valamint a funkciók számozásának logikája miatt kell a táblázatokban ezeknek helyet biztosítani.

3.5.7.2 A hangképzés szubrutinjai

A 0. funkcióval kezdünk. Nagyon lényeges szubrutin ez, s amint a sorszámából is kiderül, elsősorban az operációs rendszer megszakításkezelő alprogramjától kapja meg a vezérlést. Felügyelő rutin ez: a megszólaltatott hang időtartamának visszaszámlálása, majd a megadott idő elteltével a hangképzés leállítása a fő feladata. Mindezt az IT rutin végzi, ezért a főprogram számára ez is éppolyan tudatalatti működés, mint pl. a billentyűzet kezelése. A főprogramból a hangképzésre kiadott parancs után nem kell

többé annak részleteivel, végrehajtásával foglalkozni. Az IT alatt futó felügyelő program gondoskodik annak pontos végrehajtásáról, a munka közben folyhat tovább.

```
HANG 00          SOUND-INT
30H              A hangképzés felügyelő programja

D933 3A140B      LD    A,(0B14)          SOUND, értéke 0FFH, ha hang
                                  van folyamatban
D936 3C          INC    A              Van hang?
D937 C0          RET    NZ            Ha nincs: visszatér
```

Normális körülmények között a funkcióhívások gondoskodnak arról, hogy a SOUND-INT rutin csak akkor legyen bekapcsolva az IT-t kiszolgáló programba, ha hang is van, azonban más esetben előfordulhat az előbbi szituáció. Ekkor a program dolgavégezetlenül tér vissza.

```
D938 3A160B      LD    A,(0B16)          A hangképzés alatt a CTRL+ESC
D93B 3C          INC    A              ellenőrzése
D93C 2808        JR    Z,D946          Ha STOP volt, a hangot azonnal
                                  le kell állítani: ugrás előre!

D93E 3AEF0B      LD    A,(0BEF)          Itt tároljuk a hang még
D941 3D          DEC    A              hátralevő időtartamát, mindig
D942 32EF0B      LD    (0BEF),A        1-gyel csökkentve az értékét,
                                  0-ig
D945 C0          RET    NZ            Amíg az előírt idő nem telt
                                  le: nincs több tennivaló
```

Ha közben CTRL+ESC-t nyomtak, vagy ha az előírt időtartam letelt, akkor a program így folytatódik:

```
D946 32140B      LD    (0B14),A        00-t beírva jelezzük, hogy már
                                  nincs hang, a tartamszámláló
                                  is 0
D949 32EF0B      LD    (0BEF),A
D94C 3A100B      LD    A,(0B10)       A kurzor IT által
D94F F608        OR    08              kiszolgálható eszközök közül
D951 32100B      LD    (0B10),A        kikapcsoljuk a hang rutinját
                                  (b3=1)
D954 3A120B      LD    A,(0B12)       A 05-ös port kópiáját
                                  használva -- a többi bit
                                  változtatása nélkül -- törli
                                  a b4 és b5 bitet, ezzel
                                  ténylegesen tiltja
                                  a hangot és a hang IT-t
D957 E6CF        AND    CF
D959 32120B      LD    (0B12),A
D95C D305        OUT   (05),A
D95E 3EF5        LD    A,F5
D960 C9          RET
```

Látható, hogy a 0D960H-ra írt RET egyben a tényleges tevékenységet nem hordozó 1. és 2. funkcióhívás egyetlen utasítása.

A felhasználó szempontjából a hangképzéshez való praktikus hozzáférést a 3. funkcióhívás biztosítja.

```

HANG 03          TONE-SET
33              Hang beállítása és megszólaltatás
               in: B=időtartam, 20 ms egységekben
                   C=hangerő (0...15)
                   DE=hangmagasság (PITCH értéke, tehát a hang
                       frekvenciája: 195312.5/(4096-PITCH) Hz)

D961 3A150B     LD    A,(0B15)      Az új hang megszakíthatja a
D964 3C         INC    A            régit?
D965 2807       JR    Z,D96E       Ha igen: ugrás előre,
D967 3AEF0B     LD    A,(0BEF)     egyébként vár, amíg az előző
D96A D602       SUB    02          hang tartama 1-re
D96C 30F3       JR    NC,D961      csökken (nem 0-ra!)

D96E AF         XOR    A            Beállítja az "előző hang
D96F 32140B     LD    (0B14),A        befejeződött" jelzést
D972 3A160B     LD    A,(0B16)     Ellenőrzi a STOP-FLAG-et és
D975 3C         INC    A            ha CTRL+ESC-t nyomtak az
D976 28CE       JR    Z,D946      előző rutin végére ugorva
                                   megfelelő intézkedések után
                                   visszatér

D978 78         LD    A,B          Megnézzük a hang beírt
D979 B7         OR    A            időtartamát
D97A 2005       JR    NZ,D981     Ha nem 0: mehet tovább,
D97C CD46D9     CALL D946        míg 0 időtartam esetén
D97F AF         XOR    A            tiltjuk a hangot (előző
D980 C9         RET                rutin vége) és kész is

D981 32EF0B     LD    (0BEF),A     A tartamot betöltjük a
                                   számlálóba, majd
D984 79         LD    A,C          vesszük a megadott hangerőt,
D985 E60F       AND    0F         de csak az alsó bitek
D987 07         RLCA            kellenek. Felhasználás előtt
D988 07         RLCA            a b3...b0 biteket a b5...b2-
D989 4F         LD    C,A         re kell léptetni (ilyen a
                                   port bitkiosztása)

D98A 3A130B     LD    A,(0B13)     Ezután a 06-os port
D98D E6C3       AND    C3        kópiájában e biteket
D98F B1         OR    C          szabaddá tesszük és beírjuk
D990 32130B     LD    (0B13),A     az aktuális hangerő bitjeit
D993 D306       OUT   (06),A

```


D995 7A	LD	A,D	A hangmagasság PITCH értéke
D996 E60F	AND	OF	felső byte-jából is csak b3... b0 kell
D998 F610	OR	10	Az 5. bittel engedélyezzük a
D99A 57	LD	D,A	hangjelet, majd mindezt a 05-
D99B 3A120B	LD	A,(0B12)	ös port másolata alapján, a
D99E E6C0	AND	C0	többi
D9A0 B2	OR	D	bitet meghagyva, be is
D9A1 32120B	LD	(0B12),A	kapcsoljuk az új PITCH felső
D9A4 D305	OUT	(05),A	byte-ját
D9A6 7B	LD	A,E	A PITCH alsó byte-ját a 04
D9A7 D304	OUT	(04),A	portra kell írni
D9A9 3EFF	LD	A,FF	"Hang van folyamatban"
D9AB 32140B	LD	(0B14),A	jelzés, valamint az "órajel
D9AE 32710B	LD	(0B71),A	a soros vonalhoz rossz"
D9B1 3A100B	LD	A,(0B10)	jelzés után a
D9B4 E6F7	AND	F7	hangkezelést az IT-be
D9B6 32100B	LD	(0B10),A	bekapcsoljuk,
D9B9 AF	XOR	A	"rendben" jelzés
D9BA C9	RET		és kész

A többi már az IT alatt futó SOUND-INT rutin dolga.

3.5.8 Előzetes a magnetofonkezelő programokhoz

A következő feldolgozatlan eszköz a magnetofon. A vele kapcsolatos rutinok a ROM talán legizgalmasabb részét jelentik. Egyelőre azonban csak nagyon röviden tudjuk elintézni a vele kapcsolatos munkát, mivel az összes végrehajtó szubrutin az EXT ROM-ban foglal helyet. Ugyanez a helyzet a soros vonallal is.

A magnetofonkezelő rutinokkal tehát egyelőre várnunk kell. Itt, a ROM fő szegmensében csak az ugrótáblát, s EXT ROM-ba átvezető utasításokat találjuk meg.

ROM	Tartalom	Jelentés
D9BB	06	A hívható funkciók száma
D9BC	D9E7	0. funkció (NOP)
D9BE	D9D2	1. funkció (CAS-CHIN/OUT)
D9C0	D9D7	2. funkció (CAS-BKIN/OUT)
D9C2	D9C8	3. funkció (CAS-OPEN/CRTE)
D9C4	D9CD	4. funkció (CAS-CLOSE)
D9C6	D9DC	5. funkció (CAS-VERIFY)

A funkcióhívások vezérlőrutinjából a CPU az ugrótáblákon át a megfelelő sorszámmal kiválasztott címhez jut, amelyeken csak a következő továbblendítő utasításokat találja.

Elágazás a magnetofont kezelő rutinokhoz:

D9C8	21E2F3	LD	HL,F3E2	3.funkció EXT ROM címe
D9CB	1812	JR	D9DF	
D9CD	21E7F3	LD	HL,F3E7	4.funkció EXT ROM címe
D9D0	180D	JR	D9DF	
D9D2	21D8F3	LD	HL,F3D8	1.funkció EXT ROM címe
D9D5	1808	JR	D9DF	
D9D7	21DDF3	LD	HL,F3DD	2.funkció EXT ROM címe
D9DA	1803	JR	D9DF	
D9DC	21ECF3	LD	HL,F3EC	5.funkció EXT ROM címe
D9DF	C3F0FF	JP	FFFO	***** ugrás az EXT ROM-ba
D9E2	21F1F3	LD	HL,F3F1	A magnetofon inicializáló
D9E5	18F8	JR	D9DF	rutin EXT ROM címe
D9E7	C9	RET		Ez a 0. funkció rutinja

3.5.9 Összefoglalás

Ezzel a számítógép kincsestárában való szemlélődésünk itt megszakad.

Mitöbb, ennek az alfejezetnek is a végére értünk!

Összefoglaló áttekintésre valójában csak akkor lenne okunk, amikor már az összes eszközt, ill. kiszolgáló, vezérlő rutinjaikat megismertük. Mégis, talán így sem lesz hasztalan most megállni, és néhány gondolat erejéig értékelni az eddig bejárt utat!

Először megvizsgáltuk a számítógépet, mint fizikai eszközt. Közben megismerkedtünk a működés olyan részleteivel, amelyek egyrészt későbbi munkánkat készítették elő, másrészt tudatosították bennünk, hogy a számítógép működése csak a szerkezeti, funkcionális és logikai hierarchia, az egymást feltételező, egymásra épülő szintek figyelembevételével közelíthető meg!

Egyértelművé vált, hogy bár a mikroprocesszor utasításkészlete önmagában is lehetővé tenné az összes eszköz célszerű működtetését, mégis -- tekintettel arra, hogy az egyes eszközök feladatai ismétlődők, sőt szét is választhatók -- érdemes a velük kapcsolatos funkciókat megvalósító utasítássorozatokot egyszer, külön megírni, majd ezeket a felhasználó számára bármikor jól hozzáférhetővé tenni.

E mechanizmus egy része a mikroprocesszorok megszakításkezelő képességei révén még magas szinten automatikussá is tehető!

Egy ilyen operációs rendszer jelenléte a számítógéppel végzett munka elengedhetetlen feltétele.

Megismertük a TV-Computer memóriakezelési sajátosságait. Áttekintettük mindazokat a szubrutinokat, amelyeket felhasználva az eddigiéknél átfogóbb problémák értelmezésébe kezdetünk.

Ehhez is megtettük az első lépéseket!

Megismertük

- a TV-computerben használt ún. "elsődleges token"-eket,
- az RST 18 utasításra építhető és magasabb szintű utasítások sorozatává szervezhető műveleteket,
- az eszközök leggyakoribb feladatait egyszerűen lebonyolító funkcióhívásokat.

Megtudtuk, hogy mi történik a gépben a bekapcsolás első pillanatától kezdve (majdnem) az első gombnyomásig. Ott azt mondtuk, hogy ha a készülékhez kapcsolt külső programmodulok nem akarják átvenni a gép működésének irányítását, akkor a vezérlést a gép ROM-jába írt BASIC kapja meg, a 0D9EFH címen.

Most ehhez a címhez értünk!

3.6 B A S I C

3.6.1 Kezet nyújt a BASIC

Amikor az inicializáló program eljutott addig, hogy a gép bal oldalába dugaszolható programmodul nem akarja átvenni a vezérlést, akkor a Z80-as CPU a ROM 0D9EFH címére ugrik.

Az itt kezdődő utasítássorozat -- ha csak a RESET gombot nem nyomkodják állandóan -- egyenes és rövid úton a TV-Computer BASIC-jéhez vezet.

Ezt a rövid utat tesszük meg mi is a következőkben.

A magnetofont kezelő rutinokhoz való elágazás után néhány azonosító byte van még:

```
D9E8 28 63 29 20 49 53 4C (C) ISL
```

A CPU azonban az eszközök inicializálás után 0D9EFH-ra ugrik, s ettől kezdve már komoly utasítássorozat áll.

D9EF 210017	LD HL,1700	A BASIC munkaterület
D9F2 E5	PUSH HL	
D9F3 DDE1	POP IX	IX tehát e területre áll
D9F5 3A210B	LD A,(0B21)	Még egyszer megnézzük, hogy miről van szó?
D9F8 B7	OR A	Meleg reset van?
D9F9 C2D3DA	JP NZ,DAD3	Ha igen: ugrás előre!

A következő sorokra tehát csak hideg reset esetén kerül a vezérlés.

D9FC 01EF02	LD BC,02EF	A 1700H címtől 19EFH-ig 00-val töltjük fel az U0 RAM-ot:
D9FF 77	LD (HL),A	a teljes munkaterület törlése
DA00 EDA1	CPI	
DA02 EAFFD9	JP PE,D9FF	
DA05 222017	LD (1720),HL	VLOMEM, a BASIC felhasználói báziscíme: 19EFH
DA08 222217	LD (1722),HL	TEXT, a program inicializált kezdőcíme: 19EFH

DA0B 215BFE	LD HL,FB5B	Az U0 RAM 0008H címére
DA0E 110800	LD DE,0008	másoljuk a hibakezelő
DA11 012700	LD BC,0027	rutinokat és töröljük az
DA14 EDB0	LDIR	USRTAB címtáblázatot (a
		felhasználói gépi kódú
		rutinok kezdőcímei)
DA16 CD10DE	CALL DE10	Mint egy NEW parancs

Ez a szubrutin a BASIC munkafeltételeket kiindulási helyzetbe hozza:

- a program kezdőcímét a báziscímre teszi és az első byte-ra 00-t írva "nincs program" állapotot állít be;
- törli a BASIC vermet azáltal, hogy IY-t a HI-MEM-ben tárolt legmagasabb ROM címre állítja és 00-val tölti, ami a verem terület végét jelenti;
- a 170E-170F és 1714-1715H címeket nullázza: END állapot jelzése;
- a CHAIN (1724H) változóba 0F094H-t töltve, ami a ROM utolsó beépített szimbólumának címe, törli az össze felhasználói szimbólumot;
- a 1709H címre 09-et tölt;
- 170A-ra a 1709H címet teszi le;
- 1712-re a BASIC program kezdőcíme kerül;
- a TOP (1726H) változóba 19F0-t tesz, a szimbólumtábla következő szabad byte-jának címeként.

DA19 1115DC	LD DE,DC15	A palettaregiszterekbe a
DA1C F7 0C	RST 30: 0C	kék-vörös-sárga-cián
DA1E 3E02	LD A,02	színeket tölti és kékre
DA20 324F0B	LD (0B4F),A	állítja a bordert is
DA23 DD360500	LD (IX+05),00	A funkcióhívásokat a VIDEO-
DA27 010303	LD BC,0303	hoz irányítja és a 3. sor 3.
DA2A 3E0C	LD A,0C	pozícióján elkezdi a
DA2C C5	PUSH BC	nyitókép összeállítását
DA2D F5	PUSH AF	
DA2E F703	RST 30: 03	Pozícionál
DA30 CDF2DB	CALL DBF2	"VIDEOTON" kiírása
DA33 F1	POP AF	
DA34 F5	PUSH AF	
DA35 CD0CDC	CALL DC0C	majd A számú üres hely,
DA38 CDF2DB	CALL DBF2	megint "VIDEOTON",
DA3B CD93FE	CALL FE93	azután új sort kezd, s a
DA3E F1	POP AF	paraméterek változtatásával
DA3F C1	POP BC	létrehozza a képernyőn a
DA40 04	INC B	TV-Computer ismert
DA41 0C	INC C	bejelentkező nyitóképének
DA42 0C	INC C	felső részét
DA43 D602	SUB 02	
DA45 30E5	JR NC,DA2C	

DA47 01150D	LD BC,0B15	A 21. sor 13. oszlopra
DA4A F703	RST 30: 03	pozícionál és ismét a
DA4C CDF2DB	CALL DBFZ	"VIDEOTON" szó kiírása
DA4F 01120B	LD BC,0B12	Majd a 18. sor 11. oszlopra
DA52 F703	RST 30: 03	áll és
DA54 21FFDB	LD HL,DBFF	
DA57 46	LD B,(HL)	
DA58 23	INC HL	
DA59 7E	LD A,(HL)	elkezd
DA5A CD9AFE	CALL FE9A	kiírni a "TV COMPUTER"
DA5D E5	PUSH HL	szöveget, karakterenként és
DA5E C5	PUSH BC	az egyes betűk színét
DA5F CDD8E6	CALL E6D8	véletlenszerűen változtatva
DA62 E603	AND 03	
DA64 28F9	JR Z,DA5F	
DA66 C1	POP BC	
DA67 E1	POP HL	
DA68 324DOB	LD (0B4D),A	
DA6B 0D	DEC C	
DA6C 20FD	JR NZ,DA6B	(Kis várakozás)
DA6E 10E8	DJNZ DA58	
DA70 F793	RST 30: 93	Amikor végigért, megnézi,
DA72 0C	INC C	hogy van-e lenyomva
DA73 20DA	JR NZ,DA4F	billentyű, s ha nincs: az
		előbbieket ismételteti

Nos, itt következik az "első gombnyomás"! A program addig nem megy tovább, amíg valamelyik billentyűt le nem nyomjuk.

DA75 F791	RST 30: 91	Amikor ez megtörtént,
		kiolvasással nyugtázza a
		gombnyomást,
DA77 0E01	LD C,01	4-színű üzemmódot
DA79 F704	RST 30: 04	állít be,
DA7B 324FOB	LD (0B4F),A	a border fekete lesz,
DA7E CD18FC	CALL FC18	vigyáz, hogy az INK és a
		PAPER különböző legyen,
DA81 CD79FE	CALL FE79	majd kiírja a következő
		szöveget

```

DA84 32 54 56 20 43 4F 4D 50 55 54 45 52 20 42 41 53 49
DA95 43 20 31 2E 32 0D 0A 43 6F 70 79 72 69 67 68 74
DAA5 20 31 39 38 35 20 56 49 44 45 4F 54 4F 4E 0D 0A
DAB5 0D 0A

```

Az ASCII kódban lerakott szöveg:

```

"TV COMPUTER BASIC 1.2
Copyright 1985 VIDEOTON"

```

```
DAB7 CD4CEC      CALL EC4C
DABA CDC1FE      CALL FEC1
DABD CD1BFA      CALL FA1B
DAC0 CD79FE      CALL FE79
```

Ezután kiszámítja a BASIC-ben rendelkezésre álló szabad területet és kiírja e számot, a végére pedig ezeket:

```
DAC3 0F  20 62 79 74 65 73 20 66 72 65 65 0D 0A 0D 0A
```

tehát "bytes free" és két sorral lejjebb, a sor elejére áll.

Ezzel el is jutottunk odáig, ahová meleg reset esetén a belépés történik:

```
DAD3 AF          XOR  A          "Nem meleg reset van
DAD4 32210B      LD   (0E21),A   folyamatban" jelzés és a
DAD7 CDFCDC      CALL DCFC      BASIC munkaterület
                                     inicializálása
```

Ez a szubrutin része a 0DE100H kezdőcíműnek, működését már leírtuk. A különbség csak annyi, hogy az ott közölt első pontbeli programtörlés elmarad.

És itt most meg is állunk.

Ami ezután következik, az már a TV-Computer BASIC-jének működési területe. A mikroprocesszor természetesen éppen itt sohasem áll meg. Az ajtó, amely előtt most állunk, csak logikailag létezik.

De ez az ajtó már előttünk is teljesen nyitva áll! Amikor ide belépünk, a mikroprocesszor utasítássorozatát látjuk magunk előtt, mint eddig. Azonban e sorok mögött már egy minőségileg magasabb rendű működés rejtőzködik.

A hangsúly is óhatatlanul erre a magasabb logikai működési szintre helyeződik át. A ROM képviselte "agysejtek" tartalma ezután még távolabbra kerül a véges céltől! Az egész hatalmas szervezet egyetlen komplex funkciót: az emberrel folytatandó párbeszéd szolgálatát látja el.

Mi a továbbiakban ennek a szolgáltatnak a mikéntjét szeretnénk nyomon követni, ami bizonyos értelemben a munkánkban is stílusváltást követel.

Kezét nyújtja felénk a BASIC! Ajtaja nyitva áll előttünk, lépünk be rajta!

3.6.2 A magas szintű parancsok fogadása

Amikor a küszöbön átlépünk, látszólag nagyon lázas munkának lehetünk tanúi. Rövidesen kiderül azonban, hogy e serénykedés gépi oldalról megmutatkozó szándéka az emberrel folytatandó párbeszéd során eléggé unott, bár állandó készenlétté szelídül.

Kívülállóként kövessük nyomon az eseményeket!

Rögtön a "küszöbön" átlépve ez áll:

DADA 31AC16	LD	SP,16AC	Ezekben az utasításokban
DADD 210317	LD	HL,1703	semmi különös nincs, a
DAE0 7E	LD	A,(HL)	verem kezdőértéket kap, s
DAE1 B7	OR	A	ha a 1703H címen nem 0 áll,
DAE2 3600	LD	(HL),00	újra meghívódik a már
DAE4 C410DE	CALL	NZ,DE10	nagyvonalakban megismert
			BASIC inicializáló rutin
DAE7 DD360520	LD	(IX+05),20	Az editorhoz fordulunk

A 1705H változó arra szolgál, hogy benne a későbbi funkcióhívásainkban gyakran használt funkcióosztályt nevezzük meg. Így a tényleges hívásokat lebonyolító rutinoknak nem kell már foglalkozniuk az osztálykijelöléssel. Az itt letett értékhez csak a művelet irányát és a funkció számát kijelölő biteket teszük hozzá.

Amikor a vezérlés ezekre a sorokra kerül, akkor a BASIC kiindulási helyzetben van. Ebből adódik, hogy a későbbiekben elvégzendő tevékenység kijelölését ilyenkor az embertől várja.

Az emberrel való kapcsolattartás legkézenfekvőbb módja viszont éppen az erre a célra kifejlesztett editor, amely képes arra, hogy a gép billentyűzetét és képernyőjét egy komplett mondat, vagy bekezdés megszerkesztésének idejére az ember rendelkezésére bocsássa. Ezt követően a begépelte dolgokat már külön fel lehet dolgozni.

DAEB DDCB0096	RES	2,(IX+00)	"Nincs futó program"
DAEF DDCB004E	BIT	1,(IX+00)	Ha nem kell "ok" üzenetet
DAF3 200A	JR	NZ,DAFF	írni: ugrás, egyébként az
DAF5 CD18FC	CALL	FC18	INK és a PAPER ellenőrzése,
			illesztése

DAF8 CD79FE CALL FE79
DAFB 03 6F 6B 0B

Kiírja az "ok" üzenetet, s
törli a sor utána álló
részét

Ez az ügyes szubrutin úgy használható szövegek kiírására, hogy a hívását követő byte-okra egyszerűen le kell tenni a kiírandó szöveg hosszát és a karaktereket. A szöveg után a program folytatólagosan írható tovább.

DAFF DDCB008E RES 1,(IX+00)
DB03 CD93FE CALL FE93

Ezután kell "ok" üzenet
A következő sor elejére
áll

Ezután kezdődik a tényleges munka.

DB06 CD4FFF CALL FF4F

Ezzel a szubrutinnal a vezérlést átadja az embernek: meghívja az editor karakterbeolvasó funkcióját, amely egy egész bekezdés begépelését jelenti, s ennek végén ismételt hívásokkal az editor egyenként visszaadja az egész begépelte anyagot.

A szubrutin az átvett karaktereket a kitüntetett jelentőségű 1831H kezdőcímű pufferben tárolja úgy, hogy az elejére a beírt szöveg hosszát írja, a végére pedig egy OFFH, sorvége kódot tesz.

Visszatéréskor HL a 1831H címre mutat.

Ebben a pillanatban tehát a gép 1831H kezdőcímű pufferében rendelkezésre áll az a teljes bekezdésnyi anyag, amelyet a RETURN (vagy ESC) billentyű lenyomásáig az ember beírt.

Kezdődhet a feldolgozás.

DB09 FEF5 CP F5
DB0B CAA3FF JP Z,FFA3
DB0E FEED CP EC
DB10 28E3 JR Z,DAF5

Ha érdemi karakterbeírás
nem történt, akkor kezdődik
minden előlről

Ilyenkor CTRL+ESC-t nyomtak, ami érvényteleníti a beolvasást. A vezérlés az ezt kezelő 0FFA3H rutinon keresztül visszakerült a 0DADAH címre. Ha pedig az editor osztályhoz más eszköz volt rendelve, a beolvasás egy file-vége karakterrel szakadt meg. Ilyenkor is visszaugrunk a rutin egy korábbi pontjához.

Láttuk, hogy az editor karakterbeolvasó rutinja a billentyűzetről való beolvasás idején kitartóan várakozik a gombnyomásainkra. Itt is ez teszi ki az idő fő részét.

Feltehető azonban, hogy a 1831H kezdőcímű input pufferben előbb-utóbb nem csak egy hibakódot talál a program.

```
DB12 CD19DC      CALL DC19      Tömörített BASIC sort állít
                                     elő
```

Ennek lényege, hogy a 1831H-s input puffer tartalma alapján a benne található kulcsszavakhoz megkeresi ezek tokenjeit - egy byte-os szimbólumok --, a számokat és a nem tokenizálható karaktereket pedig változatlanul hagyja.

Az eredményt a 1732H kezdőcímű COMMAND (parancs) pufferben állítja elő. Itt a bevitt BASIC sor már majdnem teljesen abban a formában van, ahogy a BASIC nyelvű programsorok tárolódnak.

Ezután elkezdődik az aktuális parancs feldolgozása.

```
DB15 5D          LD E,L          DE=1735, a COMMAND sor
DB16 54          LD D,H          hosszára mutat,
DB17 23          INC HL          HL pedig az első
DB18 CD14F9      CALL F914      parancsbyte-ra
```

Ez a szubrutin a COMMAND pufferben levő számot dolgozza fel: az értékelhető számokat a BASIC verembe teszi. Ha nem számot talál, A=00-val és carry=0-val tér vissza.

Munkánk jelenlegi fázisában nyilván azt kell eldönteni elsőként, hogy a COMMAND pufferben egy azonnal végrehajtandó parancs van, vagy pedig egy tervezett BASIC nyelvű programsor. Ez éppen az első utasításbyte-ok vizsgálatával dönthető el.

```
DB1B 380D        JR C,DB2A        Ha a COMMAND-beli BASIC
                                     sorszámval kezdődik, ugrás
                                     előre
```

Ha a program itt marad, tehát a beírt sor nem számmal kezdődik, akkor sok tennivaló nincs: vagy valódi parancsról van szó, s akkor eszerint kell eljárni, vagy pedig egyáltalán nincs semmi tennivaló. Az egyszerűbb eljárás érdekében ezt külön érdemes tisztázni!

```
DB1D 7E          LD A,(HL)        Az első utasításbyte
DB1E FEAB        CP AB          Ez a "*" tokenje, az így
                                     kezdődő sorokat a rendszer
                                     nem értelmezi
DB20 2B03        JR Z,DB25        Ha "*": ugrás előre
DB22 3C          INC A          Sor vége?
DB23 203C        JR NZ,DB61       Ha nem, akkor ugrás a
                                     parancsok feldolgozásához
```

DB25 CD1BFA	CALL FA1B	Egyébként a verem törlése
DB28 18DC	JR DB06	után ugrás vissza

Ott tartunk tehát, hogy a BASIC bekért egy sort a billentyűzetről, vagy más eszköztől az editor segítségével a 1831H INPUT-pufferbe. Ezt tokenekkel megfogalmazott szabványos BASIC formátumúra alakította a COMMAND pufferben, majd ha az első utasításbyte alapján a beírt sor egy program része lehet - mert számmal kezdődik -- akkor a 0DB2AH címen folytatja a végrehajtást.

Ha az első byte nem szám, akkor a 0DB61H parancsvégrehajtó ágra ugrik, kivéve ha "sor vége" vagy "*" karaktert észlelt, mert ekkor visszatér egy új sor beolvasásához.

Nézzük ezután a számmal kezdődő COMMAND sorok feldolgozását!

DB2A 17	RLA		Ha a számbeírást elemző
DB2B 3834	JR	C, DB61	szubrutin szerint a beírt
DB2D 46	LD	B, (HL)	érték az A-ban visszaadott
DB2E FD7E08	LD	A, (IX+08)	jelzőbyte, vagy a szám
DB31 FE40	CP	40	exponense, vagy a számjegyek
DB33 382C	JR	C, DB61	miatt nem a 1...9999
DB35 FE44	CP	44	tartományba esik, akkor nem
DB37 3028	JR	NC, DB61	lehet szó egy BASIC
DB39 FD7E06	LD	A, (IY+06)	programsorról. Ilyenkor
DB3C B7	OR	A	megkísérli a
DB3D 2822	JR	Z, DB61	parancsfeldolgozó ágra
			küldeni

3.6.3 A programsorok beillesztése

Érvényes kezdősorszám esetén a gép a COMMAND puffer tartalmát egy új BASIC nyelvű programsorként dolgozza fel. Az eljárás a következő:

DB3F 7E	LD	A, (HL)	A sorszám után írt
DB40 23	INC	HL	esetleges szóközöket átlépi
DB41 FE20	CP	Z0	és elmegy a COMMAND-ba az
DB43 28FA	JR	Z, DB3F	első nem-szóköz karakterig
DB45 2B	DEC	HL	
DB46 EB	EX	DE, HL	
DB47 7E	LD	A, (HL)	A kezdőcímből és a
DB48 23	INC	HL	sorszámot követő szóköz
DB49 C602	ADD	A, 02	karakterek számából
DB4B ED52	SBC	HL, DE	előállítja a sor végleges
DB4D 85	ADD	A, L	hosszát

DB4E 4F	LD	C,A		
DB4F CDC3FA	CALL	FAC3	Előállítja	HL-ben a sorszámot

Az itt meghívott szubrutin a BASIC veremben tárolt számot dolgozza fel és a HL regiszterpárban már olyan formában adja vissza, ami megfelel a programsorok tárolásánál alkalmazott formátumnak: 2 byte-os egész számként.

DB52 EB	EX	DE,HL		
DB53 2B	DEC	HL		
DB54 72	LD	(HL),D		Az első utasításbyte elé
DB55 2B	DEC	HL		leteszi az így megkapott
DB56 73	LD	(HL),E		sorszámot,
DB57 2B	DEC	HL		
DB58 71	LD	(HL),C		a legelejére pedig a
				programsor hosszát, s így
				végleges formájában előállt
				az új BASIC sor
DB59 220C17	LD	(170C),HL		Kezdőcímét tárolja
DB5C CDAFDC	CALL	DCAF		

Ez a nagyon fontos szubrutin az új programsort beilleszti a többi közé és elvégzi a régebbi sorok ezzel összefüggő elmozgatását. Sajnos, egyúttal a felhasználó által esetleg korábban létrehozott szimbólumtábla is megsemmisül.

DB5F 18A5	JR	DB06	Ezután ugrás vissza: új parancsok és programsorok bevitelére a vezérlés ismét az emberhez kerül
-----------	----	------	---

És ez így zajlik egészen addig, amíg a felhasználható memória meg nem telik.

Ez a programok beírásának alapvető mechanizmusa, amelyről látnunk kell, hogy egy célszerű kezelési eljárásnak alávetett nagy adminisztráció csupán.

Gépelgethetünk akár órákig -- ha az editor közreműködésével a gépbe juttatott bekezdéseink 1...9999 közötti számmal kezdődnek, akkor az előbbi programrészek csak egy aránylag pontos másolatot készítenek a munkánkról a memóriában.

Egyébként ez így van abban az esetben is, ha a beírt szövegben egyetlen BASIC kulcsszó sincs! A munkának ez a fázisa semmiféle ellenőrzést a beírt sorok tartalmára nézve nem tartalmaz. A BASIC szabályaival kapcsolatos esetleges hibák a program végrehajtásakor derülnek ki.

3.6.4 Az utasítások végrehajtásának vezérlése

Amikor nem sorszámmal kezdődik a COMMAND puffer tartalma, vagy ez a szám nem esik a megfelelő tartományba, akkor a vezérlést egy másik ág: a 0DB61H kezdőcímű utasítássorozat kapja meg.

Ekkor a BASIC megkísérli a korábban beírtak tényleges értelmezését és ha lehet, hozzá is fog az abban foglaltak végrehajtásához.

Ez már a működésnek egy minőségileg új fázisa.

A következőkben nagyvonalakban áttekintjük, hogyan történik a BASIC nyelvű parancsok feldolgozása.

A sorszámok nélküli parancsokat és a BASIC program egy-egy sorát csaknem azonos módon hajtja végre a rendszer.

Parancssorok begépelése esetén a végrehajtás minden esetben a 0DB61H címen folytatódik.

DB61 EB	EX DE,HL	
DB62 CD1BFA	CALL FA1B	Törli a veremből a legutoljára beírt számot
DB65 4E	LD C,(HL)	Ez még mindig csak
DB66 AF	XOR A	adminisztráció. A Parancs
DB67 77	LD (HL),A	sorból egy álprogramsort
DB68 2B	DEC HL	készít úgy, hogy a sorszám
DB69 77	LD (HL),A	helyére 00-t ír és a sor
DB6A 2B	DEC HL	hosszát 2-vel megnöveli.
DB6B 0C	INC C	Végül is egy 0000 sorszámú
DB6C 0C	INC C	programsorként kezeli
DB6D 71	LD (HL),C	tovább

Ezzel azonban az adminisztrációnak vége is. A következő utasítás már nemcsak a parancsok, hanem minden új programsor végrehajtásának is a belépési pontja.

A következő sor végrehajtása

Belépéskor a HL regiszterpárban a sor kezdőcíme található (tehát a sor hosszbyte-jára mutat)

DB6E 220C17	LD (170C),HL	A kezdőcímet elteszi
DB71 23	INC HL	Tovább lép a sorszámra
DB72 DDCB0046	BIT 0,(IX+00)	Ha a TRACE funkció be van
DB76 C44DDE	CALL NZ,DE4D	kapcsolva, kiírja a megkezdett sor sorszámát

DB79	DDCB0286	RES	O, (IX+02)	Ha korábban volt IF ág, ezt törli
DB7D	23	INC	HL	
DB7E	23	INC	HL	Az első utasításbyte-ra mutat
DB7F	D9	EXX		

Így kezdődik tehát egy-egy új sor feldolgozása. Pontosabban, az eddigi szóhasználattal élve: így kezdődik egy bekezdés feldolgozása.

A bekezdések viszont mondatokból állnak, s ezeket a BASIC-ben egy ":" karakter választja el egymástól. Valahányszor a gép végrehajt egy mondatot, utána az itt következő belépési ponthoz tér vissza, itt kezdődik minden új utasítás végrehajtása.

A következő utasítás végrehajtása

HL' az aktuálisan végrehajtandó utasításbyte címére mutat

DB80	D9	EXX		Mindig a STOP-FLAG
DB81	CD9DFF	CALL	FF9D	vizsgálatával kezd, s a
DB84	FD221A17	LD	(171A), IY	veremmutatót tárolja
DB88	7E	LD	A, (HL)	Az utasításbyte
DB89	FEFE	CP	FE	REM esetén
DB8B	302E	JR	NC, DB8E	a megfelelő címre ugrik,
DB8D	23	INC	HL	egyébként megkeresi az első
DB8E	FE20	CP	Z0	nem-szóköz utasítást
DB90	28F6	JR	Z, DB88	
DB92	FECA	CP	CA	
DB94	DABCE3	JP	C, E3BC	Ugrás a LET parancshoz!

A mondat első utasítása a végrehajtandó tevékenység főirányát meghatározó elsődleges token. Ennek megadása az értékadó LET parancs esetén nem kötelező. Ezért, ha a szóbanforgó byte 0CAH-nál kisebb (az összes elsődleges token 0C9H fölött van), akkor a rendszer úgy értelmezi, hogy a LET-et kell végrehajtania!

Ha viszont egy létező elsődleges tokent talált, akkor így folytatja:

DB97	F5	PUSH	AF	
DB98	D9	EXX		Előállítja a tokenhez
DB99	2F	CPL		tartozó végrehajtási címet
DB9A	87	ADD	A, A	
DB9B	C667	ADD	A, 67	Az alsó byte:
DB9D	6F	LD	L, A	L=2*(FFH-token)+67H, míg a
DB9E	26C0	LD	H, C0	felső byte: H=C0H

DBA0 7E	LD	A, (HL)	A kapott címről betölti a
DBA1 2C	INC	L	funkciót megvalósító rutin
DBA2 66	LD	H, (HL)	kezdőcímét
DBA3 6F	LD	L, A	

Ezt a táblázatot, mely a 0C067H címen kezdődik, már ismerjük.

DBA4 F1	POP	AF	Ha a token 0FBH-nál kisebb
DBA5 FEFB	CP	FB	(nem DATA és nem REM) akkor
DBA7 DC43FC	CALL	C, FC43	előkészíti a végrehajtást,
DBAA 31AC16	LD	SP, 16AC	majd a CPU veremét
DBAD E9	JP	HL	alaphelyzetbe állítja (!) és
			ugrás a token rutinjára!

Feltétlenül meg kell beszélnünk az itt szereplő 0FC43H kezdőcímű szubrutinnak a működését.

0FC43H lehívja és előkészíti a következő parancsbyte feldolgozását a következők szerint (ld. még a 3.6.7.2 szakaszt):

- HL-t továbblépteti, amely mindig az új parancsbyte címére mutat;
- ha az új parancs FE vagy FF: megáll. A=B'=a talált kód, CY=0;
- ha FD-t talál (a ":" tokenje, mondat vége): A=B'=FD és Z=1. A következő parancs címére áll és CY=0;
- 80...FCH között A=B'=a talált token, továbblép és CY=1;
- ha nagybetű: szimbólumként értelmezi és rááll a szimbólumtáblában (ha nincs még: létrehozza). Ekkor A=B'=1 vagy 3 aszerint, hogy a betűk után a karakterlánc jele (\$) szerepel-e, vagy sem. C'=a típusbyte, DE=az első adat címe, CY=1 (a TVC szimbólum kezeléséről később beszélünk);
- ha a következő parancsbyte " (idézőjel): a szöveget a BASIC verembe teszi és HL' továbblép, A=B'=00 lesz és CY=1;
- ha pedig az előbbieket egyike sem teljesült, akkor megkísérli számként a BASIC verembe tenni. Ha ez sikerül, akkor HL' továbblép, A=B'=02 és CY=1.

A lehívott, feldolgozás előtt álló utasításbyte-ról alapvető információkat hordozó értéket (B'-ben) a továbbiakban utasításflagnek hívjuk (flag=jelzés, jeladás).

Összefoglalva az eddigieket: láttuk, hogyan bocsátja útjára a parancsértelmező ág egy-egy elsődleges token

végrehajtását. A tokenhez tartozó funkcióhoz szükséges további adatokat az utána következő parancsbyte-ok tartalmazzák. Az előbb láttuk, hogy ezek előkészítése milyen módon történik. További részletekről majd később.

Egy-egy mondat végrehajtása után a vezérlés vagy az eddig már megbeszélte belépési pontokra kerül vissza, vagy pedig az itt következő sorok valamelyikére.

DBAE CD43FC	CALL FC43	Lehívja a következő
DBB1 D9	EXX	utasítást
DBB2 78	LD A,B	Ha vége egy mondatnak, tehát
DBB3 D9	EXX	a ":" tokenje következik,
DBB4 FEFD	CP FD	akkor ugrás a megfelelő
DBB6 28C8	JR Z,DB80	helyre, egyébként a
DBB8 DA5AFD	JF C,FD5A	hibakezelőhöz

Ilyenkor még legfeljebb a REM hatású "!"-t fogadja el -- tokenje: 0FEH vagy 0FFH -- ugyanis éppen ez következik.

A REM utasítás belépési pontja

DBBB 2A0C17	LD HL,(170C)	Az aktuális sor kezdőcíme és az azon tárolt hossz alapján veszi a következő sor kezdőcímét: átlépi tehát a sor hátralevő részét. Jöhet a következő sor, ez azonban már új belépési pont
DBBE 4E	LD C,(HL)	
DBBF AF	XOR A	
DBC0 47	LD B,A	
DBC1 09	ADD HL,BC	
DBC2 B6	OR (HL)	Az új sor 00-val kezdődik? Ha nem: ugrás a megfelelő pontra
DBC3 20A9	JR NZ,DB6E	

Ha a következő végrehajtandó BASIC programsor első byte-ja 00, ami a hosszbyte, tehát végül is nincs ilyen sor, akkor elővigyázatosnak kell lenni!

DBCS FD7E00	LD A,(IY+00)	A munka befejezése előtt a BASIC vermet helyre kell hozni. Ez a számok és szövegek esetén menet közben megtörténik. Ha azonban a veremben FOR, vagy GOSUB adatok állnak: meg kell nézni
DBCS FE2B	CP 2B	
DBCA 280E	JR Z,DBDA	
DBCC FE06	CP 06	
DBCE 280A	JR Z,DEDA	

DBD0 DDCB0056	BIT	Z,(IX+00)	Ha parancsvégrehajtás volt:
DBD4 CADADA	JF	Z,DADA	ugrás vissza,
DBD7 C30EE1	JF	E10E	programból pedig az END
			rutinon át ugyanoda
DBDA FD5E03	LD	E,(IY+03)	Ha a hívás programsorból
DBDD FD5604	LD	D,(IY+04)	történt (DE=kezdőcím), akkor
DBE0 213118	LD	HL,1831	visszatérés az END rutinon át,
DBE3 ED52	SBC	HL,DE	ha pedig a COMMAND pufferből -
DBE5 38E9	JR	C,DBD0	- tehát parancsvégrehajtás
DBE7 4F	LD	C,A	volt -- akkor a BASIC verem
DBE8 0600	LD	B,00	mutatóját a megadott
DBEA FD09	ADD	IY,BC	távolsággal visszaállítja, s a
DBEC FD221A17	LD	(171A),IY	vizsgálatot tovább folytatja
DBF0 18D3	JR	DBC5	

Ezzel, kedves Olvasó, óriási lépést tettünk a BASIC működésének megértésében. Láttuk, hogyan történik a "munka felvétele", program esetén a tárolása, de azt is, hogyan történik a munkaszervezés, a munkára bocsátás az első fázisban, s melyek az egy-egy új lépéshez tartozó belépési pontok.

Feltételezem, hogy Ön jól ismeri a BASIC nyelvű programok TV-computerben való tárolásának és a szimbólumok kezelésének mechanizmusát, így ezekről csak emlékeztető jelleggel, röviden a következőket mondom el:

- minden programsor első byte-ja a sor hossza;
- utána két byte-on a program sorszáma áll, közönséges bináris alakban;
- egymás utáni byte-okon ezután a programsor tokenjei és egyéb karakterei találhatóak;
- a sor végén OFFH áll;
- a program végét 00 byte jelzi.

Szimbólumok kezelése:

- egy-egy szimbólum megkeresését és használatát egy láncolási mechanizmus teszi lehetővé: minden szimbólum első két byte-ján a következő szimbólum címe áll;
- az első szimbólum címét a CHAIN változó tartalmazza;
- a másik szimbólumra mutató cím után az aktuális szimbólum névhossza, majd nevének karakterei állnak;
- a nevet a típusbyte követi, amelyben
 - b0=0: egyszerű szimbólum 1: tömb
 - b1=0: string 1: szám
 - b2=0: egyszerű szimbólum 1: DEF-fel megadott függvény
 - b3=0: felhasználói 1: beépített függvény;
- a típusbyte után az adatbyte-ok következnek. Ezek sajátosságait itt nem részletezzük, a TVC-hez adott programozási segédletben megtalálhatóak.

Továbbmegyünk.

Az alábbi kis szubrutint már használtuk, a "VIDEOTON" szöveget írja ki:

```
DBF2 CD79FE      CALL FE79
DBF5 08 56 49 44 45 4F 54 4F 4E
DBFE C9          RET
```

Mint már említettük, az 0FE79H szubrutin hívásakor a kiírandó szöveg karaktereit a hívás utáni byte-okon hossz+szöveg formában kell felsorolni, utána közvetlenül a program többi utasítása állhat.

A következőt byte-okon pedig a "TV COMPUTER" szöveg ASCII kódú karakterei állnak:

```
DBFF 0C 54 56 20 20 43 4F 4D 50 55 54 45 52
```

Használtuk már a következő rutint is: ez A számú szóközt ír a képernyőre.

```
DC0C 3D          DEC  A
DC0D F8          RET  M
DC0E F5          PUSH AF
DC0F CDC7FE      CALL FEC7      E szubrutin kiír egy 20H
DC12 F1          POP  AF        (szóköz) karaktert
DC13 18F7        JR   DC0C
```

A következő négy byte pedig a TVC bejelentkező képének színkódjait tárolja: innen történik a kezdőképhez a palettaregiszterek betöltése:

```
DC15 01 44 54 51      azaz: sötétkék-vörös-sárga-cián
```

3.6.5 Az elsődleges tokenek rutinjai

Egyszer már kiejtettük a szót: stílusváltás!

Már korábban is beszéltünk arról, sőt konkrét példát is láttunk rá, hogy a ROM megismerésének nem kizárólagos,

sőt sokszor nem is a legüdvösebb formája a minden utasítássor minden részletkérdésére kitérni törekvő magyarázatokkal való ellátás. A könyv terjedelmére vonatkozó és nagyon is ésszerű korlátozások pedig önmagukban is lehetetlenné tennék az ilyen értelemben vett teljességet.

A következő oldalakon sok helyütt magára hagyjuk az Olvasót, bizonyos programrészeknél csak a listát írjuk le.

A magasabb szintű működés is pontosabban nyomon kísérhető akkor, ha csak azokhoz a listarészekhez fűzünk részletesebb magyarázatot, ahol ez a tisztánlátás érdekében elkerülhetetlen.

Aprólékos munkálatokban eddig is volt már részünk, s biztosíthatom az Olvasót, hogy még ezután is lesz alkalmunk a CPU egyes utasítássorozatainak részleteivel tölteni az időt.

Most azonban nagyobb szabású célok foglalkoztatnak!

Természetesen minden esetben pontosan megadjuk azt a működési szituációt, amelyben az egyes programok a vezérlést megkapják, s ahol szükséges, a működéshez egy globális leírást is közlünk. Csupán a részletes elemzés marad el.

Ezután is kellő részletességgel beszélünk viszont mindazokról a programrészekről, amelyek az eddigiekhez képest a megértésben minőségileg új elemet képviselnek, vagy jelentőségüknél fogva nem hagyhatók el.

3.6.5.1 A segédprogramok

A következő szubrutin szerepét a BASIC parancsok és programsorok bevitelében már ismerjük. A begépelte karakterek egymás utáni feldolgozásával azonosítja a BASIC kulcsszavakat. Megkeresi a hozzá tartozó tokeneket, s ezekkel előállítja a gépben tárolt végleges formátumú BASIC sort.

Alább közöljük a megvalósító programot, de nem fűzünk külön megjegyzéseket az egyes utasításokhoz: most a végeredmény a fontos. (Ezzel szemben ajánlom minden Olvasónak a szubrutin működésének lépésenkénti nyomon követését!)

A tömörített BASIC sor előállítás (tokenizálás)

Belépéskor HL az eredeti INPUT-sor kezdőcímére mutat, kilépéskor a COMMAND pufferben előállított tömörített sor elejére (1735H).

```
DC19 FDES  
DC1B EB
```

```
PUSH IY  
EX DE,HL
```

DE=az INPUT puffercím

DC1C	FD213517	LD	IY,1735
DC20	010000	LD	BC,0000
DC23	13	INC	DE
DC24	216EDE	LD	HL,DE6E
DC27	3EFE	LD	A,FE
DC29	08	EX	AF,AF
DC2A	FD23	INC	IY
DC2C	ED531C17	LD	(171C),DE
DC30	CDBFFB	CALL	FBBF
DC33	FD7700	LD	(IY+00),A
DC36	3C	INC	A
DC37	2866	JR	Z,DC9F
DC39	78	LD	A,B
DC3A	B7	OR	A
DC3B	2804	JR	Z,DC41
DC3D	FE3A	CP	3A
DC3F	200A	JR	NZ,DC4B
DC41	1A	LD	A,(DE)
DC42	FE22	CP	22
DC44	2005	JR	NZ,DC4B
DC46	48	LD	C,B
DC47	47	LD	B,A
DC48	13	INC	DE
DC49	18DF	JR	DC2A
DC4B	CDBFFB	CALL	FBBF
DC4E	13	INC	DE
DC4F	B7	OR	A
DC50	280E	JR	Z,DC60
DC52	B8	CP	B
DC53	200B	JR	NZ,DC60
DC55	41	LD	B,C
DC56	0E00	LD	C,00
DC58	FE22	CP	22
DC5A	2808	JR	Z,DC24
DC5C	3EFD	LD	A,FD
DC5E	1828	JR	DC88
DC60	04	INC	B
DC61	05	DEC	B
DC62	20C6	JR	NZ,DC2A
DC64	AE	XOR	(HL)
DC65	87	ADD	A,A
DC66	2025	JR	NZ,DC8D
DC68	23	INC	HL
DC69	30E0	JR	NC,DC4B
DC6B	08	EX	AF,AF
DC6C	48	LD	C,B
DC6D	FEA6	CP	A6
DC6F	2808	JR	Z,DC79
DC71	FEA5	CP	A5
DC73	2804	JR	Z,DC79
DC75	FEA3	CP	A3
DC77	2002	JR	NZ,DC7B

IY a COMMAND pointerre

A tokenek képzéséhez
használt, a kulcsszavakat
megfelelő sorrendben
felsoroló táblázat

Kódkonverzió

A ":" ASCII kódja

Az idézőjel kódja

kód-átalakítás

Ezeknek nem
feleltetünk
meg semmiféle
műveletet

DC79	D608	SUB	08	
DC7B	FEFD	CP	FD	
DC7D	2809	JR	Z,DC88	
DC7F	FEFB	CP	FB	
DC81	3805	JR	C,DC88	
DC83	47	LD	B,A	
DC84	2002	JR	NZ,DC88	
DC86	063A	LD	B,3A	
DC88	FD7700	LD	(IY+00),A	
DC8B	1897	JR	DC24	
DC8D	ED5B1C17	LD	DE,(171C)	
DC91	CB7E	BIT	7,(HL)	Elmegy a tokenhez tartozó
DC93	23	INC	HL	szó végéig: ezt a 7. bit 1
DC94	28FB	JR	Z,DC91	értéke jelzi
DC96	08	EX	AF,AF	Szinkronban a tokent is
DC97	3D	DEC	A	átállítja
DC98	08	EX	AF,AF	
DC99	7E	LD	A,(HL)	Ha a végére ért,
DC9A	3C	INC	A	(A=FFH),
DC9B	2886	JR	Z,DC23	ugrás vissza!
DC9D	18AC	JR	DC4B	
DC9F	FDE5	PUSH	IY	Ha a sor végére ért, itt
DCA1	E1	POP	HL	folytatja
DCA2	23	INC	HL	
DCA3	77	LD	(HL),A	FF: sorvégjel
DCA4	113517	LD	DE,1735	
DCA7	B7	OR	A	
DCA8	ED52	SBC	HL,DE	
DCAA	EB	EX	DE,HL	
DCAB	73	LD	(HL),E	A sor hossza
DCAC	FDE1	POP	IY	
DCAE	C9	RET		

Funkcióját tekintve ugyancsak lényeges segédprogram a következő.

Új BASIC sor beillesztése a programba

Belépéskor HL az új sor kezdőcíme mutat, B a sorszám utáni első karaktert, C a sor hosszát tartalmazza.

DCAF	221C17	LD	(171C),HL	
DCB2	C5	PUSH	BC	
DCB3	CD45DD	CALL	DD45	Megkeresi az új sor helyét a
DCB6	D4E6DC	CALL	NC,DCE6	programban, ha már volt ilyen
DCB9	C1	POP	BC	számú sor, azt törli
DCBA	04	INC	B	Ha az új sor első karaktere FF,
DCBB	C8	RET	Z	nem lesz sor
DCBC	0600	LD	B,00	

DCBE CDC9DC	CALL DCC9	Megfelelő méretű helyet
DCC1 EB	EX DE,HL	csinál az új sornak
DCC2 2A1C17	LD HL,(171C)	
DCC5 EDB0	LDIR	Bemozgatja, s ezután
DCC7 1833	JR DCFC	újrainicializálja a BASIC munkakörnyezetet

Ezt az utóbbi rutint később részletesebben is megismerjük majd.

Az előbbi programrészben hívott néhány szubrutin következik.

Helyfelszabadítás az új sornak a programban

Belépéskor a HL regiszterpár a felszabadítandó terület programbeli kezdőcímét, BC a hosszát tartalmazza

DCC9 CDFCDC	CALL DCFC	Inicializál és BASIC
DCCC CD8EFC	CALL FC8E	területellenőrzéseket végez
DCCF C5	PUSH BC	
DCD0 E5	PUSH HL	
DCD1 CD41DD	CALL DD41	Megkeresi a program végét
DCD4 D1	POP DE	
DCD5 E5	PUSH HL	
DCD6 B7	OR A	
DCD7 ED52	SBC HL,DE	A mozgatandó byte-szám
DCD9 4D	LD C,L	
DCDA 44	LD B,H	
DCDB D1	POP DE	
DCDC E1	POP HL	
DCDD E5	PUSH HL	
DCDE 19	ADD HL,DE	Az új program vége
DCDF EB	EX DE,HL	A régi program vége
DCE0 03	INC BC	
DCE1 EDB8	LDDR	Elmozgatás
DCE3 23	INC HL	
DCE4 C1	POP BC	
DCE5 C9	RET	

Egy sor törlése a programban

Belépéskor a HL regiszterpárban a sor kezdőcíme

DCE6 4E	LD C,(HL)	
DCE7 0600	LD B,00	
DCE9 E5	PUSH HL	
DCEA 09	ADD HL,BC	A sor végének a címe
DCEB E5	PUSH HL	
DCEC CD41DD	CALL DD41	Az egész program végcíme

DCFE D1	POP DE	
DCF0 B7	OR A	
DCF1 ED52	SBC HL,DE	A mozgatandó hossz
DCF3 4D	LD C,L	
DCF4 44	LD B,H	
DCF5 EB	EX DE,HL	
DCF6 D1	POP DE	A sor kezdőcíme
DCF7 D5	PUSH DE	
DCF8 03	INC BC	A törlés úgy valósul meg, hogy
DCF9 EDB0	LDIR	a program további részét
DCFB E1	POP HL	viSSzamoZgatjuk

Ezzel az utasítással a rutin "átcsorog", az eddig már igen sokszor külön is meghívott, inicializáló rutinra.

A BASIC munkaterület inicializálása

Ezt a fontos szubrutint minden esetben meghívjuk, amikor a beírt programon bármilyen változás történik. Törli a BASIC vermet és a felhasználói szimbólumokat, inicializálja a véletlenszám-generátort és a DATA mutatókat.

DCFC E5	PUSH HL	
DCFD D5	PUSH DE	
DCFE C5	PUSH BC	
DCFF FD2A190B	LD IY,(0B19)	A BASIC verem törlése
DD03 210000	LD HL,0000	
DD06 FD7500	LD (IY+00),L	
DD09 220E17	LD (170E),HL	END állapot
DD0C 221417	LD (1714),HL	DATA adatmutató
DD0F 2194F0	LD HL,F094	A felhasználói szimbólumok
DD12 222417	LD (1724),HL	törlése
DD15 210917	LD HL,1709	
DD18 75	LD (HL),L	RND inicializálás
DD19 220A17	LD (170A),HL	
DD1C 2A2217	LD HL,(1722)	
DD1F 221217	LD (1712),HL	DATA sormutató
DD22 CD41DD	CALL DD41	A program végére állítja a
DD25 23	INC HL	felhasználói szimbólumok
DD26 222617	LD (1726),HL	mutatóját (TOP)
DD29 C1	POP BC	
DD2A D1	POP DE	
DD2B E1	POP HL	
DD2C C9	RET	

Az előbbi, szinte csak a listák teljessége érdekében közölt programok után, most valamivel bővebb elemzéssel ellátott szubrutinok következnek. Köztük több olyan is

lesz, mely már egy-egy tokenhez tartozó belépési pont, tehát működésével BASIC kulcsszót hajt végre.

Egy programsor kiírása

Belépéskor HL a sor kezdőcímére mutat, az előírt eszköz kijelölése (funkcióosztály) korábban már megtörtént

DD2D F5	PUSH AF	
DD2E 23	INC HL	
DD2F 5E	LD E,(HL)	A sor hosszbyte-járól a
DD30 23	INC HL	sorszámra lép és azt beolvassa
DD31 56	LD D,(HL)	DE-be
DD32 23	INC HL	Az első parancsbyte címét
DD33 E5	PUSH HL	tárolja
DD34 EB	EX DE,HL	HL-be a sorszámot teszi és
DD35 CD19FF	CALL FF19	kiírja, majd az
DD38 E1	POP HL	első byte címét véve,
DD39 CDDDFE	CALL FEDD	az egész sort listázza
DD3C F1	POP AF	Egy program listázása esetén
DD3D D0	RET NC	CY=0-val visszatér
DD3E C38EFE	JP FE5E	Más esetben, ha pl. egy hibás
		sort ír ki javításra, akkor még
		törli a sor lista utáni
		részeit, s a következő sor
		elejére áll

Többször találkoztunk már az itt következő szubrutin hívásával is. Ennek főrésze megadott számú sor helyét keresi meg a programban. Használható azonban másféle célokra is.

A program végének címe

DD41 21FFFF	LD HL,FFFF	A sor helyét megkereső
DD44 EB	EX DE,HL	szubrutinnak lehetetlen nagy
		sorszámot ad meg

Ezzel át is folyik a következő rutinra.

Adott sorszámú sor helyét keresi a programban

Belépéskor DE tartalmazza a sor számát. A végén HL-ben az a cím lesz, ahová a programban a sor kerülhet, DE az itt jelenleg kezdődő sor számát tartalmazza. Fontosabb azonban a két jelzőbit: Z=1, ha van már ugyanilyen sorszámú sor, míg carry=1, ha az új sor helye két másik sor közé esik.

DD45 2A2217	LD	HL, (1722)	A program elejének címe
DD48 010000	LD	BC, 0000	
DD4B 09	ADD	HL, BC	
DD4C 4E	LD	C, (HL)	Az aktuális sor hossza
DD4D 0C	INC	C	
DD4E 0D	DEC	C	Ha a következő sor hosszbyte-ja 00: ez a program végét jelzi, tehát a kért sor a program végére kerül
DD4F 37	SCF		
DD50 C8	RET	Z	
DD51 D5	PUSH	DE	
DD52 23	INC	HL	
DD53 5E	LD	E, (HL)	
DD54 23	INC	HL	
DD55 56	LD	D, (HL)	Beolvassa az aktuális sorszámot, majd visszamegy a hosszbyte-ra
DD56 2B	DEC	HL	
DD57 2B	DEC	HL	
DD58 E3	EX	(SP), HL	
DD59 B7	OR	A	
DD5A ED52	SBC	HL, DE	Ugyanaz?
DD5C 19	ADD	HL, DE	
DD5D EB	EX	DE, HL	A hosszbyte címét HL-be tölti, s ha a sorszám azonos: kész
DD5E E1	POP	HL	
DD5F C8	RET	Z	
DD60 30E9	JR	NC, DD4B	Ha a vizsgált sor száma még kisebb: tovább megy, egyébként így is kész
DD62 C9	RET		

3.6.5.2 A kulcsszavak

A CONTINUE utasítás kezelése következik. Először, hiba esetén a kilépési pont:

DD63 CF	RST	8	
DD64 0A		hibakód: 0AH	Cannot Continue

CONTINUE belépési pont

DD65 DDCB0056	BIT	Z, (IX+00)	Futó program?
DD69 C2B1DB	JP	NZ, DBB1	Ha igen: máris mehet!
DD6C DDCB00D6	SET	Z, (IX+00)	Ha nem: beállítja a futó program jelzést
DD70 2A0E17	LD	HL, (170E)	Ha a következő sor címe 0000, ("END" állapot), akkor a program nem folytatható: hiba!
DD73 7C	LD	A, H	
DD74 B5	OR	L	
DD75 28EC	JR	Z, DD63	

DD77 220C17	LD	(170C),HL	Egyébként ez lesz az aktuális sor, veszi a következő utasítás címét és mehet tovább!
DD7A 2A1017	LD	HL, (1710)	
DD7D C381DE	JP	DE81	

Ez volt az első igazi BASIC kulcsszó, amelyre a vezérlés a megfelelő token azonosítása alapján kerül. Gondolom az Olvasó előtt is kezd kirajzolódni a BASIC működésének egyik alapelve: a korábbi működés eredményeinek nyilvántartásával a kulcsszóhoz tartozó meghatározott utasítássorozat végrehajtása, majd állandó visszatérés az új feladatokat kijelölő belépési pontokhoz.

Három olyan kulcsszó is van, amelyek a program meghatározott soraival -- vagy akár az egészszel -- foglalkoznak. A LIST, LLIST és DELETE utasítások vonatkozhatnak az egész programra, de paraméterezhetők úgy is, hogy csak megadott sorszámok közötti részt kezeljenek. Megtehetjük, hogy csak a kezdő, vagy csak a feldolgozás befejező sorszámát adjuk meg.

E változatos lehetőségek feldolgozásának és végrehajtásának megismerése önmagában is tanulságos, a későbbiekben pedig sokszor leszünk tanúi az itt megbeszélésre kerülő módszer alkalmazásának. Tovább fokozza azonban a dolog érdekességét az, hogy a három utasítás feldolgozását gyakorlatilag ugyanaz a program végzi, s csak a kifejezetten eljárás-specifikus sorokat ugorjuk át aszerint, hogy éppen melyik utasítás végrehajtásáról van szó.

LLIST belépési pont

Itt csak egy egyszerű jelzést állítunk be arra, hogy a nyomtatóról van szó.

DD80 0E40	LD	C,40	A megcímzett funkcióosztály: a nyomtató (amiből persze még nem következik, hogy az eszköz is ez lesz!)
DD82 B1	OR	C	Fontos: Z=0 !
DD83 1803	JR	DD88	Mehet tovább -- azzal, hogy átugorjuk a LIST előkészítő sorait

LIST belépési pont

DD85 OE20	LD	C,20	A kijelölt funkciósztály
DD87 AF	XOR	A	az editor, A=0, Z=1
DD88 F5	PUSH	AF	Tehát képernyő: Z=1 nyomtató: Z=0
DD89 CDEEFB	CALL	FBEE	Az esetleges # hivatkozások kezelése

Ezt kicsit jobban meg kell beszélnünk.

A hívott szubrutin a következő utasításbyte lehívásával tisztázza, hogy a felhasználó az LLIST vagy LIST után adott-e meg perifériahivatkozást (pl. LIST#5 esetén ma kiviteli eszköz a magnetofon lesz). Ha igen, akkor ezt figyelembe véve történik a funkciósztály-kijelölés tárolása a 1705H címen, egyébként pedig a korábban beállított értékek tárolódnak.

DD8C F1	POP	AF	
DD8D 2005	JR	NZ,DD94	Ha LLIST volt: ugrás
DD8F 79	LD	A,C	
DD90 FE20	CP	20	LIST parancsnál ellenőrzi a funkciósztályt
DD92 16F6	LD	D,F6	

A LIST önálló része ezzel a semmitmondó regiszter betöltéssel zárul, amelyben természetesen nem a D=F6H eredmény a fontos, hanem hogy a beállított Z jelzőbit átmentődik a következő sorokba.

A 0DD93H címen levő F6H kód már valójában egy másik tokenhez tartozó belépési pont, míg az LLIST esetén érvényes ugrási cím, 0DD94H ismét mást jelent: SCF (37H kód)!

DELETE belépési pont

DD93 F637	OR	37	Ekkor tehát Z=0 és CY=0
-----------	----	----	-------------------------

Ettől kezdve a 3 utasítás végrehajtása közös, a szükséges megkülönböztetéseket a beállított Z és CY jelzőbitek teszik lehetővé:

LLIST	CY=1, Z=0
LIST	CY=1, Z=1 a képernyő esetén Z=0, ha más eszköz kell
DELETE	CY=0, Z=0

DD95 08	EX	AF, AF	
DD96 D9	EXX		
DD97 78	LD	A, B	Nézi a következő
DD98 D9	EXX		parancsbyte-ot
DD99 210000	LD	HL, 0000	A sorszámok elvi kezdő- és
DD9C 11FFFF	LD	DE, FFFF	végértéke
DD9F FEFD	CP	FD	
DDA1 302E	JR	NC, DDD1	Ha az utasításokhoz nem adtak meg paramétereket (:,! vagy sorvége jön), ugrás előre
DDA3 1811	JR	DDB6	Ha utánuk áll valami, akkor máshová ugrás
DDA5 D9	EXX		
DDA6 78	LD	A, B	
DDA7 D9	EXX		
DDA8 FEA4	CP	A4	A ", " tokenje?
DDAA 2807	JR	Z, DDB3	Ha igen: folytatja a feldolgozást
DDAC 08	EX	AF, AF	Egyébként pedig LLIST és
DDAD DAB1DB	JP	C, DDB1	LIST esetén továbbmegy,
DDBO C3DADA	JP	DADA	DELETE után: alaphelyzet
DDB3 CD43FC	CALL	FC43	A ", " után lehívja a következő utasítást és ha utána sorszám van, azt DE-be tölti
DDB6 CDFBDD	CALL	DDFB	
DDB9 14	INC	D	Ez történt?
DDBA 2008	JR	NZ, DDC4	Ha DE-ben szám van: ugrás "- " karakter? (sorszámok között)
DDBC FEA2	CP	AZ	
DDBE C25AFD	JP	NZ, FD5A	Ha az utasítást nem szám és nem "-" követi: hiba!
DDC1 110001	LD	DE, 0100	Ha "-" az első byte, a kezdősorszám: 0, ha pedig DE-ben már előtte is szám volt: az INC D után visszaáll az eredeti
DDC4 15	DEC	D	
DDC5 D5	PUSH	DE	Ez már a kezdősorszám
DDC6 FEA2	CP	AZ	Van "-" (kötőjel)?
DDC8 2006	JR	NZ, DDD0	Ha nincs: ugrás
DDCA CD43FC	CALL	FC43	Lehívja a következő utasítást és a sorszámot a DE-be konvertálja
DDCD CDFBDD	CALL	DDFB	HL= kezdősorszám DE= végsorszám
DDD0 E1	POP	HL	
DDD1 D5	PUSH	DE	
DDD2 CD44DD	CALL	DD44	Megkeresi a sort (ha ez nincs, a következőt)

DDD5 D1	POP DE	
DDD6 7E	LD A, (HL)	Ha a hosszbyte 0, megnézi,
DDD7 B7	OR A	hogy volt-e ", " karakter:
DDD8 28CB	JR Z, DDA5	ugrás vissza
DDDA 23	INC HL	
DDDB 4E	LD C, (HL)	BC=a talált sorszám
DDDC 23	INC HL	
DDDD 46	LD B, (HL)	Ellenőrzi, hogy elérte-e már
DDDE 2B	DEC HL	a megadott tartomány
DDDF 2B	DEC HL	zárósorszámát. Ha igen, a
DDE0 EB	EX DE, HL	további vizsgálatokhoz (", ")
DDE1 ED42	SBC HL, BC	ugrás vissza
DDE3 09	ADD HL, BC	
DDE4 EB	EX DE, HL	
DDE5 38BE	JR C, DDA5	
DDE7 D5	PUSH DE	
DDE8 08	EX AF, AF	
DDE9 3005	JR NC, DDF0	DELETE esetén ugrás
DDEB CD2DD0	CALL DD2D	LIST és LLIST-nél kiírja a
DDEE 1804	JR DDF4	sort és most ez ugorja át a
		DELETE sorait
DDF0 CDE6DC	CALL DCE6	Törli a HL című sort
DDF3 B7	OR A	Carry=0!
DDF4 08	EX AF, AF	A megkülönböztető
		jelzőbitekét tovább tárolja
DDF5 CD9DFF	CALL FF9D	CTR+ESC kezelése
DDF8 D1	POP DE	
DDF9 18DB	JR DDD6	A munka folytatása

Az előbb kétszer is felhasználtuk az alábbi kis szubrutint:

Sorszám konvertálása a DE regiszterpárba

Belépéskor az A regiszterben az előző utasításflag áll

Ha a megelőző utasításle hívás után a BASIC stackbe egy sorszám került, akkor azt DE-be tölti.

DDFB 11FFFF	LD DE, FFFF	
DDFE FE02	CP 02	Feldolgozandó szám van a
		veremben?
DE00 C0	RET NZ	Ha nem: DE=FFFF és kész
DE01 CDC3FA	CALL FAC3	Ha igen: a számot HL-be
DE04 EB	EX DE, HL	konvertálja, majd DE-be
DE05 C343FC	JP FC43	Visszatérés előtt még
		lehívja a következő
		utasítást

NEW belépési pont

A rutin egy részét már használtuk többször is a BASIC inicializálása során. Az alaphelyzethez tartozó -- már ismert -- beállításokon kívül a beírt programot érvényteleníti azzal, hogy az első byte-ra 00-t ír, valamint a szimbólumtáblával kapcsolatos adminisztrációt is itt intézi el.

DE08 21DADA	LD	HL, DADA	A visszatérési címet
DE0B E5	PUSH	HL	biztosítja
DE0C DDCB0086	RES	0, (IX+00)	Törli a TRACE funkciót
DE10 2A2017	LD	HL, (1720)	A program kezdetét a megadott
DE13 222217	LD	(1722), HL	bázisra teszi, az 1. byte 00:
DE16 3600	LD	(HL), 00	program tehát nincs
DE18 C3FCDC	JP	DCFC	Ezután a már ismert rutinon át
			ugrik DADA-ra

Látszik, hogy a program ténylegesen nem törlődik, tehát némi ügyeskedéssel még megmenthető. A báziscímen álló 00-t kell átírni, s ügyelni arra, hogy a TOP változóban elhelyezett cím most a bázis 2. byte-jára mutat. Nagy figyelemmel még az elveszettnek látszó, korábban használt szimbólumok is elérhetők. Ehhez azonban már komolyabb memóriavizsgálatokra van szükség, amelynek részleteire ennek a könyvnek a keretei között nincs lehetőség.

Megjegyzem még, hogy a NEW parancs, mint látható, csak az aktuálisan beállított bázison törli a programot. Ha a felhasználó más bázisokat is használt, azok továbbra is sértetlen állapotban vannak.

RUN belépési pont

DE1B 2AZZ17	LD	HL, (1722)	A program elejére áll
DE1E FE02	CP	02	Ha utána sorszámot is
DE20 CCDEFB	CALL	Z, FBDE	megadtak: HL erre a sorra
			mutat
DE23 CDFCDC	CALL	DCFC	A működés inicializálása és a
DE26 CD3EFC	CALL	FC3E	file-ok lezárása, majd "futó
DE29 DDCB00D6	SET	2, (IX+00)	program" jelzése után A=00-t
DE2D AF	XOR	A	és ezzel a jelzőbitek
			alaphelyzetét beállítva:
DE2E C3C2DB	JP	DBC2	indulás!

TRACE belépési pont

Ebben a csokorban utolsóként kapott helyet a TRACE utasítás. Ez után is állhat perifériakijelölés, valamint az ON és OFF kulcsszók. A végrehajtás közben minden új sor parancsainak végrehajtása előtt kiírja a megjelölt eszközre az aktuális sor számát.

DE31 CDECFB	CALL FBEC	Először a periféria hivatkozásokat (#) nézi
DE34 DD7106	LD (IX+06),C	A kijelölt funkcióosztályt a 1706H-on tárolja (az általánosan használt 1705H helyett)
DE37 FEE3	CP E3	"ON"?
DE39 280B	JR Z,DE46	Ha igen: ugrás oda
DE3B FEC1	CP C1	"OFF"?
DE3D C25AFD	JP NZ,FD5A	Ha ez sem: hiba!
DE40 DDCB0086	RES 0,(IX+00)	"TRACE OFF" jelzése és kész
DE44 1804	JR DE4A	
DE46 DDCB00C6	SET 0,(IX+00)	"TRACE ON" jelzése és így
DE4A C3AEDB	JP DBAE	is kész!

Bekapcsolt TRACE funkció esetén a parancselemző minden új sor végrehajtása kezdetén meghívja a következő rutint.

A TRACE funkció ellátása

Belépéskor HL a sorszám címére mutat

DE4D DDCB0056	BIT Z,(IX+00)	Ha nem futó programról van szó, visszatér
DE51 C8	RET Z	
DE52 3A0617	LD A,(1706)	A TRACE-ra előírt funkcióosztályt érvényesíti
DE55 320517	LD (1705),A	
DE58 E5	PUSH HL	
DE59 5E	LD E,(HL)	
DE5A 23	INC HL	
DE5B 56	LD D,(HL)	Betölti az aktuális sor sorszámát HL-be,
DE5C EB	EX DE,HL	először egy "<" jelet ír ki,
DE5D 3E3C	LD A,3C	majd
DE5F CD9AFE	CALL FE9A	formátumválasztással,
DE62 0600	LD B,00	decimálisan a sorszámot,
DE64 CD1BFF	CALL FF1B	végül egy ">" jelet
DE67 E1	POP HL	
DE68 3E3E	LD A,3E	
DE6A C39AFE	JP FE9A	

3.6.5.3 Kulcsszavak táblázata

A ROM-ban ezután a tokenek azonosításához használt kulcsszótáblázat áll. Megadjuk a kezdő ROM-címeket, a szó ASCII kódú karaktereit -- az utolsó karaktert b7=1 beállítása jelzi --, valamint a szóhoz tartozó tokent.

ROM	Tartalom	Kulcsszó	Token
DE6D	FF	sorvégjel	FF
DE6E	A1	! (REM)	FE
DE6F	BA	:	FD
DE70	52 45 CD	REM	FC
DE73	44 41 54 C1	DATA	FB
DE77	43 4C 4F 53 C5	CLOSE	FA
DE7C	43 4C D3	CLS	F9
DE7F	43 4F 4E 54 49 4E 55 C5	CONTINUE	F8
DE87	44 45 C6	DEF	F7
DE8A	44 45 4C 45 54 C5	DELETE	F6
DE90	44 49 CD	DIM	F4
DE93	45 4C 53 C5	ELSE	F3
DE97	45 4E C4	END	F2
DE9A	46 4F D2	FOR	F1
DE9D	47 45 D4	GET	F0
DEA0	47 4F 53 55 C2	GOSUB	EF
DEA5	47 4F 54 CF	GOTO	EE
DEA9	47 52 41 50 48 49 43 D3	GRAPHICS	ED
DEB1	49 C6	IF	EC
DEB3	49 4E 50 55 D4	INPUT	EB
DEB8	4C 45 D4	LET	EA
DEBB	4C 49 53 D4	LIST	E9
DEBF	4C 4C 49 53 D4	LLIST	E8
DEC4	4C 4F 41 C4	LOAD	E7
DEC8	4C 4F 4D 45 CD	LOMEM	E6
DECD	4E 45 D7	NEW	E5
DEDO	4E 45 58 D4	NEXT	E4
DED4	4F CB	OK	E3
DED6	4F CE	ON	E3
DED8	4F 50 45 CE	OPEN	E2
DEDC	4F 55 54 50 55 D4	OUTPUT	E1
DEE2	4F 55 D4	OUT	E0
DEE5	50 4C 4F D4	PLOT	DF
DEE9	50 4F 4B C5	POKE	DE
DEED	50 52 49 4E D4	PRINT	DD
DEF2	52 41 4E 44 4F 4D 45 C5	RANDOMIZE	DC
DEFB	52 45 41 C4	READ	DB
DEFF	52 45 53 54 4F 52 C5	RESTORE	DA
DF06	52 45 54 55 52 CE	RETURN	D9
DF0C	52 55 CE	RUN	D8
DF0F	53 41 56 C5	SAVE	D7
DF13	53 45 D4	SET	D6

Kulcsszavak és tokenek (folytatás)

ROM	Tartalom	Kulcsszó	Token
DF16	53 4F 55 4E C4	SOUND	D5
DF1B	53 54 4F D0	STOP	D4
DF1F	54 52 41 43 C5	TRACE	D3
DF24	56 45 52 49 46 D9	VERIFY	D2
DF2A	45 58 D4	EXT	D1
DF2D	4C 50 52 49 4E D4	LPRINT	D0
DF33	A1	!	CF
DF34	A1	!	CE
DF35	A1	!	CD
DF36	A1	!	CC
DF37	A1	!	CB
DF38	A1	!	CA
DF39	41 4E C4	AND	C9
DF3C	43 48 41 52 41 43 54 45 D2	CHARACTER	C8
DF45	44 45 4C 41 D9	DELAY	C7
DF4A	44 55 52 41 54 49 4F CE	DURATION	C6
DF52	49 4E 4B 45 59 A4	INKEY\$	C5
DF58	49 4E CB	INK	C4
DF5B	4D 4F 44 C5	MODE	C3
DF5F	4E 4F D4	NOT	C2
DF62	4F 46 C6	OFF	C1
DF65	4F 52 C4	ORD	C0
DF68	4F D2	OR	BF
DF6A	50 41 49 4E D4	PAINT	BE
DF6F	50 41 4C 45 54 54 C5	PALETTE	BD
DF76	50 41 50 45 D2	PAPER	BC
DF7B	50 49 54 43 C8	PITCH	BB
DF80	50 52 4F 4D 50 D4	PROMPT	BA
DF86	52 41 54 C5	RATE	B9
DF8A	53 54 45 D0	STEP	B8
DF8E	53 54 59 4C C5	STYLE	B7
DF93	54 41 C2	TAB	B6
DF96	54 48 45 CE	THEN	B5
DF9A	54 CF	TO	B4
DF9C	56 4F 4C 55 4D C5	VOLUME	B3
DFA2	58 4F D2	XOR	B2
DFA5	41 54 CE	ATN	B1
DFA8	41 D4	AT	B0
DFAA	55 53 49 4E C7	USING	AF
DFAF	42 4F 52 44 45 D2	BORDER	AE
DFB5	A1	!	AD
DFB6	A1	!	AC
DFB7	A1	!	AB
DFB8	A1	!	AA
DFB9	A1	!	A9
DFBA	AA	*	A8
DFBB	AS	#	A7
DFBC	3D BE	=>	A6

Kulcsszavak és tokenek (folytatás)

ROM	Tartalom	Kulcsszó	Token
DFBE	3E BC	><	A5
DFC0	AC	,	A4
DFC1	3D BC	=<	A3
DFC3	AD	-	A2
DFC4	AF	/	A1
DFC5	BB	;	A0
DFC6	DE	^	9F
DFC7	3E BD	>=	9E
DFC9	3C BE	<>	9D
DFCB	BE	>	9C
DFCC	3C BD	<=	9B
DFCE	BD	=	9A
DFCF	BC	<	99
DFD0	AB	+	98
DFD1	A6	&	97
DFD2	A8	(96
DFD3	A9)	95
DFD4	FF	táblázat vége	

Végül a számítógép néhány üzenetének karaktereit találjuk, a lezáró kódokban b7=1.

ROM	Tartalom	Üzenet	Token
DFD5	20 6D 69 73 73 69 6E E7	" missing"	94
DFDD	72 67 75 6D 65 6E F4	"rgument"	93
DFE6	42 61 64 A0	"Bad "	92
DFE8	4E 6F A0	"No "	91
DFEB	43 61 6E 6E 6F 74 A0	"Cannot "	90

Az "rgument" üzenet elejéről az "a" vagy "A" éppen amiatt hiányzik, mert előfordul, hogy kisbetűvel, máskor nagybetűvel kell kezdeni. Kiírásakor a kezdőbetűt a rendszer külön írja a szó elé.

Ezután a kulcsszavakkal és tokenekkel kapcsolatos kis kitérő után, minden eddiginél nagyobb tempóban vesszük sorra az eddig még nem tárgyalt BASIC utasításokhoz tartozó rutinokat.

3.6.5.4 Kulcsszavak rutinjai

DATA belépési pont

Ez a rutin érdemi munkát nem végez: átlépjük a DATA utáni karaktereket mindaddig, amíg új utasításhoz nem érünk a programban (amíg a ":", "!" vagy a sor végét jelző tokennel nem találkozunk).

DFF2 D9	EXX		
DFF3 7E	LD	A, (HL)	Ha a következő parancsbyte
DFF4 FEFD	CP	FD	0FD...0FFH, akkor
DFF6 D281DB	JP	NC, DBS1	folytatódik a végrehajtás,
DFF9 23	INC	HL	egyébként lépeget előre a
DFFA 18F7	JR	DFF3	memóriában

CLS belépési pont

DFFC F705	RST	30: 05	Az ismert funkcióhívást
DFFE D7	RST	10	használja, s ha nincs hiba,
DFFF C3B1DB	JP	DBB1	mehet tovább!

DEF utasítás belépési pontja

Ez is csak egyszerű adminisztrációnak tekinthető, jöllehet a létrehozott függvény beépítése a szimbólumláncba komoly munkát jelent.

Belépéskor a következő utasítást lehívó rutin már ráállt a függvény nevével azonosított szimbólumra. Létrehozta azt, hiszen feltehetően nem volt még ilyen néven más szimbólum. DE' az új szimbólumelem első adatbyte-jára mutat.

E002 E681	AND	81	Ha a DEF után nem egy
E004 FE01	CP	01	függvény neve állt,
E006 C25AFD	JP	NZ, FD5A	súlyos hiba!
E009 DDCB0056	BIT	2, (IX+00)	Futó program?
E00B 281F	JR	Z, EOZE	Ha nem: ugrás előre

Ilyenkor semmi más tennivaló nincs, mint megkeresni a következő utasítás elejét.

E00F D9	EXX		
E010 E5	PUSH	HL	A következő parancsbyte és
E011 D5	PUSH	DE	az adatbyte címe az új
E012 D9	EXX		szimbólumelemben

E013	E1	POP	HL	
E014	2B	DEC	HL	A típusbyte
E015	CB7E	BIT	7, (HL)	Ha már volt korábban is
E017	2835	JR	Z, E04E	ilyen szimbólum: hiba!
E019	CBBE	RES	7, (HL)	Törli az "új" jelzést és
E01B	CBD6	SET	2, (HL)	DEF szimbólumként jegyzi
E01D	23	INC	HL	
E01E	ED5B0C17	LD	DE, (170C)	Ezután az a databyte-okra
E022	73	LD	(HL), E	sorra lerakja az aktuális
E023	23	INC	HL	BASIC sor kezdőcímét
E024	72	LD	(HL), D	
E025	23	INC	HL	
E026	D1	POP	DE	
E027	73	LD	(HL), E	majd a függvény nevét
E028	23	INC	HL	követő első parancsbyte
E029	72	LD	(HL), D	címét
E02A	23	INC	HL	
E02B	222617	LD	(1726), HL	Beállítja a TOP új értékét
E02E	D9	EXX		
E02F	0E01	LD	C, 01	Ezután megkeresi a
E031	0B	DEC	C	következő utasítás elejét
E032	3E0C	LD	A, 0C	A függvénydefiníció két
E034	7E	LD	A, (HL)	zárójelét a DEC C és az
E035	FEFE	CP	FE	INC C figyelni, ez az LD
E037	D24BE0	JP	NC, E04B	A, 0C tehát ismét csak
E03A	23	INC	HL	álutasítás, benne az
E03B	FE96	CP	96	E033H ugrással INC C van
E03D	28F4	JR	Z, E033	A következő parancsbyte
E03F	FE95	CP	95	
E041	28EE	JR	Z, E031	A sor végén: ugrás előre
E043	FEFD	CP	FD	Ha "("-t vagy ")"-t talál,
E045	20ED	JR	NZ, E034	akkor a továbblépés előtt
E047	0C	INC	C	C-t megfelelően
E048	0D	DEC	C	változtatja
E049	20E9	JR	NZ, E034	
E04B	C381DB	JP	DB81	Amíg ":"-ot nem talál,
				vagy a zárójeleket nem
				találja korrektül
				lezártnak, előrehaladva a
				sorban keresi a kifejezés
				végét
				Ha a végére ért: mehet a
				következő utasításra

Erre és a következő kulcsszóra is vonatkozó hiba:

E04E	CF	RST	8	
E04F	0F	hibakód:	0F	Variable declared twice

A következő sor már a DIM utasításhoz tartozik:

E050 CD43FC	CALL FC43	Lehívja a következő utasítást
-------------	-----------	-------------------------------

DIM belépési pont

Ez az utasítás hasonlóan működik, mint az előző DEF, csak a struktúrája, s így a vele kapcsolatos teendők is bonyolultabbak.

E053 B7	OR A	Ha az utasításflagben b7=1: hiba
E054 FA5AFD	JP M,FD5A	
E057 CB47	BIT 0,A	
E059 CA5AFD	JP Z,FD5A	Csak szimbólum lehet
E05C D9	EXX	
E05D 79	LD A,C	
E05E E688	AND 88	Ha már régebben is definiált szimbólum: hiba
E060 28EC	JR Z,E04E	
E062 E608	AND 08	Ha beépített: új szimbólum is lehet
E064 C40BF4	CALL NZ,F40B	
E067 1B	DEC DE	
E068 1A	LD A,(DE)	Típusbyte
E069 EE81	XOR 81	
E06B 320117	LD (1701),A	
E06E 12	LD (DE),A	Végleges típusbyte
E06F 13	INC DE	
E070 3EFF	LD A,FF	Dimenziószám, először 0FFH-ra állítva
E072 12	LD (DE),A	
E073 D5	PUSH DE	
E074 13	INC DE	
E075 ED532617	LD (1726),DE	TOP
E079 D9	EXX	
E07A CD43FC	CALL FC43	A következő utasítás le- hívása
E07D DDCB014E	BIT 1,(IX+01)	
E081 2020	JR NZ,E0A3	Ha szám: ugrás
E083 FEAB	CP AB	"*" ? (szöveghossz)
E085 201C	JR NZ,E0A3	Ha nem: ugrás

A szöveghossz-deklaráció feldolgozása:

E087 CD16FB	CALL FB16	A-ban a hossz
E08A 3C	INC A	
E08B 2857	JR Z,E0E4	FF nem lehet!
E08D 3D	DEC A	
E08E 4F	LD C,A	
E08F E1	POP HL	
E090 222617	LD (1726),HL	Most egyszerű szöveges változó lesz, ezért törli a "tömb" jelzést
E093 2B	DEC HL	
E094 CB86	RES 0,(HL)	

E096	CDC1F3	CALL	F3C1	Biztosít C db byte-ot
E099	D9	EXX		
E09A	78	LD	B,A	
E09B	D9	EXX		Ha a következő parancsbyte
E09C	FEA4	CF	A4	",": folytatja a
E09E	28B0	JR	Z,E050	dimenzionálást, egyébként
E0A0	C3B4DB	JP	DBB4	jöhet a következő utasítás
E0A3	3E96	LD	A,96	
E0A5	CD54FD	CALL	FD54	Csak "("-et fogad el
E0A8	37	SCF		
E0A9	D443FC	CALL	NC,FC43	", " után itt folytatja
E0AC	CDC4FA	CALL	FAC4	HL=elemszám
E0AF	FA14FB	JP	M,FB14	Túl nagy
E0B2	23	INC	HL	
E0B3	EB	EX	DE,HL	
E0B4	2A2617	LD	HL,(1726)	
E0B7	73	LD	(HL),E	Leteszi a dimenzióhoz
E0B8	23	INC	HL	tartozó elemszámot a
E0B9	72	LD	(HL),D	szimbólumtáblába
E0BA	23	INC	HL	
E0BB	222617	LD	(1726),HL	
E0BE	EB	EX	DE,HL	Az indexet a verembe
E0BF	CD2BFA	CALL	FA2B	konvertálja
E0C2	E1	POP	HL	
E0C3	34	INC	(HL)	Növeli a dimenziószámot
E0C4	E5	PUSH	HL	A veremben összeszorozza az
E0C5	C412F5	CALL	NZ,F512	indexeket: ez adja az összes
E0C8	D9	EXX		elemszámot
E0C9	78	LD	A,B	
E0CA	D9	EXX		Ha a szám után ", " áll, akkor
E0CB	FEA4	CF	A4	még van több dimenzió: ugrás
E0CD	28DA	JR	Z,E0A9	vissza
E0CF	3E95	LD	A,95	Végül csak a ")"-et fogadja
E0D1	CD54FD	CALL	FD54	el
E0D4	E1	POP	HL	
E0D5	34	INC	(HL)	A végleges dimenziószám
E0D6	DDCBO14E	BIT	1,(IX+01)	
E0DA	200D	JR	NZ,E0E9	Ha számtömb: ugrás előre

Itt a szöveges tömbök feldolgozásával folytatja.

E0DC	FEA8	CF	A8	"*" ? (szöveg hossz)
E0DE	3E12	LD	A,12	Az alapértéket veszi
E0E0	CC16FB	CALL	Z,FB16	Ha azonban van hosszmegadás
E0E3	3C	INC	A	is, ez lesz az érvényes, ha
E0E4	CA14FB	JP	Z,FB14	nem túl nagy
E0E7	3D	DEC	A	
E0E8	4F	LD	C,A	A tömbelem hossza
E0E9	C5	PUSH	BC	

Most már csak a kellő nagyságú területet kell lefoglalni.

EOEA DF	RST 18	A veremben duplikálja az
EOEB OC	: DUP	elemek számát, s még betölt
EOEC 8526	: NZ6	8000H-t
EOEF CD93F6	CALL F693	Ha az elemek száma több:
EOF1 F2B1FC	JP P,FCB1	"No memory" hiba,
EOF4 CDC3FA	CALL FAC3	egyébként HL-be tölti
EOF7 C1	POP BC	
EOF8 E5	PUSH HL	Az elemhossz és az elemszám
EOF9 CDBBF3	CALL F3BB	alapján a megfelelő
EOFC E1	POP HL	területellenőrzésekkel
EOFD 2B	DEC HL	elvégzi a szükséges területek
EOFE 7D	LD A,L	felszabadítását
EOFF B4	OR H	
E100 20F6	JR NZ,EOF8	
E102 1895	JR E099	Ugrás vissza: folytatja
		a dimenzionálást

Az előbbi rutin működése látszólagos bonyolultsága ellenére is egészen egyszerű. Ha pl. ezt írjuk be:

```
DIM a$(2,3,4,5)*6
```

akkor nyilvánvalóan úgy kell eljárni, hogy következő szimbólumelemként a megadott névvel létre kell hozni az a\$ nevű tömböt. Ennek dimenziószáma 4, tehát le kell rakni a 4 index maximális értékét, majd kiszámítani az összes elemek számát: $2*3*4*5=120$. Végül fel kell szabadítani -- ha lehet -- 120-szor 6-6 byte-os területet.

Pontosan ezt teszi az előbbi programrész is. A benne szereplő szubrutinokról elég csak annyit tudnunk, amit róluk leírtunk.

Beszélnünk kell viszont a területfelszabadítási részben alkalmazott RST 18-as utasításról.

Korábban részletesen leírtuk már az RST 18 után írható funkcióbyte-ok értelmezését, s azt is, hogy ezek sorozatát lehet egyetlen FST 18 után láncolni.

Itt csak arra a formára hívom felt az Olvasó figyelmét, ahogyan ezeket a funkcióbyte-okat használjuk. Kérem, hogy tanulmányozza ezeket a sorokat türelmesen, mert a későbbiekben sokszor találkozunk hasonlókkal.

A fenti, könnyednek nem mondható rutin után néhány egyszerűbb következik. Ezek értelmezését könnyű szívvel az érdeklődő Olvasóra bízom.

ELSE belépési pont

```
E104 DDCBOZ46 BIT 0,(IX+OZ)
E108 CA5AFD JP Z,FD5A
E10B C3BBDB JP DBBE
```

END utasítás végrehajtása

```
E10E 210000 LD HL,0000
E111 220E17 LD (170E),HL
E114 C3DADA JP DADA
```

EXT belépési pont

```
E117 CD1BFB CALL FB1B A rutin sorszama A-ba
E11A FE07 CP 07
E11C D214FB JP NC,FB14 Hibakezelő
E11F 87 ADD A,A
E120 F5 PUSH AF
E121 D9 EXX
E122 78 LD A,B
E123 D9 EXX
E124 FEA4 CP A4 A "," tokenje
E126 2016 JR NZ,E13E
E128 CDBAFA CALL FABA HL-ben az -- esetleg egy
E12B EB EX DE,HL kifejezéssel megadott --
E12C FEA4 CP A4 érték
E12E 200E JR NZ,E13E
E130 CDBAFA CALL FABA
E133 FEA4 CP A4
E135 2007 JR NZ,E13E
E137 E5 PUSH HL
E138 CDBAFA CALL FABA
E13B 4D LD C,L
E13C 44 LD B,H A megadott input adatok
E13D E1 POP HL előálltak

E13E E3 EX (SP),HL
E13F D5 PUSH DE
E140 5C LD E,H
E141 1600 LD D,00
E143 212100 LD HL,0021
E146 19 ADD HL,DE A rutin címére mutat
E147 5E LD E,(HL)
E148 23 INC HL
E149 56 LD D,(HL) A kezdőcím
E14A 7A LD A,D
E14B B3 OR E
```


E14C CAB1DB	JP	Z, DBB1	Ha a kezdőcím 0: tovább!
E14F ED531600	LD	(0016), DE	
E153 D1	POP	DE	
E154 21B1DB	LD	HL, DBB1	
E157 E3	EX	(SP), HL	Visszatérési cím!
E158 EB	EX	DE, HL	
E159 C31500	JP	0015	A USER-rutin mehet

FOR utasítás végrehajtása

Mint tudjuk, ez az utasítás sok elemet tárol a BASIC veremben. Kicsit hosszabb is, mint az előzőek voltak, de lényegileg ugyanolyan egyszerű, mint azok. Biztos vagyok benne, hogy figyelmesen olvasva a listát, még a szűkös magyarázatok ellenére is hiánytalanul lehet követni a működést.

E15C FE03	CP	03	A következő parancsbyte csak numerikus változó lehet
E15E 2807	JR	Z, E167	
E160 E682	AND	82	
E162 C25AFD	JP	NZ, FD5A	Ha nem az: hiba!
E165 CF	RST	8	
E166 0E		hibakód: 0E	"Type mismatch" hiba
E167 CD2EF4	CALL	F42E	A ciklusváltozó első adatbyte-jára áll a szimbólumtáblában
E16A 3E9A	LD	A, 9A	
E16C CD54FD	CALL	FD54	Csak "=" jelet fogad el
E16F CD8EFC	CALL	FC8E	Területellenőrzés
E172 E5	PUSH	HL	
E173 CDA7F0	CALL	FOA7	Kiértékeli az "=" utáni kifejezést
E176 E1	POP	HL	Az értéket a szimbólumtáblába másolja (kezdőérték)
E177 E5	PUSH	HL	
E178 CD28FB	CALL	FB28	
E17B 3EB4	LD	A, B4	A következő lehívott parancsbyte csak TO lehet
E17D CD54FD	CALL	FD54	Kiértékeli a kifejezést
E180 CDA7F0	CALL	FOA7	A BASIC0 veremben két számmal lejjebb áll
E183 11EEFF	LD	DE, FFEE	Van STEP?
E186 FD19	ADD	IY, DE	Ha igen: ugrás előre
E188 FEB8	CP	B8	Ha nincs: a lépésköz 1 lesz, ezt a verembe konvertálja
E18A 2809	JR	Z, E195	
E18C 08	EX	AF, AF	
E18D 210100	LD	HL, 0001	
E190 CD2BFA	CALL	FA2B	
E193 08	EX	AF, AF	
E194 37	SCF		

E195 D4A4F0	CALL NC,FOA4	Kiértékeli a STEP utáni
E198 FEFD	CP FD	kifejezést. Ezután már csak a
E19A DA5AFD	JP C,FD5A	":" állhat, különben hiba!
E19D FDE5	PUSH IY	
E19F E1	POP HL	
E1A0 D1	POP DE	A ciklusváltozó első
E1A1 2B	DEC HL	adatbyte-jának címe a
E1A2 72	LD (HL),D	szimbólumtáblában,
E1A3 2B	DEC HL	
E1A4 73	LD (HL),E	
E1A5 ED5B0C17	LD DE,(170C)	az aktuális BASIC sor
E1A9 2B	DEC HL	kezdőcíme,
E1AA 72	LD (HL),D	
E1AB 2B	DEC HL	
E1AC 73	LD (HL),E	
E1AD D9	EXX	majd a FOR...TO...STEP utáni
E1AE E5	PUSH HL	utasítás címe a programban
E1AF D9	EXX	
E1B0 D1	POP DE	
E1B1 2B	DEC HL	
E1B2 72	LD (HL),D	
E1B3 2B	DEC HL	
E1B4 73	LD (HL),E	Ezeket mind a veremben
E1B5 2B	DEC HL	helyezi el, ami elé még a
E1B6 362B	LD (HL),2B	"2BH" azonosító byte kerül és
E1B8 E5	PUSH HL	utána mehet tovább
E1B9 FDE1	POP IY	
E1BB C3B1DB	JP DBB1	

Láthatóan jelentős adminisztrációra van szükség ahhoz, hogy ez a BASIC-ből nagyon kényelmes és egyszerű utasítás precízen működjön.

A következő utasítás, az INPUT, már kevesebb adminisztrációval dolgozik, de szerteágazóbb tevékenységet végez.

Először megismerjük, hogyan kezeli a BASIC az adatbevitel közben kapott CTRL+ESC billentyűk lenyomását.

E1BE FEF5	CP F5	Amennyiben az adatbevitel
E1C0 D7	RST 10	közben hiba lépett fel, s
E1C1 D9	EXX	eredete nem a CTRL+ESC, akkor
E1C2 2B	DEC HL	általános hibakezelés lesz
E1C3 7E	LD A,(HL)	
E1C4 FEFC	CP EC	Egyébként visszaállunk az
E1C6 20FA	JR NZ,E1C2	INPUT utasítás elejére, majd
E1C8 C3A4FF	JP FFA4	ezután ugrunk a STOP kezelő
		ágra

Az INPUT utasítás normál körülmények között a billentyűzetről kéri be az adatokat. Ez az editor hívásával biztosítható és tulajdonképpen ugyanazt a műveletet jelenti, mint a BASIC sorok bekérése általában.

Most azonban a begépelte sorban adatot vagy ezek sorozatát várjuk, amelyeket az INPUT kulcsszót követő változóban kell tárolni -- vagyis megfelelő szimbólumtáblakezelési tevékenységgel jár. Az adatok természetesen az editor által feltöltött INPUT pufferből olvashatók ki.

Látható, hogy az editor meghívását követő munka a gép oldaláról sok hasonlóságot mutat a READ utasítás végrehajtásával: ott a READ után megadott változóba töltendő adatokat korábban bevitt DATA sorok tartalmazzák, de lényegileg ugyanazt kell tenni mindkét esetben.

A két utasítás nagyfokú hasonlósága, a funkcionálisan egybeeső munkamozzanatok magyarázzák, hogy végrehajtó rutinjaik is összefonódnak. Hasonlóan ahhoz, amit már egyszer az LLIST-LIST-DELETE utasításoknál tapasztaltunk.

INPUT belépési pont

E1CB 08	EX	AF,AF	
E1CC 37	SCF		
E1CD 08	EX	AF,AF	
E1CE 21EBE2	LD	HL,E2EB	Rááll a "?" szövegre, amely a felhasználót az INPUT-ra figyelmezteti
E1D1 DD360520	LD	(IX+05),20	Az editorhoz fordul
E1D5 37	SCF		
E1D6 D443FC	CALL	NC,FC43	Új INPUT-nál fogja hívni
E1D9 FEBA	CP	BA	Van "PROMPT"?
E1DB 201F	JR	NZ,E1FC	Ha nincs: átugorja
E1DD CD43FC	CALL	FC43	"PROMPT" esetén előkészíti az utána álló szöveget: a verembe viszi és rááll a szöveg hosszbyte-jára
E1E0 CD94F2	CALL	F294	
E1E3 FDE5	PUSH	IY	
E1E5 CD21FA	CALL	FA21	
E1E8 E1	POP	HL	
E1E9 23	INC	HL	
E1EA D9	EXX		
E1EB 78	LD	A,B	A következő parancsbyte
E1EC D9	EXX		
E1ED 08	EX	AF,AF	
E1EE B7	OR	A	Z=0, CY=0
E1EF 08	EX	AF,AF	
E1F0 FEA4	CP	A4	Ha "," karaktert talál,
E1F2 28E2	JR	Z,E1D6	folytatja az adatbekérést

E1F4 FEFD	CP	FD	":"
E1F6 280C	JR	Z,E204	
E1F8 300B	JR	NC,E207	Sor vége
E1FA CF	RST	8	Egyéb parancsbyte nem értelmezhető:
E1FB 01		hibakód: 01	"Not understood"
E1FC CDFBFB	CALL	FBFB	Ha volt perifériakijelölés: érvényesíti
E1FF 28EC	JR	Z,E1ED	
E201 08	EX	AF,AF	
E202 30F6	JR	NC,E1FA	
E204 D443FC	CALL	NC,FC43	A ":" utáni byte-lehívás
E207 3A0517	LD	A,(1705)	
E20A FE20	CP	20	
E20C CC7FFE	CALL	Z,FE7F	Kiír egy szöveget, a kurzorpozíciót megjegyzi (ld. editor) majd bekér egy sort
E20F F724	RST	30: 24	Ha valami hiba volt: lekezeli
E211 CD4FFF	CALL	FF4F	
E214 20A8	JR	NZ,E1BE	
E216 23	INC	HL	Az adat memóriacímét későbbi felhasználásra tárolja
E217 221617	LD	(1716),HL	
E21A F637	OR	37	

Ez itt érdektelen, de átvezet a READ utasítás rutinjára. A második byte (37H) már ahhoz tartozik, s ott lényeges funkciója van.

READ belépési pont

E21B 37	SCF		
E21C 08	EX	AF,AF	
E21D D9	EXX		
E21E 78	LD	A,B	A következő parancsflag
E21F D9	EXX		Először nem, de később, ha majd ","-t talál, folytatja a feldolgozást
E220 37	SCF		Ha a végére ér: ugrás vissza a következő utasításra
E221 D443FC	CALL	NC,FC43	
E224 FEFD	CP	FD	
E226 D2B4DB	JP	NC,DBB4	
E229 E681	AND	81	
E22B FE01	CP	01	Ha az INPUT vagy READ után nem változónév áll: hiba!
E22D C25AFD	JP	NZ,FD5A	
E230 08	EX	AF,AF	
E231 3049	JR	NC,E27C	INPUT-nál ugrás előre

E233	2A1417	LD	HL, (1714)	DATA adatmutató
E236	7D	LD	A, L	
E237	B4	OR	H	Ha még nincs beállítva, ugrás előre
E238	282E	JR	Z, E268	
E23A	7E	LD	A, (HL)	
E23B	23	INC	HL	Megkeresi a következő nem-
E23C	FE20	CP	Z0	szóköz karaktert,
E23E	28FA	JR	Z, E23A	ill. ennek címét
E240	2B	DEC	HL	
E241	FE21	CP	Z1	Ha "!"-et talál, új DATA sort
E243	2819	JR	Z, E25E	próbál keresni
E245	FEFD	CP	FD	Ha elfogadható adat lehet:
E247	3833	JR	C, E27C	feldolgozás
E249	D9	EXX		
E24A	E5	PUSH	HL	
E24B	D5	PUSH	DE	
E24C	C5	PUSH	BC	
E24D	CDC5FC	CALL	FCC5	Elmegy a soron belüli
E250	D9	EXX		következő ":"-ig
E251	C1	POP	BC	
E252	D1	POP	DE	
E253	E1	POP	HL	
E254	D9	EXX		
E255	23	INC	HL	
E256	FEFB	CP	FB	DATA?
E258	28E0	JR	Z, E23A	Ha igen, megvan az új adat
E25A	FEFE	CP	FE	helye: ugrás vissza
E25C	38EB	JR	C, E249	Egyébként a sor végéig tovább
E25E	2A1217	LD	HL, (1712)	nézi
E261	5E	LD	E, (HL)	Ha pedig a sor végére ért,
E262	1600	LD	D, 00	akkor az aktuális DATA
E264	19	ADD	HL, DE	sormutató alapján áttér a
E265	221217	LD	(1712), HL	következő sorra
E268	2A1217	LD	HL, (1712)	Az új DATA sormutató
E26B	7E	LD	A, (HL)	Ha a DATA sormutató már a
E26C	B7	OR	A	program végére mutat (a
E26D	280B	JR	Z, E27A	hosszbyte 00), akkor hiba:
E26F	23	INC	HL	nincs elég adat!
E270	23	INC	HL	
E271	23	INC	HL	Átlépi a sorhossz- és a
E272	7E	LD	A, (HL)	sorszám-byte-okat, majd
E273	23	INC	HL	megkeresi az első nem szóköz
E274	FE20	CP	Z0	karaktert, s ugrik vissza a
E276	28FA	JR	Z, E272	sor vizsgálatához: DATA-t
E278	18DC	JR	E256	keres

Ha a program végéig sem talált adatot, akkor a következő hibakezelő ágra ugrik.

```

E27A CF          RST  8
E27B 07          hibakód: 0F          No data

```

Amikor sikerült megtalálni a beolvasandó adat címét, akkor az INPUT és READ utasítás végrehajtása itt folytatódik.

```

E27C 08          EX   AF,AF
E27D E5          PUSH HL          Az adat címe
E27E D9          EXX
E27F C5          PUSH BC          Utasításflag, típusbyte
E280 CD2FF4      CALL F42F
E283 C1          POP  BC
E284 CB49        BIT   1,C          Ha az adat szám: ugrás előre,
E286 E3          EX   (SP),HL      itt a szövegek feldolgozásával
E287 200B        JR   NZ,E294        folytatja. A BASIC verembe
E289 CDA1F8      CALL F8A1        viszi, majd bemásolja a
E28C E3          EX   (SP),HL      megfelelő szimbólumelembe és
E28D CD41FB      CALL FB41        ugrás előre
E290 181E        JR   E2B0

```

```

E292 CF          RST  8
E293 0C          hibakód: 0C          Cannot read

```

```

E294 CD8EFC      CALL FC8E        Számok          esetén          először
E297 CD14F9      CALL F914        területellenőrzés, majd a
E29A 08          EX   AF,AF        számot a verembe viszi
E29B 3009        JR   NC,E2A6     INPUT-nál ugrás előre
E29D 08          EX   AF,AF
E29E 30F2        JR   NC,E292     Ellenőrzi az adatot, s ha nem
E2A0 17          RLA              megfelelő:
E2A1 DA12F9      JP   C,F912      ugrás a hibakezelőhöz
E2A4 1806        JR   E2AC        READ-nél ugrás előre

```

```

E2A6 08          EX   AF,AF
E2A7 E5          PUSH HL
E2A8 CD7FEC      CALL EC7F        Az INPUT-tal kapott szám
E2AB E1          POP  HL          ellenőrzés után
E2AC E3          EX   (SP),HL    a BASIC veremből a
E2AD CD28FB      CALL FB28        szimbólumtáblába kerül

```

```

E2B0 E1          POP  HL          A következő adatbyte címe
E2B1 08          EX   AF,AF
E2B2 301A        JR   NC,E2CE     INPUT-nál ugrás előre

```

```

E2B4 08          EX   AF,AF
E2B5 7E          LD   A,(HL)
E2B6 23          INC  HL          Megkeresi az első nem szóköz
E2B7 FE20        CP   20          karaktert
E2B9 28FA        JR   Z,E2B5
E2BB FE20        CP   20
E2BD 280A        JR   Z,E2C9      Ha ","-t talál, akkor a címet

```

E2BF 2B	DEC	HL	
E2C0 FE21	CP	Z1	Egyébként, ha a következő
E2C2 2805	JR	Z,E2C9	karakter nem "!" vagy nem
E2C4 CB7E	BIT	7,(HL)	token: akkor hibát jelez (80H-
E2C6 CA92E2	JP	Z,E292	nál kisebb)
E2C9 221417	LD	(1714),HL	Az új adatmutatót tárolja és
E2CC 180E	JR	E2DC	ugrás előre
E2CE 08	EX	AF,AF	INPUT-nál a következő byte az
E2CF 7E	LD	A,(HL)	input pufferben
E2D0 23	INC	HL	
E2D1 FE2C	CP	ZC	Ha ",","-t talál, a címet
E2D3 2804	JR	Z,E2D9	megjegyzi, egyébként elmegy a
E2D5 3C	INC	A	bekért sor végére
E2D6 20F7	JR	NZ,E2CF	
E2D8 2B	DEC	HL	Az INPUT mutatót tárolja
E2D9 221617	LD	(1716),HL	
E2DC D9	EXX		
E2DD 78	LD	A,B	A következő utasításflag
E2DE D9	EXX		
E2DF FEFD	CP	FD	Ha az utasítás vége, akkor
E2E1 D2B4DB	JP	NC,DBB4	ugrás vissza a következő
E2E4 FEA4	CP	A4	"mondathoz",
E2E6 CA21E2	JP	Z,E221	"," karakter esetén pedig
E2E9 CF	RST	8	folytatja az adatbeolvasást
E2EA 01		hibakód: 01	
			Minden más lehetőség
			értelmetlen!

Végül az INPUT utasítás végrehajtásakor a képernyőn megjelenő "?" karakterei, a szokásos hossz+szöveg formában:

E2EB 02 3F 20 "? "

Azt hiszem egyet lehet érteni azzal, hogy az két rutinban felmerülő speciális vizsgálatok és eljárások sorozata miatt mégiscsak szerencsésebb dolog lett volna a kettőt külön választani. Hiszen amit így a programozók néhány közös utasítással megspóroltak, azt az ismétlődő feltételvizsgálatok és az ezzel járó ugrási utasítások révén el is veszítették. Nem is beszélve a működés áttekinthetőségéről!

Szerencsére ezen a programon már nem kell javítani. A magam részéről óva intem az Olvasót, hogy hasonló esetekben a funkcionális hasonlóság jegyében több ilyen rutint összevonjon!

Ezekután az IF, az ON, a GOSUB és a GOTO kulcsszavak rutinja összehasonlíthatatlanul egyszerűbbnek fog tűnni, és valójában is az.

A helykimélés érdekében -- meg a fontosabb dolgokról való figyelemelvonás elkerülésére -- ezekhez a rutinokhoz még kevesebb megjegyzést fűzünk.

IF belépési pont

E2EE	D0CB02C6	SET	0,(IX+02)	IF jelzése
E2F2	CDA7F0	CALL	FOA7	Kiértékeli a kifejezést
E2F5	FD7E06	LD	A,(IY+06)	Az eredmény
E2F8	B7	OR	A	
E2F9	08	EX	AF,AF	
E2FA	CD1BFA	CALL	FA1B	Törli a stackből
E2FD	D9	EXX		
E2FE	E5	PUSH	HL	
E2FF	D9	EXX		
E300	E1	POP	HL	Az IF utáni első byte
E301	3EB5	LD	A,B5	
E303	CD54FD	CALL	FD54	Csak a THEN-t fogadja el
E306	08	EX	AF,AF	
E307	2813	JR	Z,E31C	Ha a feltétel nem igaz:
E309	08	EX	AF,AF	ugrás az ELSE ágra
E30A	1808	JR	E314	
E30C	23	INC	HL	
E30D	E5	PUSH	HL	Lépteti az utasítás cím
E30E	D9	EXX		mutatót
E30F	E1	POP	HL	
E310	D9	EXX		
E311	CD43FC	CALL	FC43	A következő utasítás
E314	FE02	CP	02	Ha a THEN vagy ELSE után szám
E316	CAB2E3	JP	Z,E3B2	áll: ugrás GOTO-hoz
E319	C381DB	JP	DB81	Folytatja a programot
E31C	FD2A1A17	LD	IY,(171A)	A mutató eredeti értéke
E320	CDC5FC	CALL	FCC5	Megkeresi a soron belüli
E323	FEF4	CP	F4	":"-ot. Utána ELSE van?
E325	28E5	JR	Z,E30C	Ha igen: végrehajtás
E327	FEFE	CP	FE	
E329	D2BBDB	JP	NC,DBBB	Egyébként az ELSE keresését
E32C	E5	PUSH	HL	a sor végéig folytatja
E32D	D9	EXX		
E32E	E1	POP	HL	
E32F	D9	EXX		
E330	18EE	JR	E320	

ON utasítás belépési pontja

E332	DDCB02C6	SET	0, (IX+02)	IF jelzése
E336	CD1BFB	CALL	FB1E	Az ON utáni változó értéke
E339	47	LD	B, A	
E33A	D9	EXX		
E33B	78	LD	A, B	A következő utasításflag
E33C	D9	EXX		
E33D	FEEF	CP	EF	GOTO?
E33F	286E	JR	Z, E3AF	
E341	FEF0	CP	FO	GOSUB?
E343	2024	JR	NZ, E369	Ha egyik sem: hiba!
E345	CD57E3	CALL	E357	Megkeresi a feltételhez tartozó sorszámot, majd az adott sort a programban, ezt tárolja
E348	CDDEFB	CALL	FBDE	
E34B	FD2A1A17	LD	IY, (171A)	
E34F	E5	PUSH	HL	
E350	CDD1FC	CALL	FCD1	Megkeresi a következő utasítás címét és ezekkel ugrik a GOSUB rutinjára
E353	EB	EX	DE, HL	
E354	E1	POP	HL	
E355	1837	JR	E38E	

Kijelölt sorszám keresése

A rutint az ON utasítás használja arra, hogy a kapcsoló értékének megfelelően a GOTO vagy GOSUB utáni sorszámok sorozatából kiválassza az adott sorszámú elemet. Ha ilyet nem talál, ELSE ágat keres és azt hajtja végre. Ha pedig ELSE sincs, a következő utasításra áll.

E357	CD43FC	CALL	FC43	Lehívja a következő utasítást
E35A	05	DEC	B	
E35B	C8	RET	Z	
E35C	D9	EXX		
E35D	E5	PUSH	HL	
E35E	D9	EXX		
E35F	E1	POP	HL	Ha nem az első kellett, tovább nézi
E360	CD43FC	CALL	FC43	
E363	3006	JR	NC, E36B	Ha nincs több: ugrás előre, ", " esetén pedig folytatja a keresést
E365	FEA4	CP	A4	
E367	28EE	JR	Z, E357	
E369	CF	RST	8	Minden más eset értelmetlen
E36A	01		hibakód: 01	
E36B	D1	POP	DE	Ha nincs ilyen sorszám, eldobja a visszatérési címet és az ELSE funkció szerint jár el
E36C	FD2A1A17	LD	IY, (171A)	
E370	D9	EXX		
E371	CDC5FC	CALL	FCC5	
E374	FEF4	CP	F4	
E376	2894	JR	Z, E30C	

```

E378 FEFE      CP   FE
E37A D214FB    JP   NC,FB14
E37D E5        PUSH HL
E37E D9        EXX
E37F E1        POP  HL
E380 18EE      JR   E370

```

Folytatja a keresést!

GOSUB belépési pont

Az utasítás végrehajtása a verem kezelésével kapcsolatos adminisztrációk elvégzését jelenti, azt követően pedig a vezérlés átadását a meghatározott címre.

A leírt utasításokhoz semmiféle kommentárt nem szükséges fűzni.

```

E382 CDDEFB    CALL FBDE      Megkeresi az adott sorszámú
                                     sort
E385 CD43FC    CALL FC43      A következő utasítás
E388 38DF      JR   C,E369    Ez csak új utasítás lehet,
                                     különben: hiba
E38A D9        EXX
E38B E5        PUSH HL
E38C D9        EXX
E38D D1        POP  DE
E38E E5        PUSH HL
E38F CD8EFC    CALL FC8E      Területellenőrzés
E392 FDE5      PUSH IY
E394 E1        POP  HL
E395 2B        DEC  HL
E396 3A0217    LD   A,(1702)   ON GOSUB-nál ELSE van-e?
E399 77        LD   (HL),A
E39A ED4B0C17 LD   BC,(170C)  Az aktuális sor kezdőcíme
E39E 2B        DEC  HL
E39F 70        LD   (HL),B
E3A0 2B        DEC  HL
E3A1 71        LD   (HL),C
E3A2 2B        DEC  HL
E3A3 72        LD   (HL),D      A GOSUB utáni utasítás címe
E3A4 2B        DEC  HL
E3A5 73        LD   (HL),E
E3A6 2B        DEC  HL
E3A7 3606      LD   (HL),06    GOSUB azonosító
E3A9 E5        PUSH HL
E3AA FDE1      POP  IY
E3AC E1        POP  HL
E3AD 180A      JR   E3B9      Átadja a vezérlést

```

Ez a cím közvetlenül még csak a GOTO rutinba mutat, azonban ott ugyanilyen vezérlésátadásról van szó. A tényleges ugrási cím a RUN-nál már megismert sorokra mutat.

A GOTO utasítás rutinja következik. Előfordul, hogy a vezérlést nem sorszámmal, hanem indexszel kell átadni, mint az előbbi ON rutinból.

Ehhez az utasítás végrehajtását a megfelelő segédrutin készíti elő, amit már ismerünk.

E3AF CD57E3	CALL E357	Az indexből a megfelelő sorszámot választja ki a GOTO számára
-------------	-----------	---

GOTO belépési pont

E3B2 CDDEFB	CALL FBDE	Megkeresi az adott számú sort a programban és átadja a vezérlést
E3B5 FD2A1A17	LD IY, (171A)	
E3B9 C329DE	JP DE29	

LET utasítás feldolgozása

Megengedett, hogy az értékadó utasításokban a LET szó elmaradjon. Ebben az esetben a parancselemző, ha tokent nem talál, automatikusan LET-ként dolgozza fel a parancsot. Ilyenkor azonban néhány előkészítő műveletet külön el kell végezni.

Kulcsszó nélküli belépés:

E3BC 2B	DEC HL	Előkészíti a változót
E3BD D9	EXX	
E3BE CD43FC	CALL FC43	

LET kulcsszó megadása esetén itt lépünk be:

E3C1 E6FD	AND FD	Ha a következő utasítás nem változónév: hiba!
E3C3 FE01	CP 01	
E3C5 C25AFD	JP NZ,FD5A	DEF azonosító nem lehet!
E3C8 D9	EXX	
E3C9 CB51	BIT 2,C	A beépített függvény újradefiniálható
E3CB C25AFD	JP NZ,FD5A	
E3CE CB59	BIT 3,C	Az adat helye a szimbólumtáblában
E3D0 C40BF4	CALL NZ,F40B	
E3D3 DD7101	LD (IX+01),C	A lehívott utasításflag
E3D6 D9	EXX	
E3D7 CD2EF4	CALL F42E	
E3DA D9	EXX	
E3DB 78	LD A,B	
E3DC D9	EXX	

E3DD E5	PUSH HL	Az adat helye
E3DE CDE8E3	CALL E3E8	Kiértékeli az "=" utáni
E3E1 E1	POP HL	kifejezést és beteszi a
E3E2 CD3BFB	CALL FB3B	szimbólumtáblába
E3E5 C3B1DB	JP DBB1	Mehet tovább

Kifejezés értékelése a LET végrehajtásához

E3E8 D696	SUB 96	A "(" tokenje
E3EA 205E	JR NZ, E44A	Ha nem "("-lel kezdődik,
		akkor ugrik az "="
		feldolgozásához
E3EC CB49	BIT 1, C	
E3EE C25AFD	JP NZ, FD5A	Ha nem szöveg-változó: hiba

Szövegszelet értékadása

A megadott rész-sztring és a változó aktuális tartalma alapján a BASIC veremben állítjuk össze a teljes szöveget

E3F1 23	INC HL	
E3F2 E5	PUSH HL	
E3F3 4E	LD C, (HL)	
E3F4 47	LD B, A	A szöveg teljes hossza
E3F5 CD27FD	CALL FD27	A zárójelben megadott
E3F8 7A	LD A, D	szeletparamétereket adja
E3F9 BB	CP E	vissza DE-ben: E=kezdőérték,
E3FA 3002	JR NC, E3FE	D=végérték
E3FC 7B	LD A, E	
E3FD 3D	DEC A	
E3FE 57	LD D, A	Ellenőrzi a megadott
E3FF 79	LD A, C	paraméterek viszonyát, s a
E400 92	SUB D	szeletképzést az ismert
E401 E1	POP HL	BASIC szabályok szerint
E402 FDE5	PUSH IY	szervezi meg
E404 E5	PUSH HL	
E405 D5	PUSH DE	
E406 C5	PUSH BC	
E407 380E	JR C, E417	
E409 280C	JR Z, E417	
E40B 09	ADD HL, BC	
E40C 4F	LD C, A	
E40D FDE5	PUSH IY	
E40F D1	POP DE	
E410 1B	DEC DE	
E411 EDB8	LDDR	A szöveg végét másolja,
E413 13	INC DE	
E414 D5	PUSH DE	
E415 FDE1	POP IY	majd a megadott részt

```

E417 CD4AE4      CALL E44A
E41A C1          POP  BC
E41B D1          POP  DE
E41C E1          POP  HL
E41D 7B          LD   A,E
E41E FDE5        PUSH IY
E420 D1          POP  DE
E421 13          INC  DE
E422 FE02        CP   02
E424 380C        JR   C,E432
E426 3D          DEC  A
E427 B9          CP   C
E428 3001        JR   NC,E42B
E42A 4F          LD   C,A
E42B 09          ADD  HL,BC
E42C 0C          INC  C
E42D 0D          DEC  C
E42E 2802        JR   Z,E432
E430 EDB8        LDDR
E432 E1          POP  HL
E433 2B          DEC  HL
E434 B7          OR   A
E435 ED52        SBC  HL,DE
E437 24          INC  H
E438 25          DEC  H
E439 C212F9      JP   NZ,F912
E43C 2C          INC  L
E43D CA12F9      JP   Z,F912
E440 2D          DEC  L
E441 EB          EX  DE,HL
E442 73          LD  (HL),E
E443 2B          DEC  HL
E444 3601        LD  (HL),01
E446 E5          PUSH HL
E447 FDE1        POP  IY
E449 C9          RET

```

A szöveg elejének másolása

"Overflow" hiba

A karakterlánc hossza

A szöveg azonosítója

A "=" után álló kifejezés értékelése

```

E44A 3E9A        LD   A,9A
E44C CD54FD      CALL FD54
E44F C38DF2      JP   F28D

```

Csak az "=" tokenje lehet a
következő lehívott
utasításbyte
Értékeli a kifejezést és
visszatér

A következő kulcsszavak megvalósítása -- néhány később megtárgyalásra kerülő szubrutin hivatkozástól eltekintve -- ugyancsak annyira egyszerű, hogy kis figyelemmel és türelemmel, önállóan és részletesen kidolgozhatók az egyes sorok funkciói.

LOMEM belépési pont

E452 CDC4FA	CALL FAC4	HL-ben előállítja a megadott számot
E455 ED5B2017	LD DE, (1720)	
E459 B7	OR A	
E45A ED52	SBC HL, DE	
E45C 19	ADD HL, DE	
E45D DA14FB	JP C, FB14	Ha a bázisnál is kisebb: hiba!
E460 ED5B2217	LD DE, (1722)	
E464 ED52	SBC HL, DE	Ha TEXT és a beírt szám azonos: semmit sem kell csinálni
E466 CAB4DB	JP Z, DBB4	
E469 E5	PUSH HL	
E46A DC86FC	CALL C, FC86	Ha a távolság negatív: szorzás (-1)-gyel
E46D 4D	LD C, L	
E46E 44	LD B, H	
E46F EB	EX DE, HL	Ha hátra kell mozgatni: ugrás!
E470 3813	JR C, E485	
E472 D1	POP DE	
E473 D5	PUSH DE	
E474 E5	PUSH HL	
E475 14	INC B	Ha a távolság nagy: hiba
E476 CA14FB	JP Z, FB14	Területellenőrzés
E479 CD99FC	CALL FC99	
E47C E1	POP HL	
E47D E5	PUSH HL	Mozgatás előre
E47E CDC9DC	CALL DCC9	
E481 E1	POP HL	Az új TEXT
E482 09	ADD HL, BC	
E483 1808	JR E48D	
E485 B7	OR A	
E486 ED42	SBC HL, BC	
E488 E5	PUSH HL	
E489 CDE9DC	CALL DCE9	Mozgatás vissza
E48C E1	POP HL	
E48D 222217	LD (1722), HL	Az új TEXT tárolása
E490 C1	POP BC	
E491 DDCB0056	BIT Z, (IX+00)	Futó program
E495 280C	JR Z, E4A3	Ha nem: ugrás előre!
E497 2A0C17	LD HL, (170C)	
E49A 09	ADD HL, BC	
E49B 220C17	LD (170C), HL	Az aktuális sor címét is átállítja
E49E C5	PUSH BC	
E49F D9	EXX	
E4A0 D1	POP DE	
E4A1 19	ADD HL, DE	A következő végrehajtandó utasításbyte címét is átírja,
E4A2 D9	EXX	majd inicializálja a szimbólumtáblát és kész
E4A3 CDFCDC	CALL DCFC	
E4A6 C3B1DB	JP DBB1	

Jelentős munkálatokat végez a NEXT BASIC utasítás. A továbbiakban ezzel ismerkedhet meg az Olvasó. A tényleges belépési pontot megelőzi a hozzá tartozó hibakilépő ág:

```
E4A9 CF          RST  8
E4AA 08          hibakód: 08          No for
```

Utána következik a többszörös ciklusokat egymás után lezáró vezérlőhurok. Egyetlen NEXT után több ciklusváltozó állhat, egymástól vesszővel elválasztva.

```
E4AB D9          EXX
E4AC 78          LD   A,B          A következő utasításflag
E4AD D9          EXX
E4AE FEA4        CP   A4          Ha nem vessző: a vezérlés a
E4B0 C2B4DB      JP   NZ,DBB4        következő utasítással,
E4B3 CD43FC      CALL FC43        egyébként az új NEXT
                                        kezelésével folytatódik
```

NEXT belépési pont

```
E4B6 DDCB00D6   SET  Z,(IX+00)        Futó program jelzése
E4BA 3027        JR   NC,E4E3        Ha a NEXT után nincs változó:
E4BC FE03        CP   03          ugrás (lehet!)
E4BE 2807        JR   Z,E4C7
E4C0 E682        AND  82          Ha nem numerikus változó:
E4C2 C25AFD      JP   NZ,FD5A        hiba!

E4C5 CF          RST  8
E4C6 0E          hibakód: 0E          Type mismatch

E4C7 CD2EF4      CALL F42E        A változó értékére mutat HL, a
E4CA EB          EX  DE,HL        szimbólumtáblában
E4CB 21FD09      LD   HL,09FD      Ez most érdektelen, de
                                        legközelebb a 0E4CCH-ra ugrik,
                                        s ekkor már:

E4CC FD09        ADD  IY,BC        ez lesz az értelme
E4CE 3E2B        LD   A,2B        Megkeresi ezt az azonosítót a
E4D0 CDE7FC      CALL FCE7        BASIC veremben
E4D3 38D4        JR   C,E4A9        Ha nincs: hiba!
E4D5 0E05        LD   C,05
E4D7 09          ADD  HL,BC        Az aktuális érték címe
E4D8 0E2B        LD   C,2B
E4DA 7E          LD   A,(HL)
E4DB BB          CP   E          Ellenőrzi, hogy a veremben
E4DC 20EE        JR   NZ,E4CC      talált ciklusterület aktuális
E4DE 23          INC  HL          értékének címe a keresettel
E4DF 7E          LD   A,(HL)      azonos-e. Ha nem, tovább
E4E0 BA          CP   D          keres: ugrás vissza
E4E1 20E9        JR   NZ,E4CC
```

Ha ez a keresett FOR terület vagy nem volt megadva ciklusváltozó, itt folytatja.

E4E3	3E2B	LD	A,2B	
E4E5	CDE7FC	CALL	FCE7	Rááll a területre
E4E8	38BF	JR	C,E4A9	(ha nincs: hiba)
E4EA	23	INC	HL	
E4EB	5E	LD	E,(HL)	
E4EC	23	INC	HL	
E4ED	56	LD	D,(HL)	
E4EE	D5	PUSH	DE	A ciklusfej utáni cím
E4EF	23	INC	HL	
E4F0	5E	LD	E,(HL)	
E4F1	23	INC	HL	
E4F2	56	LD	D,(HL)	
E4F3	D5	PUSH	DE	A sor kezdőcíme
E4F4	23	INC	HL	
E4F5	5E	LD	E,(HL)	
E4F6	23	INC	HL	
E4F7	56	LD	D,(HL)	
E4F8	D5	PUSH	DE	A változó aktuális értékének
E4F9	23	INC	HL	címe a szimbólumtáblában
E4FA	EB	EX	DE,HL	
E4FB	112200	LD	DE,0022	
E4FE	FD19	ADD	IY,DE	
E500	CD63FA	CALL	FA63	Az aktuális értéket a
E503	1B	DEC	DE	végérték elé teszi,
E504	21F7FF	LD	HL,FFF7	
E507	19	ADD	HL,DE	
E508	010900	LD	BC,0009	hozzá még bemásolja a
E50B	EDB8	LDDR		lépésközt,
E50D	13	INC	DE	
E50E	D5	PUSH	DE	
E50F	FDE1	POP	IY	
E511	FDCB087E	BIT	7,(IY+08)	
E515	F5	PUSH	AF	ennek előjelét megjegyzi és a
E516	CD93F4	CALL	F493	kettőt összeadva kapja az új
E519	F1	POP	AF	aktuális értéket
E51A	E1	POP	HL	
E51B	F5	PUSH	AF	
E51C	FDE5	PUSH	IY	Az új aktuális értéket a
E51E	CD28FB	CALL	FB28	szimbólumtáblába teszi
E521	FDE1	POP	IY	
E523	CD93F6	CALL	F693	Összehasonlítja a végértékkel
E526	C1	POP	BC	
E527	D1	POP	DE	Ezt a lépésköz előjele
E528	E1	POP	HL	figyelembevételével teszi (C-
E529	79	LD	A,C	be PUSH AF-ből) az F regiszter
E52A	17	RLA		került)
E52B	3006	JR	NC,E533	Ha ciklus lefutott, ugrás az
E52D	2807	JR	Z,E536	esetleges másik ciklus
E52F	F2ABE4	JP	P,E4AB	lezárásához

E532 AF	XOR	A	
E533 FAABE4	JP	M,E4AB	Ugrás másik ciklushoz
E536 ED530C17	LD	(170C),DE	Újra ez a sor lesz aktuális,
E53A 11D5FF	LD	DE,FFD5	visszaállítja a BASIC
E53D FD19	ADD	IY,DE	veremmutatót és
E53F C381DB	JP	DBB1	kész

OUT belépési pont

E542 CD1BFB	CALL	FB1B	A beírt számot A-ba
E545 4F	LD	C,A	konvertálja: port-cím
E546 3EA4	LD	A,A4	A következő parancsbyte csak
E548 CD54FD	CALL	FD54	"," lehet
E54B CD1BFB	CALL	FB1B	A másik számot is A-ba teszi,
E54E ED79	OUT	(C),A	kiküldi a portra,
E550 C3B1DB	JP	DBB1	s mehet tovább

POKE utasítás végrehajtása

E553 CDFFFC	CALL	FCFF	Ellenőrzi a címet
E556 3EA4	LD	A,A4	A cím után csak "," állhat
E558 CD54FD	CALL	FD54	
E55B CD1BFB	CALL	FB1B	A másik szám A-ban
E55E 4F	LD	C,A	
E55F CDE7FF	CALL	FFE7	Ha VID-t címeztek, akkor
E562 3803	JR	C,E567	átállítja a címet és
E564 F3	DI		belapozza az U0-U1-VID-SYS
E565 D302	OUT	(02),A	konfigurációt
E567 71	LD	(HL),C	Ez a "POKE" művelet
E568 3E70	LD	A,70	Visszalapozás
E56A D302	OUT	(02),A	
E56C FB	EI		
E56D C3B1DB	JP	DBB1	és tovább

Az Olvasó a kézikönyvekből tudja, hogy a BASIC a videomemóriát nem lapozással, hanem POKE utasítással, a 0C000H-nál nagyobb címeket beírva éri el. Ennek kivitelezését láttuk itt.

Ugyanezt az eljárást használjuk a fordított esetben is. Amikor a BASIC felismeri, hogy 0C000H fölötti címeket adtak meg, akkor belapozza a VID-et, s a címet a megfelelő 8000H...0BFFFH tartományba konvertálja.

Hatalmasabb munka kezdődik ezután: az LPRINT és a PRINT utasítások végrehajtása.

LPRINT belépési pont

Ez egyetlen önálló utasítást tartalmaz, ill. még egy felet, amely a már megszokott trükkel, ártalmatlanul átvezet a másik utasításba:

E570 0E40	LD	C,40	A nyomtató funkcióosztály kijelölése
E572 110E20	LD	DE,200E	Ez itt semmi, de általa átlépte a PRINT elejét

PRINT belépési pont

E573 0E20	LD	C,20	A kijelölt osztály az editor
E575 DD7105	LD	(IX+05),C	
E578 08	EX	AF,AF	
E579 AF	XOR	A	
E57A 08	EX	AF,AF	
E57B 37	SCF		A későbbi visszatérések alkalmával lehívja a következő utasítást
E57C D443FC	CALL	NC,FC43	USING van?
E57F FEAF	CP	AF	Ha nem: ugrás előre
E581 2023	JR	NZ,E5A6	

USING feldolgozása

E583 CD43FC	CALL	FC43	Feldolgozza az USING karakterláncot
E586 CD94F2	CALL	F294	
E589 213118	LD	HL,1831	
E58C 367F	LD	(HL),7F	
E58E E5	PUSH	HL	
E58F CD41FB	CALL	FB41	Az USING karakterláncot a 1832H pufferbe másolja
E592 E1	POP	HL	
E593 23	INC	HL	
E594 5E	LD	E,(HL)	
E595 1600	LD	D,00	DE=USING hossz
E597 23	INC	HL	
E598 222E17	LD	(172E),HL	Az 1. USING byte címe
E59B 19	ADD	HL,DE	
E59C 72	LD	(HL),D	az USING utáni címre 00, a címet tárolja
E59D 223017	LD	(1730),HL	
E5A0 08	EX	AF,AF	USING jelzése
E5A1 C0C7	SET	O,A	
E5A3 08	EX	AF,AF	Ugrás előre
E5A4 1814	JR	E5BA	
E5A6 FEB0	CP	B0	AT van?
E5A8 2022	JR	NZ,E5CC	Ha nem: ugrás előre

Itt az AT feldolgozásával folytatja.

AT feldolgozása:

E5AA CD16FB	CALL FB16	A sorszám előállítás
E5AD 4F	LD C,A	
E5AE 3EA4	LD A,A4	
E5B0 CD54FD	CALL FD54	Utána csak "," állhat
E5B3 CD1BFB	CALL FB1B	Az oszloppozíció A-ban
E5B6 47	LD B,A	
E5B7 CD8BE6	CALL E68B	Beállítja a pozíciót
E5BA D9	EXX	
E5BB 78	LD A,B	
E5BC D9	EXX	
E5BD 08	EX AF,AF	
E5BE B7	OR A	CY=0; Z=0: USING
E5BF 08	EX AF,AF	
E5C0 FEA4	CP A4	Ezután csak ",", ":", "!"
E5C2 28B8	JR Z,E57C	és 0FFH állhat, különben
E5C4 FEFD	CP FD	hiba!
E5C6 280E	JR Z,E5D6	
E5C8 300F	JR NC,E5D9	
E5CA CF	RST 8	
E5CB 01	hibakód: 01	Not understood
E5CC CD00FC	CALL FC00	A perifériakijelölések
E5CF 28EC	JR Z,E5BD	kezelése: ha van
E5D1 08	EX AF,AF	
E5D2 38F6	JR C,E5CA	Itt CY=1 nem lehet!
E5D4 08	EX AF,AF	
E5D5 37	SCF	
E5D6 D443FC	CALL NC,FC43	A következő utasítás
E5D9 08	EX AF,AF	lehívása
E5DA 37	SCF	
E5DB 08	EX AF,AF	
E5DC 08	EX AF,AF	
E5DD CB47	BIT 0,A	
E5DF F5	PUSH AF	
E5E0 C49CE6	CALL NZ,E69C	Az USING első, nem
E5E3 F1	POP AF	formátumkaraktereit kiírja
E5E4 08	EX AF,AF	
E5E5 D9	EXX	
E5E6 78	LD A,B	
E5E7 D9	EXX	
E5E8 FEA4	CP A4	"," parancs?
E5EA 2006	JR NZ,E5F2	Ha nem: ugrás tovább
E5EC 3E09	LD A,09	
E5EE CD9AFE	CALL FE9A	Tabulál
E5F1 C2FEA0	JP NZ,A0FE	Ez így értelmetlen, de nem is
		valószínűleg soha!

Az igazi ugrási cím, mint láttuk: 0E5F2H.

E5F2 FEA0	CP	AO	";" parancs?
E5F4 2008	JR	NZ,E5FE	Ha nem: tovább!
E5F6 CD43FC	CALL	FC43	Lehívja a következő utasítást
E5F9 08	EX	AF,AF	
E5FA B7	OR	A	
E5FB C37FE6	JP	E67F	Ez tulajdonképpen egy visszaugrás, további kiírások vizsgálatára
E5FE FEB6	CP	B6	TAB parancs?
E600 2015	JR	NZ,E617	Ha nem: tovább!
E602 3E96	LD	A,96	"("
E604 CD51FD	CALL	FD51	A feldolgozás abból áll, hogy kiértékeli a zárójelek közötti részt.
E607 CD1BFB	CALL	FB1B)"
E60A 47	LD	B,A	Ugyanaz a sor;
E60B 3E95	LD	A,95	pozícionál
E60D CD54FD	CALL	FD54	Ez kész, tovább!
E610 0E00	LD	C,00	
E612 CD8BE6	CALL	E68B	
E615 1866	JR	E67D	
E617 FEFD	CP	FD	Ha csak magányos PRINT volt: soremelés lesz
E619 3068	JR	NC,E683	
E61B 08	EX	AF,AF	USING?
E61C CB47	BIT	0,A	
E61E F5	PUSH	AF	
E61F 2817	JR	Z,E638	Ha nem volt: ugrás!
E621 08	EX	AF,AF	
E622 2A2E17	LD	HL,(172E)	Az USING még nem használt eleje és a vége
E625 ED5B3017	LD	DE,(1730)	
E629 B7	OR	A	
E62A ED52	SBC	HL,DE	
E62C 380B	JR	C,E639	Ha még nem kész: ugrás
E62E 213318	LD	HL,1833	
E631 222E17	LD	(172E),HL	
E634 CD9CE6	CALL	E69C	Egyébként a nem formátumbyte-ok kiírása
E637 08	EX	AF,AF	
E638 08	EX	AF,AF	
E639 D9	EXX		
E63A 78	LD	A,B	
E63B D9	EXX		
E63C FE02	CP	02	Szövegkiírás?
E63E 3025	JR	NC,E665	Ha szám: ugrás előre
E640 CD94F2	CALL	F294	Elvégzi a megengedett szövegműveleteket, s ha hiba van, visszatér
E643 17	RLA		
E644 3023	JR	NC,E669	
E646 F1	POP	AF	
E647 2190FF	LD	HL,FF9C	Ha volt USING, akkor az EXT ROM-ban folytatja!
E64A C2F0FF	JP	NZ,FFFO	

E64D	F6FF	OR	FF	
E64F	FDE5	PUSH	IY	USING után ide tér vissza
E651	E1	POP	HL	
E652	23	INC	HL	
E653	F5	PUSH	AF	
E654	CD7FFE	CALL	FE7F	Kiírja az összeállított
E657	CD21FA	CALL	FA21	szöveget, majd a vermet
E65A	F1	POP	AF	helyreállítja
E65B	3D	DEC	A	
E65C	FA7DE6	JP	M,E67D	USING után még kellő számú
E65F	F5	PUSH	AF	szóközt ír
E660	CDC7FE	CALL	FEC7	
E663	18F5	JR	E65A	
E665	CDA7F0	CALL	FOA7	Kiértékeli a számkifejezést
E668	17	RLA		
E669	D25AFD	JP	NC,FD5A	Hibakezelő
E66C	F1	POP	AF	
E66D	21B5FC	LD	HL,FCB5	USING esetén végrehajtás az
E670	C2F0FF	JP	NZ,FFFO	EXT ROM-ban
E673	FDE5	PUSH	IY	
E675	E1	POP	HL	
E676	23	INC	HL	
E677	CDBAFE	CALL	FEBA	A számot a veremből a 1930H
E67A	CD1BFA	CALL	FA1B	pufferbe konvertálja, majd
				kiírja
E67D	08	EX	AF,AF	
E67E	37	SCF		
E67F	08	EX	AF,AF	Ezután folytatja a
E680	C3DCE5	JP	E5DC	feldolgozást

Ezen a rutinon jól látszik, hogy teljesen pontos értelmezése csakis a benne előforduló összes egyéb hivatkozás gondos elemzésével történhet meg. Ez többnyire nem jelent kifejezett felhasználói igényt. Annál is inkább, mert e rutin túlságosan komplex ahhoz, hogy egyedi célokra érdemes lenne -- a BASICS-től függetlenül -- használni.

Ide tartozik még a sosemeléssel megvalósuló kilépés a rutinból.

E683	08	EX	AF,AF	
E684	DC93FE	CALL	C,FE93	Ha CY=1: kiír egy 0DH és 0AH
E687	08	EX	AF,AF	karaktert és mehet tovább
E688	C3B4DB	JP	DBE4	

Mint az AT, mint a TAB parancs használja a következő pozícióbeállító rutint:

Pozícionálás

E68B	3A0517	LD	A,(1705)	A kijelölt funkcióosztály
E68E	FE20	CP	20	
E690	2803	JR	Z,E695	A kiíratás csak az editorral
E692	FE00	CP	00	vagy a videoval történhet.
E694	C0	RET	NZ	Ekkor normál funkcióhívással
E695	F603	OR	03	és azt követő
E697	CD1B00	CALL	001B	hibavizsgálattal oldja meg a
E69A	D7	RST	10	feladatot
E69B	C9	RET		

USING segédrutin

A korábban a 1831H kezdőcímű pufferbe helyezett USING szöveg karaktereit a standard formátumbyte-okkal összehasonlítva, az első nem formátumbyte-okat kiírja, a lánc még fel nem használt első karakterének címét pedig tárolja.

E69C	ED5B3017	LD	DE,(1730)	Az USING szöveg végének és
E6A0	2A2E17	LD	HL,(172E)	elejének címe
E6A3	7E	LD	A,(HL)	Egy megadott USING byte
E6A4	B7	OR	A	
E6A5	ED52	SBC	HL,DE	
E6A7	D0	RET	NC	Ha már kész: RET
E6A8	19	ADD	HL,DE	
E6A9	E5	PUSH	HL	
E6AA	21BDE6	LD	HL,E6BD	A formátumbyte-ok
E6AD	010A00	LD	BC,000A	felsorolása, 10 byte-on
E6B0	EDB1	CPIR		Keresi a karaktert
E6B2	E1	POP	HL	
E6B3	C8	RET	Z	Ha van ilyen, visszatér,
E6B4	CD9AFE	CALL	FE9A	egyébként kiírja
E6B7	23	INC	HL	
E6B8	22ZE17	LD	(172E),HL	A következő cím mentése
E6BB	18E6	JR	E6A3	Folytatja a vizsgálatot

Az alkalmazható standard formátumbyte-ok táblázata:

ROM	E6BD	E6BE	E6BF	E6C0	E6C1	E6C2	E6C3	E6C4	E6C5	E6C6
kód	3C	3E	23	2A	25	2B	2D	24	5E	2E
form.	<	>	#	*	%	+	-	\$	^	.

Ezek a ROM területek voltak tehát a kiírással kapcsolatos BASIC végrehajtó programok hordozói.

Én úgy vélem, hogy ami még ebből a részfejezetből hátra van, azon könnyűszerrel túltehetjük magunkat.

RANDOMIZE belépési pont

E6C7 ED5F	LD	A,R	A frissítő regiszterrel,
E6C9 320917	LD	(1709),A	valamint a megszakítások
E6CC 2A1D0B	LD	HL,(0B1D)	kezelésénél használt IT-
E6CF 220A17	LD	(170A),HL	számlálóval inicializálja a
E6D2 CDD8E6	CALL	E6D8	véletlenszám-generátort
E6D5 C3B1DB	JP	DBB1	

Véletlenszámok törzsrutinja

A következő véletlen érték előállítására a 1709...170BH tárrekeszekben az alábbi eljárással történik:

E6D8 0610	LD	E,10	
E6DA 3A0917	LD	A,(1709)	
E6DD 2A0A17	LD	HL,(170A)	Kezdőértékek
E6E0 4F	LD	C,A	
E6E1 0F	RRCA		
E6E2 0F	RRCA		
E6E3 0F	RRCA		A ciklus magja ez a 6
E6E4 A9	XOR	C	bitművelet, amely végsősoron
E6E5 17	RLA		egyetlen bitet generál az utána
E6E6 17	RLA		következő összeadásokhoz
E6E7 ED6A	ADC	HL,HL	
E6E9 79	LD	A,C	
E6EA 8F	ADC	A,A	
E6EB 10F3	DJNZ	E6E0	Mindezt 16-szor
E6ED 220A17	LD	(170A),HL	
E6F0 320917	LD	(1709),A	Új értékek
E6F3 C9	RET		

RESTORE belépési pont

E6F4 2A2217	LD	HL,(1722)	TEXT, a program aktuális
E6F7 FE02	CP	02	kezdőcíme
E6F9 2006	JR	NZ,E701	Ha sorszámot nem adtak meg ez
E6FB CDDEFB	CALL	FBDE	lesz az érvényes, egyébként
E6FE CD43FC	CALL	FC43	megkeresi a megadott sor címét
E701 221217	LD	(1712),HL	Lehívja a következő utasítást
E704 210000	LD	HL,0000	Az új DATA sormutató
E707 221417	LD	(1714),HL	Az adatmutatót az elejére
E70A C3B4DB	JP	DBB4	állítja és
			mehet tovább

Ezután a GOSUB utasítás lezáró párja, a RETURN következik. Itt is először a hibakezelő felé való kilépő ág szerepel:

```
E70D CF          RST  8
E70E 09          hibakód: 09          No gosub
```

RETURN belépési pont

```
E70F 3E06        LD  A,06          GOSUB terület azonosítója a
E711 CDE7FC      CALL FCE7        BASIC veremben. Ha illet nem
E714 38F7        JR   C,E70D        talál: hiba!

E716 DDCB00D6    SET  2,(IX+00)    Futó program jelzése
E71A 23          INC  HL
E71B 5E          LD   E,(HL)      A veremből kiolvassa a
E71C 23          INC  HL      visszatérési címet (a GOSUB
E71D 56          LD   D,(HL)    utáni utasítás),
E71E 23          INC  HL
E71F 4E          LD   C,(HL)    majd az ezt tartalmazó sor
E720 23          INC  HL      kezdőcímét
E721 46          LD   B,(HL)    -- újra ez a sor lesz az
E722 ED430C17    LD   (170C),BC    aktuális --,
E726 23          INC  HL      majd az esetleges ELSE ágra
E727 7E          LD   A,(HL)    utaló jelzőbyte-ot
E728 320217      LD   (1702),A
E72B 23          INC  HL
E72C E5          PUSH HL
E72D FDE1        POP  IY      Beállítja a veremmutatót
E72F EB          EX  DE,HL    HL-be tölti a következő
E730 C381DB      JP  DB81    utasítás címét és kész
```

A ROM következő részében a képernyő grafikus kezelésével kapcsolatos BASIC utasításokat feldolgozó rutinok találhatók.

GRAPHICS belépési pont

```
E733 CD1BFB      CALL FB1B        Az üzemmódra megadott értéket
E736 0E00        LD   C,00        állítja elő az A-ban, majd
E738 FE02        CP   02          ennek alapján C-be a megfelelő
E73A 280B        JR   Z,E747    értéket töltve az ismert
E73C 0C          INC  C          funkcióhívással dolgozik
E73D FE04        CP   04
E73F 2806        JR   Z,E747
E741 0C          INC  C
E742 FE10        CP   10
```


E744 C214FB	JP	NZ,FB14	Ha egyik elfogadható értékkel
E747 F704	RST	30: 04	sem egyezik: hiba
E749 D7	RST	10	Hibaellenőrzés után mehet
E74A C3B1DB	JP	DBB1	tovább

A PLOT utasítás rutinja a képzeletbeli toll lerakásával indul - erre, mint tudjuk a PLOT után felsorolható adatok közötti ";" esetén lehet szükség -- majd a minden új pont magrajzolása kezdetén esedékessé váló utasításle hívást is itt lehet elintézni.

A tényleges belépési pont ezek után következik.

E74D F708	RST	30: 08	B-ON, a tollat leteszi
E74F CD43FC	CALL	FC43	Lehívja a következő utasítást.
E752 3039	JR	NC,E78D	Ha vége: kilépés

PLOT belépési pont

E754 302C	JR	NC,E78Z	Ha csak PLOT-ot írtak, ugrás előre
E756 FEBE	CP	BE	PAINT?
E758 2005	JR	NZ,E75F	Ha nem: tovább!
E75A F70A	RST	30: 0A	Kiadja a PAINT funkcióhívást,
E75C D7	RST	10	hibaellenőrzés
E75D 181B	JR	E77A	és mehet tovább
E75F FEA0	CP	A0	","?
E761 28EA	JR	Z,E74D	Ha igen: leteszi a tollat és folytatja
E763 FEA4	CP	A4	","?
E765 281D	JR	Z,E784	Ha igen: felemeli a tollat és folytatja
E767 CDC4FA	CALL	FAC4	HL-be konvertálja az első beírt számot: a pont x koordinátája
E76A 4D	LD	C,L	BC-be
E76B 44	LD	B,H	Ezután csak a "," karaktert fogatja el
E76C 3EA4	LD	A,A4	Majd az y koordinátát számítja ki DE-be és elvégzi a toll pozícionálását
E76E CD54FD	CALL	FD54	
E771 CDC4FA	CALL	FAC4	
E774 EB	EX	DE,HL	
E775 F5	PUSH	AF	
E776 F706	RST	30: 06	
E778 D7	RST	10	Hibaellenőrzés után folytatja
E779 F1	POP	AF	
E77A FEA0	CP	A0	","?
E77C 28CF	JR	Z,E74D	Ha igen: leteszi a tollat és folytatja

E77E FEA4	CP	A4	"," karakter esetén felemeli a
E780 2802	JR	Z,E784	tollat és folytatja
E782 F708	RST	30: 08	Toll le -- toll fel: ez egy pont
E784 F709	RST	30: 09	
E786 D9	EXX		
E787 78	LD	A,B	A legutóbb lehívott
E788 D9	EXX		utasításflag
E789 FEFD	CP	FD	Ha a PLOT-tal kapcsolatban van
E78B 38C2	JR	C,E74F	még tennivaló: akkor folytatja,
E78D C3B4DB	JP	DBB4	egyébként mehet tovább

Ezután a SET utasítás feldolgozása következik. Ez a kulcsszó rendkívül összetett utasításcsoportot vezet be.

A SET után a következő funkciók valamelyike állhat:

- PALETTE: a palettaregiszterek aktuális tartalmának megváltoztatására szolgál;
- INK, PAPER vagy BORDER: ezek aktuális értékét állítja be;
- STYLE, MODE: a vonaltípus és a vonalkeresztezési mód beállítására szolgál;
- CHARACTER: új, meghatározott kóddal megjeleníthető felhasználói karakterek definiálása.

A sokféle lehetőséget megvalósító rutin elvileg meglehetősen egyszerű. A SET után lehívott utasításflag alapján elágazik az egyes funkciók rutinjaira, amelyek bizonyos esetekben csak meghatározott rendszerváltozók értékének beírásából állnak. Máskor bizonyos funkcióhívásokat készítenek elő és hajtanak végre.

A megengedett funkciók szétválasztása hatékony ugrótáblakezelési eljárással történik. A tábla egy-egy eleme kétbyte-os:

- az első a funkció tokenje, amivel az elem és maga a funkció is azonosítható;
- a második byte pedig egy eltolási értéket ad meg valamilyen báziscímhez viszonyítva, amelyből a rendszer a kiválasztott funkció végrehajtási címét tudja elérni.

SET belépési pont

E790 21E4E7	LD HL,E7E4	Az ugrótábla eleje
E793 CD12FD	CALL FD12	Ez a rutin az ugrótábla alapján kikeresi és végre is hajtja a kijelölt funkciót, s ha sikeres volt CY=1-el tér vissza
E796 3842	JR C,E7DA	
E798 CD1BFB	CALL FB1B	A-ban az érték
E79B 1E00	LD E,00	
E79D 321F00	LD (001F),A	A funkciókód betöltése
E7A0 214119	LD HL,1941	Puffercím
E7A3 E5	PUSH HL	Mint tudjuk, a PALETTE és a CHARACTER utasítások esetében a kulcsszavak után meg kell adni azokat a számokat, amelyek a szóbanforgó utasításokat specifikálják
E7A4 067F	LD B,7F	
E7A6 1D	DEC E	
E7A7 F2B4E7	JP P,E7B4	
E7AA D9	EXX	
E7AB 78	LD A,B	
E7AC D9	EXX	
E7AD FEA4	CP A4	
E7AF 201C	JR NZ,E7CD	Ez a PALETTE esetén 4, a CHARACTER utasításnál 11 szám (kód + 10 byte pontmátrix)
E7B1 CD43FC	CALL FC43	
E7B4 FE02	CP 02	
E7B6 300D	JR NC,E7C5	
E7B8 CD94F2	CALL F294	
E7BB 213F19	LD HL,193F	Ezeket az adatokat az utasításból le kell hívni, s megfelelő konverzió után egy pufferverületen kell elhelyezni ahhoz, hogy az ismert funkcióhívásokat használhassuk. Ennek összeállítása történik meg e sorokban
E7BE 367F	LD (HL),7F	
E7C0 CD41FB	CALL FB41	
E7C3 180D	JR E7D2	
E7C5 CD1BFB	CALL FB1B	
E7C8 77	LD (HL),A	
E7C9 23	INC HL	
E7CA 10DA	DJNZ E7A6	
E7CC 04	INC B	
E7CD 3600	LD (HL),00	
E7CF 23	INC HL	
E7D0 10FB	DJNZ E7CD	
E7D2 D1	POP DE	A puffercím DE-be
E7D3 CD1E00	CALL 001E	Lebonyolítja a funkcióhívást, majd hibellenőrzés után, ";" karakter esetén folytatja a következő SET parancs végrehajtását, egyébként pedig áttér a következő utasítás végrehajtására
E7D6 D7	RST 10	
E7D7 D9	EXX	
E7D8 78	LD A,B	
E7D9 D9	EXX	
E7DA FEA0	CP A0	
E7DC C2B4DB	JP NZ,DBB4	
E7DF CD43FC	CALL FC43	
E7E2 18AC	JR E790	

A többi SET utasítás végrehajtása egyenértékű a megfelelő rendszerváltozóba az előírt adat betöltésével.

Most ennek is sajátos technikájával találkozhatunk. A MODE rendszerváltozó (0B4BH) címéhez viszonyított eltolási értéket töltünk minden belépési pontban az E regiszterbe. Ám az egyes belépési pontokat a CPU elől elrejtjük úgy, hogy elé egy 21H kódot írunk. Ettől a CPU a következő sorokat már HL betöltési utasításnak értelmezi, ami számunkra most érdektelen. Így azonban nem rontja el a belépési pontban E-be töltött értéket.

Előbb azonban nézzük az ugrótáblát.

Táblázat a SET utáni elágazáshoz.

Az egyes utasítások tokenje után a táblázatban az az eltolási érték szerepel, amit a következő ROM-címhez kell adni a kívánt utasítás belépési pontjának eléréséhez.

ROM-cím	Token	Eltolás	Előírt cím	Utasítás
E7E4	C3	11	E7F7	MODE
E7E6	B7	12	E7FA	STYLE
E7E8	C4	13	E7FD	INK
E7EA	BC	14	E800	PAPER
E7EC	C7	15	E803	DELAY
E7EE	B9	16	E806	RATE
E7F0	C8	24	E816	CHARACTER
E7F2	BD	2D	E821	PALETTE
E7F4	AE	34	E82A	BORDER

E7F6 00 A táblázat végét jelzi

Ezekután következnek az egyes rendszerváltozók betöltésével járó utasítások belépési pontjai. Az E regiszter betöltése után tehát ugró utasítások nincsenek. Az utána álló betöltési parancsok a CPU elől az LD HL,nn1E utasítás mögé bújtatva, rejtve maradnak.

E7F7	1E00	LD	E,00		0B4B	MODE	
E7F9	211E01	LD	HL,011E				
(E7FA)	1E01			LD	E,01	0B4C	STYLE
E7FC	211E02	LD	HL,021E				
(E7FD)	1E02			LD	E,02	0B4D	INK
E7FF	211E03	LD	HL,031E				
(E800)	1E03			LD	E,03	0B4E	PAPER
E802	211E1A	LD	HL,1A1E				
(E803)	1E1A			LD	E,1A	0B65	DELAY-KEY
E805	211E1C	LD	HL,1C1E				
(E806)	1E1C			LD	E,1C	0B67	RATE-KEY

Ezután már közös a munka, az E regiszterben a megfelelő eltolás van.

E808 1600	LD	D,00	
E80A 214B0B	LD	HL,0B4B	
E80D 19	ADD	HL,DE	
E80E CD1BFB	CALL	FB1B	A betöltendő érték A-ban és a
E811 77	LD	(HL),A	változó betöltése
E812 D9	EXX		
E813 78	LD	A,B	
E814 D9	EXX		
E815 C9	RET		

A 6 rendszerváltozó betöltése után a CHARACTER végrehajtási rutinja következik.

E816 E1	POP	HL	Eldobja a visszatérési címeket
E817 E1	POP	HL	
E818 CD1BFB	CALL	FB1B	Az első konvertált érték a
E81B 4F	LD	C,A	definiálandó karakter kódja
E81C 3E0B	LD	A,0B	A karakterdefiniálás
E81E C39BE7	JP	E79B	funkciókódjával ugrás

A PALETTE rutin

E821 E1	POP	HL	Eldobja a visszatérési címeket
E822 E1	POP	HL	
E823 3E0C	LD	A,0C	Betölti a megfelelő
E825 1E01	LD	E,01	funkciókódot és mehet
E827 C39DE7	JP	E79D	

Végül a BORDER beállító BASIC-rutin

E82A CD1BFB	CALL	FB1B	A beírt értéket konvertálja A-
E82D 87	ADD	A,A	ba, majd szorozza 2-vel,
E82E 324F0B	LD	(0B4F),A	betölti a hozzá tartozó
E831 18DF	JR	E812	rendszerváltozóba és kész

A SET után, amely az egyik legösszetettebb funkciójú BASIC utasítás, a hangkeltés teljesen egyértelmű paranccsal foglalkozunk.

SOUND belépési pont

Először az önálló SOUND parancshoz tartozó paramétereket állítja be:

E833	21150D	LD	HL,0B15	PITCH érték a "C" hanghoz
E836	222A17	LD	(172A),HL	
E839	210732	LD	HL,3207	Időtartam: 32H=50 dec.=1
E83C	222C00	LD	(002C),HL	másodperc, majd 7-es hangerő
E83F	21150B	LD	HL,0B15	beállítása
E842	36FF	LD	(HL),FF	Az új hang megszakítja a régít
E844	37	SCF		
E845	D443FC	CALL	NC,FC43	Lehívja a következő utasítást
E848	FEA0	CP	A0	";"?
E84A	2823	JR	Z,E86F	Ha az új hangnak meg kell
E84C	217BE8	LD	HL,E87B	várnia, hogy az előző
E84F	CD12FD	CALL	FD12	befejeződjön: ugrás előre
E852	FEA4	CP	A4	A többi beállítható
E854	28EF	JR	Z,E845	hangparaméter kezelése
E856	F5	PUSH	AF	(ugrótábla alapján)
E857	ED5B2A17	LD	DE,(172A)	"," karakter esetén a többi
E85B	ED4B2C00	LD	BC,(002C)	paraméter kezelése
E85F	F733	RST	30:33	
E861	AF	XOR	A	
E862	32150B	LD	(0B15),A	PITCH, időtartam és hangerő, funkcióhívás
E865	F1	POP	AF	Az új hang ezt nem szakíthatja
E866	FEFD	CP	FD	meg
E868	300E	JR	NC,E878	
E86A	FEA0	CP	A0	Ha vége az utasításnak: mehet
E86C	C25AFD	JP	NZ,FD5A	tovább, egyébként csak ";"-t
E86F	AF	XOR	A	fogad már el
E870	32150B	LD	(0B15),A	Ha nem ez van: hiba, különben
E873	CD43FC	CALL	FC43	folytatja a további,
E876	38CD	JR	C,E845.	hangképzésre vonatkozó
E878	C3B4DE	JP	DBB4	parancsok végrehajtását

Sajnos, a programozók a 0E83H címre olyan utasítást tettek, amely a HL regiszterpár tartalmát a 002CH címre tölti. Így ez az utasítás elrontja az EXT5 és EXT6 utasításokhoz tartozó táblaelemet, amely a felhasználói USR rutinok kezdőcímeit tartalmazza!

Táblázat a SOUND utáni elágazáshoz:

A hangképzésre megadható előírások végrehajtását most is azzal a módszerrel végezzük, mint ahogy a SET-nél történt. Az ugrótáblában az egyes parancsok tokenjét és a hozzá tartozó eltolást adjuk meg. A táblázat kezelését is ugyanaz az 0FD12H címen kezdődő rutin végzi, amely egyben le is bonyolítja az azonosított funkció végrehajtását.

ROM	Token	Eltolás	Rutincím	Utasítás
E87B	BB	05	E882	PITCH
E87D	C6	0C	E88B	DURATION
E87F	B3	0B	E88C	VOLUME
E881	00			A táblázat végét jelzi

Az előbbi paraméterbeállítások kizárólag egy-egy változó (belső) átírását jelentik, tehát a feladat lényegi része éppen ez a szétválasztás, ill. parancsazonosítás.

PITCH utasítás kezelése

E882	CDC4FA	CALL	FAC4	A beírt értéket a HL-be
E885	Z22A17	LD	(172A),HL	konvertálja
E888	F0	RET	P	
E889	CF	RST	8	Ha túl nagy: hiba!
E88A	04		hibakód: 04	Bad argument

DURATION kezelése

E88B	F637	OR	37	carry=0
------	------	----	----	---------

Ez volt az önálló rész.

VOLUME kezelése

E88C	37	SCF		carry=1 az azonosító jel
E88D	112C00	LD	DE,002C	DE-t a megfelelő címre állítja
E890	3801	JR	C,E893	
E892	13	INC	DE	
E893	CD1BFB	CALL	FB1B	A beírt számot A-ba
E896	12	LD	(DE),A	konvertálja és leteszi a
E897	D9	EXX		megfelelő belső változóba
E898	78	LD	A,B	
E899	D9	EXX		
E89A	C9	RET		

A hangkezelésre vonatkozó parancs után a háttértárolóhoz intézett parancsok feldolgozása folyik.

CLOSE belépési pont

E89B	0E50	LD	C,50	A magnetofon funkciósztályához fordul
E89D	CDEEFB	CALL	FBEE	Lekezeli a periféria hivatkozásokat
E8A0	FEE1	CP	E1	OUTPUT?
E8A2	Z80B	JR	Z,E8AF	Ha igen: ugrás
E8A4	CBF9	SET	7,C	Az input jelzése
E8A6	FEFD	CP	FD	Ha a parancs csak ebből áll:
E8A8	3008	JR	NC,E8B2	ugrás előre
E8AA	FEEC	CP	EC	INPUT?
E8AC	C25AFD	JP	NZ,FD5A	Ha ez sem: hiba!
E8AF	CD43FC	CALL	FC43	Továbblép a következő utasításra
E8B2	79	LD	A,C	
E8B3	E67F	AND	7F	Ellenőrzi a kijelölt perifériát:
E8B5	FE50	CP	50	ez csak a magnetofon, vagy
E8B7	Z804	JR	Z,E8BD	bővítőkártya lehet, különben:
E8B9	FE60	CP	60	hiba!
E8BB	200A	JR	NZ,E8C7	
E8BD	79	LD	A,C	Ha a periféria jó, akkor
E8BE	F604	OR	04	beállítja a CLOSE funkciót és
E8C0	CD1B00	CALL	001B	végrehajtja
E8C3	D7	RST	10	Hibaellenőrzés után mehet tovább
E8C4	C3B1DE	JP	DBB1	
E8C7	CF	RST	8	
E8C8	FF		hibakód: FF	Nem létező hívási kód!

OPEN belépési pont

E8C9	0E50	LD	C,50	A magnetofonhoz fordul
E8CB	CDEEFB	CALL	FBEE	A periféria-hivatkozások kezelése
E8CE	08	EX	AF,AF	
E8CF	79	LD	A,C	Ellenőrzi a kijelölt eszközt:
E8D0	FE50	CP	50	itt csak a magnetofon, vagy
E8D2	Z804	JR	Z,E8D8	bővítőkártya lehet, különben
E8D4	FE60	CP	60	hiba!
E8D6	Z0EF	JR	NZ,E8C7	
E8D8	F603	OR	03	Beállítja az OPEN funkciót
E8DA	4F	LD	C,A	

E8DB 08	EX	AF,AF	A file-név puffer
E8DC 11CE19	LD	DE,19CE	OUTPUT?
E8DF FEE1	CP	E1	
E8E1 280F	JR	Z,E8F2	Ha igen: ugrás előre
E8E3 CBF9	SET	7,C	Beállítja az input jelet
E8E5 FE02	CP	OZ	Ha van megadva file-név:
E8E7 3810	JR	C,E8F9	ugrás előre
E8E9 FEFD	CP	FD	Ha pedig itt az utasítás vége,
E8EB 3018	JR	NC,E905	akkor ugrás ennek
E8ED FEEC	CP	EC	megfelelően. Ha nem is INPUT,
E8EF C25AFD	JP	NZ,FD5A	akkor hiba!
E8F2 CD43FC	CALL	FC43	Az iránykijelölések után
E8F5 FE02	CP	OZ	ismét ellenőrzi a nevet
E8F7 300C	JR	NC,E905	Ha nincs név: ugrás
E8F9 C5	PUSH	BC	
E8FA CD94F2	CALL	F294	A nevet a verembe teszi, és
E8FD FDE5	PUSH	IY	azonnal vissza is állítja a
E8FF CD21FA	CALL	FA21	mutatót (törli)
E902 D1	POP	DE	
E903 13	INC	DE	DE a névre mutat
E904 C1	POP	BC	
E905 79	LD	A,C	A= funkciókód
E906 326B0B	LD	(0B6B),A	Mivel A=00 nem lehet, így
E909 CD1B00	CALL	001B	pufferelt file lesz
E90C D7	RST	10	Funkcióhívás hibellenőrzés-
E90D C3B1DB	JP	DBB1	sel és
			kész

GET utasítás végrehajtása

E910 0E10	LD	C,10	Elsődlegesen a
E912 CDEEFB	CALL	FBEE	billentyűzethez fordult, de
E915 08	EX	AF,AF	meg kell nézni a periféria
E916 79	LD	A,C	hivatkozásokat is
E917 F681	OR	81	Funkciókód egy karakter
E919 CD1B00	CALL	001B	beolvasására és a hívás
E91C FDE5	PUSH	IY	lebonyolítása
E91E E1	POP	HL	
E91F FEEC	CP	EC	File vége?
E921 2806	JR	Z,E929	Ha igen: ugrás előre
E923 B7	OR	A	Hibellenőrzés
E924 D7	RST	10	
E925 2B	DEC	HL	Leteszi a karaktert
E926 71	LD	(HL),C	A=01-et állít be: hossz
E927 3C	INC	A	
E928 0EAF	LD	C,AF	

Ez így érdektelen utasítás, ha azonban file-vég hiba volt, akkor a CPU a 0E929H-ra ugrik, a 0AFH kódra.

```

E929 AF      XOR  A
E92A 2B      DEC  HL
E92B 77      LD   (HL),A
E92C 2B      DEC  HL
E92D 3601    LD   (HL),01
E92F 08      EX  AF,AF
E930 FEFD    CP   FD
E932 D2B4DB  JP   NC,DBB4
E935 E5      PUSH HL
E936 FDE1    POP  IY
E938 FE01    CP   01
E93A 2003    JR   NZ,E93F
E93C D9      EXX
E93D CB51    BIT  2,C
E93F C25AFD  JP   NZ,FD5A
E942 CB59    BIT  3,C
E944 C40BF4  CALL NZ,F40B
E947 D9      EXX
E948 CD2EF4  CALL F42E
E94B CD41FB  CALL FB41
E94E C3B1DB  JP   DBB1

```

Ez lesz a szöveg hossz
 Leteszi a hosszt (0, 1)
 Elé a szövegazonosítót
 Ha magányos GET volt,
 visszatér,
 ha a GET után változónév is
 szerepel, akkor a beolvasott
 karaktert oda kell betölteni
 Ha a változó nem szövegtípusú,
 vagy DEF-t azonosít: hiba!
 Ha beépített függvény:
 újradefiniálható
 Rááll a szimbólum adatbyte-
 jára, bemásolja a karaktert
 és kész

LOAD belépési pont

```

E951 CD35EA  CALL EA35
E954 CD10DE  CALL DE10
E957 E5      PUSH HL
E958 ED5BE119 LD  DE,(19E1)
E95C D5      PUSH DE
E95D CD99FC  CALL FC99
E960 C1      POP  BC
E961 D1      POP  DE
E962 D5      PUSH DE
E963 3A0517  LD   A,(1705)
E966 F682    OR   82
E968 CD1B00  CALL 001B
E96B F5      PUSH AF
E96C C410DE  CALL NZ,DE10
E96F F1      POP  AF
E970 D7      RST 10
E971 CDFCFC  CALL DCFC
E974 CD5AEA  CALL EA5A
E977 E1      POP  HL
E978 3AE319  LD   A,(19E3)
E97B B7      OR   A
E97C CADADA  JP   Z,DADA
E97F C323DE  JP   DE23

```

Megnyitja a file-t és
 beolvassa a fejléct
 BASIC inicializálás
 A báziscím
 A program mérete
 Területellenőrzés
 BC=hossz
 DE=betöltési cím
 A kijelölt funkciósztály és
 funkciókód a
 blokkbeolvasáshoz, ezzel a
 funkcióhívás lebonyolítása
 Ha hiba volt: ismét
 inicializálás
 Inicializálja a
 szimbólumtáblát és lezárja a
 file-t, majd veszi a program
 kezdőcímét
 AUTORUN?
 Ha nem, akkor ugrás a
 parancsbekérési ponthoz,
 ha igen: RUN

SAVE utasítás végrehajtása

E982 AF	XOR A	OUTPUT jelzése
E983 CDFBE9	CALL E9FB	Megnyitja a file-t
E986 11EF19	LD DE,19EF	
E989 0610	LD B,10	
E98B AF	XOR A	Először törli a kiküldendő
E98C 1B	DEC DE	fejlécet
E98D 12	LD (DE),A	
E98E 10FC	DJNZ E98C	
E990 D5	PUSH DE	
E991 3C	INC A	
E992 32E019	LD (19E0),A	A=01: programfile jele
E995 3A0717	LD A,(1707)	Beállítja az AUTORUN jelzést
E998 32E319	LD (19E3),A	-- amint a felhasználó kérte
E99B CD41DD	CALL DD41	
E99E 23	INC HL	Megkeresi a program végét és
E99F ED5B2217	LD DE,(1722)	a kezdőcímből kiszámítja a
E9A3 AF	XOR A	tárolandó blokk hosszát
E9A4 320717	LD (1707),A	Az AUTORUN törlése
E9A7 ED52	SBC HL,DE	
E9A9 22E119	LD (19E1),HL	A hossz a fejlécbe
E9AC E3	EX (SP),HL	
E9AD D5	PUSH DE	
E9AE 0610	LD B,10	
E9B0 3A0517	LD A,(1705)	
E9B3 F601	OR 01	Ezután beállítja az egy
E9B5 321F00	LD (001F),A	karakter kiküldéséhez tartozó
E9B8 4E	LD C,(HL)	funkciókódot, majd
E9B9 23	INC HL	karakterenként kiküldi a
E9BA C5	PUSH BC	fejlécet a kijelölt eszközre
E9BB CD1E00	CALL 001E	
E9BE D7	RST 10	
E9BF C1	POP BC	
E9C0 10F6	DJNZ E9B8	
E9C2 D1	POP DE	
E9C3 C1	POP BC	
E9C4 3A0517	LD A,(1705)	A fejléc után a blokkoutput
E9C7 F602	OR 02	funkcióhívással tárolja az
E9C9 CD1E00	CALL 001E	egész programot
E9CC D7	RST 10	
E9CD CD5AEA	CALL EA5A	Lezárja a file-t és kéri a
E9D0 C3B1DE	JP DEB1	következő utasítást

VERIFY belépési pont

E9D3 CD35EA	CALL EA35	Megnyitja a file-t
E9D6 CD41DD	CALL DD41	Megkeresve a program végét és kiolvasva az elejének címét, megállapítja a program hosszát
E9D9 23	INC HL	
E9DA ED5B2217	LD DE,(1722)	
E9DE B7	OR A	
E9DF ED52	SBC HL,DE	Ha ez nem egyezik a beolvasott fejlécben megadott hosszal, akkor már is hiba!
E9E1 ED4BE119	LD BC,(19E1)	
E9E5 ED42	SBC HL,BC	
E9E7 C258EA	JP NZ,EA58	
E9EA 3A0517	LD A,(1705)	Beállítja a VERIFY funkciót és végrehajtja
E9ED F685	OR 85	
E9EF CD1B00	CALL 001B	
E9F2 D7	RST 10	Hibaellenőrzés, majd lezárja a file-t és mehet tovább
E9F3 CD5AEA	CALL EA5A	
E9F6 C3B1DB	JP DBB1	

A magnetofonra -- vagy más perifériákra, pl. a diszkre -- vonatkozó alapvető parancsok után még három, ezek munkáját segítő szubrutinnal kell megismerkednünk. E rutinok a file-ok megnyitásával, zárásával és a fejléc kezelésével kapcsolatosak.

File megnyitása

E9F9 3E80	LD A,80	Input jelzése
E9FB DDCB00DE	SET 3,(IX+00)	Nyitott file jelzése
E9FF F5	PUSH AF	
EA00 0E50	LD C,50	Alapeszköz a magnetofon, de kielégíti a periféria-hivatkozásokat is
EA02 CDEEFB	CALL FBEE	
EA05 08	EX AF,AF	
EA06 79	LD A,C	
EA07 FE50	CP 50	Ha a kijelölt eszköz nem magnetofon vagy bővítőkért, akkor hiba!
EA09 2805	JR Z,EA10	
EA0B FE60	CP 60	
EA0D C2C7E8	JP NZ,E8C7	
EA10 C1	POP BC	
EA11 B0	OR B	Az I/O irány beállítása
EA12 320517	LD (1705),A	
EA15 F603	OR 03	Az OPEN funkció kijelölése
EA17 321F00	LD (001F),A	
EA1A 08	EX AF,AF	
EA1B 11CE19	LD DE,19CE	
EA1E FE02	CP 02	Megnézzük, hogy van-e megadva név, s ha igen, a nevet a BASIC verembe másoljuk
EA20 300A	JR NC,EA2C	
EA22 CD94F2	CALL F294	

EA25 FDE5	PUSH IY	A mutatót visszaállítva a
EA27 CD21FA	CALL FA21	bejegyzést érvénytelenítjük,
EA2A D1	POP DE	de a DE a név címére áll
EA2B 13	INC DE	
EA2C AF	XOR A	
EA2D 326B0B	LD (0B6B),A	Nem puffereelt file
EA30 CD1E00	CALL 001E	Normál funkcióhívással
EA33 D7	RST 10	megnyitjuk a file-t
EA34 C9	RET	

File megnyitása és a fejléc kezelése

EA35 CDF9E9	CALL E9F9	Megnyitjuk a file-t
EA38 3A0517	LD A,(1705)	Karakterbeolvasási
EA3B F681	OR 81	funkciókódot állítunk be
EA3D 321F00	LD (001F),A	
EA40 21DF19	LD HL,19DF	
EA43 E5	PUSH HL	
EA44 0610	LD B,10	
EA46 C5	PUSH BC	
EA47 CD1E00	CALL 001E	A fejléc-pufferbe karakteren-
EA4A D7	RST 10	ként, hibellenőrzéssel beol-
EA4B 71	LD (HL),C	vassuk a fejlécet
EA4C 23	INC HL	
EA4D C1	POP BC	
EA4E 10F6	DJNZ EA46	
EA50 E1	POP HL	Ellenőrizzük a fejlécet:
EA51 B6	OR (HL)	ha az 1. byte nem 00,
EA52 2004	JR NZ,EA58	ugrás előre, hiba!
EA54 23	INC HL	
EA55 3C	INC A	A 2. byte-nak 1-nek kell
EA56 96	SUB (HL)	lennie
EA57 C8	RET Z	Ha az: rendben van,
EA58 CF	RST 8	egyébként hiba:
EA59 10	hibakód: 10	Bad file

File lezárása

EA5A 3A0517	LD A,(1705)	Beállítjuk a kijelölt
EA5D F604	OR 04	osztálynak a CLOSE funkciót és
EA5F CD1B00	CALL 001B	végrehajtjuk
EA62 D7	RST 10	Hibaellenőrzés után beállítjuk
EA63 DDCB009E	RES 3,(IX+00)	a lezárt file jelzést és kész
EA67 C9	RET	

3.6.5.5 Összefoglaló tanulságok

Ezzel, kedves Olvasó, ismét egy jelentős rész végére értünk!

A TV-COMPUTER BASIC parancselemzője az azonosított tokenek alapján az előző oldalakon megismert rutinokkal hajtja végre a felhasználó BASIC nyelven beírt utasításait. Utólag visszagondolva a több-kevesebb részletességgel átfutott rutinokra, az egész mechanizmus összességében nem tűnhet bonyolultnak.

Azokkal a szubrutinokkal, amelyekre az előző oldalakon oly sok hivatkozás történt, többségükben még részletesebben is foglalkozunk a megfelelő résznél. A működés nagyszerű logikája azonban enélkül is teljesen egyértelmű lehet.

A mikroprocesszor byte-onként halad előre a memóriában tárolt parancsok sorozatán. A talált tokenek alapján tisztázza az éppen végrehajtandó feladatot, s a megfelelő végrehajtási címre ugrik. Ott újabb és újabb parancsbyte-ok lehívásával pontosítja a feladatot, mégpedig a kulcsszó "nyelvtanilag" rögzített szabályait alapul véve, s amennyiben az előírt formáktól való eltéréseket talál, akkor hatékony és differenciált hibakezeléssel e hibákról üzeneteket küld a felhasználónak. Ha viszont minden rendben van, akkor a feladat elvégzése után visszatér a parancselemző részhez, ahol új feladatot kap.

3.6.6 Beépített függvények

3.6.6.1 Bevezetés

Tulajdonképpen ezek is a BASIC "aprópénzéhez" tartoznak. Azért kerültek mégis új fejezetbe, mert itt elvileg másról van szó, mint az előző részben.

Az itt közölt rutinok már nem önálló kulcsszavakhoz tartoznak, csak valamely elsődleges token után állva értelmezhetők. Erre is láttunk már példákat, itt azonban a tokenekhez kapcsolódó hivatkozások nem valamely kulcsszó

értelmét pontosítják, hanem speciális kifejezések, ismert vagy kevésbé ismert matematikai függvények. Ezek értékét a program működése közben kell kiszámolni, hogy azután az eredményt maga a program már konkrét adatként használhassa.

Ezek a kifejezések sokszor nem számolási műveletet takarnak. Pl. a LEN függvény egy megadott karakterlánc hosszát jelenti, vagy az IN egy portról olvas be adatot.

Nagyon fontos tudnunk, hogy a TV-Computer belső függvényeit a szimbólumok láncolt struktúrájába építették be. Ez tehát azt jelenti, hogy a számítógép ugyanazzal a technikával éri el mondjuk az abszolútérték-függvényt, mint amellyel a felhasználó által létrehozott változókat. Erre a szimbólumok kezelésének speciális módszere, a láncolási mechanizmus és az azonosítók használata ad lehetőséget.

A szimbólumok kezeléséről a 3.6.4 pontban már tettünk említést. Itt ismét kérem az Olvasót, hogy ha ezzel kapcsolatos ismereteiben bizonytalan, lapozza fel a TVC kézikönyveit, ahol megtalálja a szükséges alapvető tudnivalókat.

Mint minden szimbólum, a beépített függvények első két byte-ja is az előző szimbólumelem címét tartalmazza. Utána a szimbólum azonosítója: a névhossz és a név karakterei állnak, majd a típusbyte, s az ezt követő címen -- beépített függvények esetén -- már a végrehajtást jelentő CPU utasítássorozat kezdődik, ezek a szimbólum adatbyte-jei.

Az első ilyen beépített függvény a 0F094H címen kezdődik. Az utolsó címe pedig 0EAE2H, ennek első két byte-ja 00-kat tartalmaz. Ez jelenti azt, hogy nincs több szimbólum. A szimbólumok keresése a CHAIN változóban megadott címen kezdődik, a felhasználó által létrehozott szimbólumokkal. Így a beépített függvények átdefiniálhatók.

Amikor a BASIC parancselemző egy szimbólummal találkozik, akkor veszi a CHAIN változóból az utoljára definiált szimbólum címét, s a korábbi szimbólumokra mutató címekkel megvalósított láncolás révén végignézheti az összes -- már létező -- szimbólumokat. A felhasználói szimbólumok közül az első bevezeti a gépet ROM-ba beépített szimbólumokba és ott folytatódik a keresés.

Hogy a beépített függvények eredményeivel hogyan kell tovább dolgozni, azt az elsődleges tokenek, vagy az utasításban utánuk álló többi parancs határozza meg. A függvények kiértékelése minden esetben csak egy közbülső szubrutin az adott feladat végrehajtásában.

Sokszor használjuk ebben a részben az RST 18 utasításra épített "közbülső nyelvet".

Végül még egy jellegzetesség: itt különösen sok olyan részt talál majd az Olvasó, amelynek részletes elemzését -- ha szüksége lesz rá -- magának kell elvégeznie. A már többször is kifejtett okokból az ilyen értelemben vett alaposságról a könyvnek ebben a részében lemondunk -- bár lesznek azért ebben az esetben is kivételek.

3.6.6.2 Hasznos segédrutinok

Zárójeles számkifejezések kiértékelése

EA68 3E96	LD	A,96	Először tehát csak a "(" esetén megy tovább, azután kiértékeli a kifejezést, utána pedig csak a ")" jelet fogadja el, különben hibát jelez (Not understood)
EA6A CD54FD	CALL	FD54	
EA6D CDA7F0	CALL	F0A7	
EA70 3E95	LD	A,95	
EA72 C354FD	JP	FD54	

Zárójeles szövegkifejezések feldolgozása

EA75 3E96	LD	A,96	Teljesen azonos az előbbivel de itt a karakterláncok feldolgozó rutinját hívja meg
EA77 CD54FD	CALL	FD54	
EA7A CD94F2	CALL	F294	
EA7D 3E95	LD	A,95	
EA7F C354FD	JP	FD54	

Di Az i-edik konstans töltése a BASIC verembe

Ezt az RST 18-ra épített műveleteknél Di-vel jelöljük. Ha az RST 18 után írt funkciószám 05, akkor a következő byte-on meg kell adnunk annak a konstansnak a sorszámát, amely a ROM 0C111H című rekesztől 7-7 byte-on, lebegőpontos, tömörített BCD kódban van lerakva. Az itt következő, rutin az így kiválasztott számot tölti a verembe.

Belépéskor az RST 18 utáni aktuális memóriarekesz címe a 1718H címen van tárolva.

EA82 2A1817	LD	HL,(1718)	A pointer által mutatott címről kiolvassa a konstans sorszámát, s megnöveli a pointert
EA85 7E	LD	A,(HL)	
EA86 23	INC	HL	
EA87 221817	LD	(1718),HL	

EA8A 6F	LD	L,A	
EA8B AF	XOR	A	
EA8C 67	LD	H,A	HL=a konstans sorszáma
EA8D E5	PUSH	HL	
EASE 29	ADD	HL,HL	
EAEF 29	ADD	HL,HL	
EA90 29	ADD	HL,HL	
EA91 D1	POP	DE	
EA92 ED52	SBC	HL,DE	HL=a sorszám * 7
EA94 1111C1	LD	DE,C111	A konstansok kezdőcíméhez ezt hozzáadva, eljutunk a keresett számhoz
EA97 19	ADD	HL,DE	
EA98 1808	JR	EAAZ	

Y Az Y lebegőpontos regiszter betöltése a verembe

A feladat ugyanaz, mint az előbb, csak most a betöltendő adat helye -- az Y regiszter címe -- meghatározott.

EA9A 21C719	LD	HL,19C7	A lényeg itt a cím betöltése HL-be, a többit majd később intézzük
EA9D 1803	JR	EAAZ	

X Az X lebegőpontos regiszter betöltése a verembe

EA9F 21C019	LD	HL,19C0	Az X regiszter címe. Ezután a három rutin közös területellenőrzés
EAA2 CD8EFC	CALL	FC8E	

Ezután következik a tényleges bemásolás. Emlékezzünk rá, hogy a veremben nem 7, hanem 8 byte-on tároljuk a számokat -- erre a másolásnál oda kell figyelni.

EAA5 110600	LD	DE,0006	
EAA8 19	ADD	HL,DE	A szám exponense
EAA9 FDE5	PUSH	IY	
EAAB D1	POP	DE	
EAAC 1B	DEC	DE	
EAAD EDAB	LDD		Betölti az exponenst
EAAF AF	XOR	A	
EAB0 12	LD	(DE),A	Ez a 00 érték kell a veremben a túlcsoordulások kezeléséhez
EAB1 1B	DEC	DE	
EAB2 010600	LD	BC,0006	
EAB5 EDB8	LDDR		Utána a többi 6 byte, a végére a 09 azonosító
EAB7 3E09	LD	A,09	
EAB9 12	LD	(DE),A	
EABA D5	PUSH	DE	Beállítjuk az új veremmutatót és
EABB FDE1	POP	IY	
EABD C9	RET		kész

SYD A BASIC veremből a számot az Y regiszterbe tölti, a szám a veremből elveszik

EABE 11C719	LD	DE,19C7	Y címe
EAC1 1803	JR	EAC6	A végrehajtás most is közös lesz

SXD A BASIC veremből a számot az X regiszterbe tölti, a szám a veremből elveszik

EAC3 11C019	LD	DE,19C0	X regiszter címe
EAC6 CDD5EA	CALL	EAD5	Elvégzi a bemozgatást, majd a
EAC9 E5	PUSH	HL	veremmutató átállításával
EACA FDE1	POP	IY	érvényteleníti az adatot
EACC C9	RET		

SY A számot a BASIC veremből az Y regiszterbe másolja

EACD 11C719	LD	DE,19C7	Y címe
EAD0 1803	JR	EAD5	A végrehajtás közös lesz

SX A számot a veremből az X-be másolja

EAD2 11C019	LD	DE,19C0	
EAD5 FDE5	PUSH	IY	
EAD7 E1	POP	HL	
EAD8 23	INC	HL	Először átmásolja a 6
EAD9 010600	LD	BC,0006	számjegyet, majd
EADC EDB0	LDIR		
EADE 23	INC	HL	átlépi az exponens előtti plusz
EADF EDA0	LDI		byte-ot, s az exponenst is
EAE1 C9	RET		leteszi

3.6.6.3 Belső szimbólumok

ABS A veremben a meghatározott szám abszolút értékét hagyja

EAE2 00 00	Az első szimbólumbejegyzés, ezért a megelőző szimbólumra mutató cím helyén: 0000
------------	--

EAE4 03	Az azonosító hossza: 03
EAE5 41 42 53	neve: ABS
EAE8 0A	típusa: beépített numerikus

EAE9 CD68EA	CALL EA68	Értékeli a kifejezést
EAEC FDCB08BE	RES 7, (1Y+08)	Az előjelbitet törli
EAFO C9	RET	

A következő rutin kilóg a sorból. Nem beépített szimbólumként van beláncolva, ezért a kifejezéseket kezelő ROM rész is külön foglalkozik vele.

ATN A megadott kifejezés arkusztangensét veszi a $-\pi/2$ és $+\pi/2$ intervallumban. Azaz olyan számot ad vissza ezen határok között, amelynek tangense az eredeti kifejezés értéke

EAF1 CD43FC	CALL FC43	Lehívja a következő utasítást
EAF4 CD68EA	CALL EA68	és kiértékeli az ATN után álló
EAF7 010000	LD BC,0000	kifejezést
EAFA FD7E08	LD A, (1Y+08)	Az exponens
EAFD E680	AND 80	
EAFF 4F	LD C,A	A szám előjele
EB00 C5	PUSH BC	
EB01 C426F7	CALL NZ,F726	Ha negatív: * (-1)
EB04 DF	RST 18	
EB05 08	: SX	Tárolás X-ben
EB06 8501	: D1	1.00 betöltése a verembe
EB08 CD93F6	CALL F693	Összehasonlítás
EB0B FA1AEB	JP M,EB1A	Ha az érték kisebb, vagy
EB0E 280A	JR Z,EB1A	egyenlő: 1-gyel: ugrás
EB10 C1	POP BC	
EB11 0602	LD B,02	Ha 1-nél nagyobb volt,
EB13 C5	PUSH BC	feljegyzí
EB14 DF	RST 18	
EB15 0501	: D1	1.0
EB17 06	: X	X
EB18 01	: /	1/X
EB19 8A	: SxD	X-ben 0...1 közötti szám

A számolás második szakasza következik.

EB1A DF	RST 18	
EB1B 06	: X	X
EB1C 8502	: D2	D2=0.267949192431
EB1E CD93F6	CALL F693	Összehasonlítás
EB21 FA3CEB	JP M,EB3C	
EB24 2816	JR Z,EB3C	Ha $X \leq D2$: ugrás előre
EB26 C1	POP BC	

EB27 04	INC	B	adminisztrál
EB28 05	PUSH	BC	
EB29 0F	RST	18	
EB2A 0504	:	D4	D4=0.732050807569
EB2C 06	:	X	X a stackbe
EB2D 02	:	*	D4*X
EB2E 0500	:	D0	D0=0.5
EB30 03	:	-	D4*X - 0.5
EB31 0500	:	D0	
EB33 03	:	-	D4*X - 1
EB34 06	:	X	
EB35 00	:	+	D4*X-1+X=(D4+1)*X - 1
EB36 0503	:	D3	
EB38 06	:	X	
EB39 00	:	+	D3+X
EB3A 01	:	/	
EB3B 8A	:	SXD	$X = ((D4+1)*X-1) / (D3+X)$

Ez volt a számolás második szakasza, a verem ismét üres. A harmadik szakasz:

EB3C 0F	RST	18	
EB3D 06	:	X	Legyen ez d
EB3E 0C	:	DUP	d;d
EB3F 0C	:	DUP	d;d;d
EB40 0C	:	DUP	d;d;d;d
EB41 02	:	*	d;d; d*d
EB42 08	:	SX	X=d*d
EB43 0506	:	D6	
EB45 02	:	*	d;d; D6*d*d
EB46 0505	:	D5	
EB48 00	:	+	d;d; D6*d*d+D5
EB49 06	:	X	d;d; D6*d*d+D5; d*d
EB4A 02	:	*	d;d; (D6*d*d+D5)*d*d
EB4B 06	:	X	d;d; (D6*d*d+D5)*d*d; d*d
EB4C 0508	:	D8	d;d; (D6*d*d+D5)*d*d; d*d; D8
EB4E 00	:	+	d;d; (D6*d*d+D5)*d*d; D8+d*d
EB4F 06	:	X	d;d; (D6*d*d+D5)*d*d; D8+d*d; d*d
EB50 02	:	*	d;d; (D6*d*d+D5)*d*d; (D8+d*d)*d*d
EB51 0507	:	D7	d;d; (D6*d*d+D5)*d*d; (D8+d*d)*d*d; D7
EB53 00	:	+	d;d; (D6*d*d+D5)*d*d; (D8+d*d)*d*d+D7
EB54 01	:	/	d;d; (D6*d*d+D5)*d*d / /((D8+d*d)*d*d+D7)
EB55 02	:	*	
EB56 80	:	+	

Így végül a stackben egyetlen szám van:

$$d+d*d*d(D6*d*d+D5)/((D8+d*d)*d*d+D7)$$

A befejezés:

EB57 C1	POP	BC	
EB58 78	LD	A,B	A korábbi adminisztráció
EB59 FE02	CP	O2	
EB5B D426F7	CALL	NC,F726	* (-1)
EB5E C5	PUSH	BC	
EB5F 2114C2	LD	HL,C214	A PI/6 konstans címe
EB62 05	DEC	B	
EB63 280C	JR	Z,EB71	
EB65 2106C2	LD	HL,C206	A PI/2 címe
EB68 05	DEC	B	
EB69 2806	JR	Z,EB71	
EB6B 210BC2	LD	HL,C20B	A PI/3 címe
EB6E 05	DEC	B	
EB6F 2006	JR	NZ,EB77	
EB71 CDA2EA	CALL	EAA2	A feltételek szerint kiválasztott konstanst a verembe tölti és
EB74 CD93F4	CALL	F493	hozzáadja a kiszámolt értékhez
EB77 C1	POP	BC	
EB78 79	LD	A,C	Az előjel
EB79 FDAE08	XOR	(IY+08)	Beállítja az eredmény előjelét
EB7C FD7708	LD	(IY+08),A	és kész
EB7F C9	RET		

A kedves Olvasó ne várja azt, hogy az előbbieknél most részletes magyarázatát adjam. A speciális matematikai függvények értékeinek közelítő, ám nagy pontosságú és viszonylag mégis elviselhetően gyors kiszámítására a számítástechnikában különböző módszerek használatosak. Ezek matematikai megalapozása meglehetősen messzire vezetne, de valójában nem is számítástechnikai probléma. Így az itt leírtak kizárólagosan a közlés teljességét szolgálják.

Most viszont egymás után több beépített függvény következik -- ezek már a szimbólumtáblába láncolt elemek.

CHR\$ Az ismert BASIC függvény

EB80 E2EA Az előző szimbólum címe: EAE2
EB82 04 Az azonosító hossza: 04
EB83 43 48 52 24 neve: CHR\$
EB87 08 típusa: beépített szöveges

EB88 CD68EA CALL EA68 Kiértékeli a megadott kifejezést
EB8B CD1AFB CALL FB1A A-ban a kiszámolt kód
EB8E FDE5 PUSH IY
EB90 E1 POP HL
EB91 2B DEC HL
EB92 77 LD (HL),A Leteszi az ASCII kódot,
EB93 2B DEC HL
EB94 3601 LD (HL),01 elé a hosszt, majd
EB96 2B DEC HL
EB97 3601 LD (HL),01 az azonosítót

EB99 E5 PUSH HL Beállítva a veremmutatót kész
EB9A FDE1 POP IY is
EB9C C9 RET

COS A megadott kifejezés koszinuszát számolja ki. Felhasználja, hogy matematikailag a szinuszos függvényből $\pi/2$ eltolással származtatható

EB9D 80EB Az előző szimbólum címe: EB80
EB9F 03 Az azonosító hossza: 03
EBA0 43 4F 53 neve: COS
EBA3 0A típusa: beépített numerikus

EBA4 CD68EA CALL EA68 Kiértékeli a megadott kifejezést
EBA7 DF RST 18
EBA8 0523 : D3 PI/2 a stackbe
EBA9 80 : + Hozzáadja megadott számhoz és
EBAB C35EEE JP EE5E ugrás a SIN-hoz

A szögfüggvények számításakor -- ebből is látszik -- a szögeket nem fokokban, hanem radiánban kell megadni. Ha mégis fokokban akarunk dolgozni, akkor a szükséges átszámítást a kifejezésekbe be kell írni. Pl. 48 fok koszinusza:

$\text{COS } (48/180*\pi)$.

EXP A természetes logaritmus alapszámának, az e számnak megadott kitevőjű hatványát számítja ki

EBAE 9DEB Az előző szimbólum címe: EB9D
 EBB0 03 Az azonosító hossza: 03
 EBB1 45 58 50 neve: EXP
 EBB4 0A típusa: beépített numerikus

EBB5 CD68EA CALL EA68 Kiértékeli a kifejezést
 EBB8 DF RST 18 Legyen a kitevő: k
 EBB9 0C : DUP k;k
 EBBA 0509 : D9 k;k;D9
 EBBC 82 : * k; D9*k
 EBBD CDAFEC CALL ECAF Veszi az egészrészét: n
 EBC0 FD7E08 LD A, (IY+08) Az exponens
 EBC3 4F LD C,A
 EBC4 E67F AND 7F
 EBC6 FE43 CP 43 Ha ez 1000-nél nagyobb és pozitív kitevő, akkor túl nagy szám lenne:
 EBC8 380A JR C,EBD4 Overflow hiba
 EBCA B1 OR C
 EBCB F212F9 JP P,F912
 EBCE CD1BFA CALL FA1B
 EBD1 C3F9F9 JP F9F9 Ha a kitevő ilyen nagy negatív szám: az eredmény 0 lesz

EBD4 CD92FA CALL FA92 DUP, így a verem: k;n;n
 EBD7 CDC3FA CALL FAC3 HL-be konvertálja n-t
 EBDA 5D LD E,L
 EDBB 54 LD D,H DE=HL
 EBDC 017E00 LD BC,007E További ellenőrzéseket végez, s ha az eredmény túl nagy lenne visszatér (hibajelzéssel),
 EBE0 B7 OR A
 EBE2 FACEEB JP M,EBCE
 EBE5 CB21 SLA C túl kis szám esetén pedig 0
 EBE7 ED42 SBC HL,BC eredményt ad
 EBE9 D212F9 JP NC,F912
 EBEC D5 PUSH DE Elteszi n-t

EBED DF RST 18
 EBEE 09 : SY n-t Y-ba másolja;
 EBEF 050A : D10 k;n;D10
 EBF1 02 : * k; n*D10
 EBF2 03 : - k-n*D10
 EBF3 07 : Y k-n*D10; n
 EBF4 050E : D11 k-n*D10; n; D11
 EBF6 02 : * k-n*D10; n*D11
 EBF7 03 : - k-n*D10-n*D11 = m
 EBF8 0C : DUP m;m
 EBF9 0C : DUP m;m;m
 EBFA 02 : * m; m*m
 EBFB 08 : SX m*m-t X-be másolja

EBFC 0510	:	D16	m; m*m; D16
EBFE 00	:	+	m; m*m+D16
EBFF 06	:	X	m; m*m+D16; m*m
EC00 02	:	*	m; (m*m+D16)*m*m
EC01 050F	:	D15	m; (m*m+D16)*m*m; D15
EC03 00	:	+	m; (m*m+D16)*m*m+D15
EC04 0B	:	SYD	Az utolsó érték Y-ba kerül, a verem: m
EC05 050E	:	D14	m; D14
EC07 06	:	X	m; D14; m*m
EC08 02	:	*	m; D14*m*m
EC09 050D	:	D13	m; D14*m*m; D13
EC0B 00	:	+	m; D14*m*m+D13
EC0C 06	:	X	m; D14*m*m+D13; m*m
EC0D 02	:	*	m; (D14*m*m+D13)*m*m
EC0E 050C	:	D12	m; (D14*m*m+D13)*m*m; D12
EC10 00	:	+	m; (D14*m*m+D13)*m*m+D12
EC11 02	:	*	((D14*m*m+D13)*m*m+D12)* *m
EC12 0B	:	SX	A kapott értéket X-be másolja

Jelöljük az X-beli értéket p-vel, az Y értékét q-val.

EC13 07	:	Y	P; q
EC14 06	:	X	P; q; P
EC15 03	:	-	P; q-P
EC16 01	:	/	p/(q-p)
EC17 0500	:	DO	p/(q-p); DO
EC19 00	:	+	DO+p/(q-p)
EC1A 0C	:	DUP	
EC1B 00	:	+	2*(DO+p/(q-p)) = r
EC1C 8A	:	SXD	r-t X-be tölti, a verem üres

EC1D 01	:	POP	DE
EC1E CB43	:	BIT	0, E
EC20 2815	:	JR	Z, EC37
EC22 D5	:	PUSH	DE

Ha n páros szám: ugrás előre

EC23 DF	:	RST	18
EC24 06	:	X	r
EC25 0520	:	D32	r; D32
EC27 02	:	*	r*D32 = v
EC28 8A	:	SXD	v-t tölti X-be

Ezzel a számítások lényegileg befejeződtek, mindössze az maradt még hátra, hogy a DE-ben tárolt n értéke szerint az X regiszterbeli eredmény exponensét kiigazítsuk.

EC29 D1	:	POP	DE
---------	---	-----	----

EC2A 14	INC	D	
EC2B 15	DEC	D	
EC2C 13	INC	DE	
EC2D 2808	JR	Z, EC37	
EC2F FA37EC	JP	M, EC37	
EC32 21C619	LD	HL, 19C6	Az X regiszter exponense
EC35 34	INC	(HL)	
EC36 1B	DEC	DE	
EC37 21C619	LD	HL, 19C6	
EC3A AF	XOR	A	
EC3B B3	OR	E	
EC3C 1F	RRA		
EC3D 86	ADD	A, (HL)	
EC3E E67F	AND	7F	Az exponens kiigazítása után
EC40 77	LD	(HL), A	még az X értékét bemásolja a
EC41 C39FEA	JP	EA9F	BASIC verembe

FREE Kiírja a BASIC számára még rendelkezésre álló területet

EC44 AEEB	Az előző szimbólum címe:	EBAE
EC46 04	Az azonosító hossza:	04
EC47 46 52 45 45	neve:	FREE
EC4B 0A	típusa:	beépített numerikus

EC4C FDE5	PUSH	IY	
EC4E E1	POP	HL	A pillanatnyi veremhatár
EC4F ED5B2617	LD	DE, (1726)	és a TOP
EC53 B7	OR	A	
EC54 ED52	SBC	HL, DE	A távolság
EC56 110001	LD	DE, 0100	
EC59 ED52	SBC	HL, DE	Még 0100H-val kisebb
EC5B 11FF7F	LD	DE, 7FFF	
EC5E ED52	SBC	HL, DE	-7FFFH
EC60 D5	PUSH	DE	
EC61 CD2BF8	CALL	FA2B	HL-t a verembe teszi,
EC64 E1	POP	HL	
EC65 CD2BF8	CALL	FA2B	7FFFH-t is,
EC68 C393F4	JP	F493	majd összeadja és kész

IN A megadott fizikai portról beolvasott adatot tölti a BASIC verembe

EC6B 44EC	Az előző szimbólum címe:	EC44
EC6D 02	Az azonosító hossza:	02
EC6E 49 4E	neve:	IN
EC70 0A	típusa:	beépített numerikus

EC71 CD68EA	CALL EA68	Kiértékeli a kifejezést
EC74 CD1AFB	CALL FB1A	A megadott értéket konvertálja A-ba
EC77 4F	LD C,A	
EC78 ED68	IN L,(C)	Beolvassa a portot
EC7A 2600	LD H,00	
EC7C D32BFA	JP FA2B	HL-t a verembe konvertálja
EC7F D0	RET NC	
EC80 17	RLA	
EC81 D0	RET NC	
EC82 CD1BFA	CALL FA1B	Ha az érték nem a megfelelő tartományban van,
EC85 DF	RST 18	a 10 63. hatványát tölti a verembe (dec.)
EC86 8527	: D39	
EC88 C9	RET	

INKEY\$ Ez is önálló függvény, aktiválása nem a szimbólumtáblán keresztül, hanem önálló tokenel történik

EC89 CD43FC	CALL FC43	Lehívja a következő utasítást
EC8C F793	RST 30: 93	Beolvassa a billentyűzet állapotát
EC8E D7	RST 10	
EC8F FDE5	PUSH IY	
EC91 E1	POP HL	
EC92 2B	DEC HL	
EC93 79	LD A,C	A beolvasott adat
EC94 B7	OR A	Ha nincs lenyomott billentyű: ugrása előre
EC95 2806	JR Z,EC9D	Ha van: beolvassa a kódját, leteszi a verembe,
EC97 F791	RST 30: 91	
EC99 D7	RST 10	
EC9A 71	LD (HL),C	
EC9B 2B	DEC HL	
EC9C 3C	INC A	(A=01)
EC9D 77	LD (HL),A	elé a hosszt, amely 00, ha nincs lenyomott gomb,
EC9E 2B	DEC HL	végül a szövegazonosítót
EC9F 3601	LD (HL),01	
ECA1 E5	PUSH HL	
ECA2 FDE1	POP IY	
ECA4 C9	RET	

Az INKEY\$ tehát a "" üres karakterlánc kódját adja vissza akkor, ha semmit sem nyomunk le, egyébként a lenyomott billentyű kódját.

INT A megadott kifejezés értékének egészrészét teszi a verembe

ECA5 6BEC Az előző szimbólum címe: EC6B
ECA7 03 Az azonosító hossza: 03
ECA8 49 4E 54 neve: INT
ECAB 0A típusa: beépített numerikus

ECAC CD68EA CALL EA68 Kiértékeli a kifejezést
ECAE FD7E08 LD A,(IY+08) Az exponens
ECB2 E67F AND 7F
ECB4 D640 SUB 40
ECB6 3820 JR C,ECB8 Ha 1-nél kisebb tört: ugrás előre
ECB8 3C INC A
ECB9 F5 PUSH AF
ECBA CD07F7 CALL F707 A negatív számot szorozza (-1)-gyel
ECBD CDDFF9 CALL F9DF Ha a mantissza 00, akkor az exponenst is nullázza
ECC0 F1 POP AF
ECC1 C603 ADD A,03
ECC3 47 LD B,A
ECC4 3E00 LD A,00
ECC6 DCE3F7 CALL C,F7E3
ECC9 04 INC B
ECCA 78 LD A,B
ECCB FE10 CP 10
ECCD 38F5 JR C,ECC4
ECCF CD07F7 CALL F707
ECD2 CDDFF9 CALL F9DF
ECD5 C334F7 JP F734 Normalizálás

Ha a szám -1 és +1 közötti tört, akkor:

ECDB FD7E08 LD A,(IY+08) Pozitív szám esetén 0-t ír ki,
ECDB B7 OR A ha pedig negatív, akkor előbb
ECDC F2F9F9 JP P,F9F9 1.0-t ír be, majd szorozza
ECDF CD08FA CALL FA08 (-1)-gyel
ECE2 C326F7 JP F726

IO Nem deklarált függvény. El lehet gondolkodni, hogy mit lehet kezdeni vele!

ECE5 A5EC Az előző szimbólum címe: ECA5
ECE7 02 Az azonosító hossza: 02
ECE8 49 4F neve: IO
ECEA 0A típusa: beépített numerikus

ECEB CD68EA	CALL EA68	Kiértékeli a kifejezést
ECEE CD1AFB	CALL FB1A	Az értéket A-ba konvertálja,
ECF1 6F	LD L,A	
ECF2 2600	LD H,00	majd HL-be, s ezután
ECF4 114119	LD DE,1941	
ECF7 29	ADD HL,HL	az érték kétszeresét hozzáadja
ECF8 19	ADD HL,DE	1941H-hoz,
ECF9 5E	LD E,(HL)	
ECFA 23	INC HL	
ECFB 56	LD D,(HL)	az ezen a címen kezdődő 2 byte
ECFC EB	EX DE,HL	tartalmát teszi a verembe
ECFD C32BFA	JP FA2B	

LEN A megadott szövegkifejezés hosszát hagyja a veremben

ED00 E5EC	Az előző szimbólum címe:	ECE5
ED02 03	Az azonosító hossza:	03
ED03 4C 45 4E	neve:	LEN
ED06 0A	típusa:	beépített numerikus

ED07 CD75EA	CALL EA75	Kiértékeli a szövegkifejezést,
ED0A FDE5	PUSH IY	majd
ED0C CD21FA	CALL FA21	a mutatót visszaállítva
ED0F E1	POP HL	érvényteleníti
ED10 23	INC HL	
ED11 6E	LD L,(HL)	A szöveg hossza
ED12 2600	LD H,00	
ED14 C32BFA	JP FA2B	A hosszt konvertálja a verembe
		és kész

LOG A megadott kifejezés természetes alapú logaritmusát hagyja a veremben

ED17 00ED	Az előző szimbólum címe:	ED00
ED19 03	Az azonosító hossza:	03
ED1A 4C 4F 47	neve:	LOG
ED1D 0A	típusa:	beépített numerikus

ED1E CD68EA	CALL EA68	Kiértékeli a kifejezést,
ED21 CDC3EA	CALL EAC3	majd az X regiszterben helyezi
		el, a stack üres
ED24 21C519	LD HL,19C5	Az első 2 jegy címe
ED27 AF	XOR A	
ED28 57	LD D,A	
ED29 B6	OR (HL)	A szám 0?

ED2A CA14FB	JP	Z,FB14	Ha igen a szám 0: hiba!
ED2D 23	INC	HL	
ED2E 5E	LD	E, (HL)	Az exponens
ED2F 1C	INC	E	
ED30 1D	DEC	E	
ED31 FA14FB	JP	M,FB14	Ha negatív: hiba!
ED34 363F	LD	(HL),3F	Csak a törtrészt nézzük
ED36 D5	PUSH	DE	
ED37 E5	PUSH	HL	
ED38 DF	RST	18	
ED39 06	:	X	Az eredeti érték: x
ED3A 8520	:	D32	x;D32
ED3C CD93F6	CALL	F693	Összehasonlítás
ED3F E1	POP	HL	
ED40 2803	JR	Z,ED45	Ha egyenlők: ugrás
ED42 F249ED	JP	P,ED49	Ha x nagyobb: ugrás

Ha egyenlők, vagy ha x kisebb:

ED45 D1	POP	DE	E=exponens
ED46 1B	DEC	DE	
ED47 34	INC	(HL)	Az X regiszter exponense
ED48 D5	PUSH	DE	
ED49 DF	RST	18	
ED4A 06	:	X	Legyen ismét: x
ED4B 0500	:	D0	x; 0.5
ED4D 03	:	-	x-0.5
ED4E 0500	:	D0	x-0.5; 0.5
ED50 03	:	-	x-1
ED51 06	:	X	x-1; x
ED52 0501	:	D1	x-1; x; 1.0
ED54 00	:	+	x-1; x+1
ED55 01	:	/	(x-1)/(x+1) = p
ED56 08	:	SX	p-t X-be másolja
ED57 0C	:	DUP	p;p
ED58 0C	:	DUP	p;p;p
ED59 02	:	*	p; p*p = q
ED5A 08	:	SX	p;q (X-ben is)
ED5B 0514	:	D20	p;q;D20
ED5D 02	:	*	p; q*D20
ED5E 0513	:	D19	p; q*D20; D19
ED60 00	:	+	p; D20*q+D19
ED61 06	:	X	p; D20*q+D19; q
ED62 02	:	*	p; (D20*q+D19)*q
ED63 0512	:	D18	p; (D20*q+D19)*q; D18
ED65 00	:	+	p; (D20*q+D19)*q+D18
ED66 06	:	X	p; (D20*q+D19)*q+D18; q
ED67 02	:	*	p; ((D20*q+D19)*q+D18)*q
			legyen ez: s
ED68 06	:	X	p;s;q
ED69 0517	:	D23	p;s;q;D23

```

ED6B 00      : +      P; s; q+D23
ED6C 06      : X      P; s; q+D23; q
ED6D 02      : *      P; s; (q+D23)*q
ED6E 0516    : D22    P; s; (q+D23)*q; D22
ED70 00      : +      P; s; (q+D23)*q+D22
ED71 06      : X      P; s; (q+D23)*q+D22; q
ED72 02      : *      P; s; ((q+D23)*q+D22)*q
ED73 0515    : D21    P; s; ((q+D23)*q+D22)*q;
          D21
ED75 00      : +      P; s; n
          n=((q+D23)*q+D22)*q+D21
ED76 01      : /      P; s/n
ED77 0518    : D24    P; s/n; D24
ED79 00      : +      P; s/n+D24
ED7A 82      : *      p*(s/n+D24)
ED7B E1      POP    HL      HL=a kitevő
ED7C 113F00  LD     DE,003F
ED7F B7      OR     A
ED80 ED52    SBC   HL,DE
ED82 CD2BFA  CALL  FA2B      HL-t a stackbe tölti: k
ED85 DF      RST   18
ED86 00
ED87 0511    : +      p*(s/n+D24)+k
ED89 82      : D17    p*(s/n+D24)+k; D17
ED8A C9      : *      (p*(s/n+D24)+k)*D17
          RET     Ez a végső érték

```

Osztási maradék kiszámítása

Bár BASIC-ből nem érhető el, de sokszor nagyon jó szolgálatot tehet a következő kis szubrutin, amely a verem utolsó előtti számának az utolsóval való osztási maradékát hagyja a veremben. (Egész számok esetén modulo számítás.)

```

ED8B CDA6FA  CALL  FAA6      A verem tetején két szám
          duplikálása: a;b helyett
          a; b; a; b
ED8E CDD2EA  CALL  EAD2      b-t X-be másolja
ED91 CDF8F5  CALL  F5FB      Osztás után: a; b; a/b
ED94 CDAFEC  CALL  ECAF      INT után: a; b; INT(a/b)
ED97 BF      RST   18
ED98 02      : *      a; INT(a/b)*b
ED99 03      : -      a-INT(a/b)*b = m
ED9A 8C      : DUP    m; m
ED9B FDCB08BE RES  7,(IY+08) m; ABS(m)
ED9F CD9FEA  CALL  EA9F      m; ABS(m); b
          (mert X-t a verembe másolta)

EDA2 CD93F6  CALL  F693      Összehasonlítás
EDA5 F8      RET   M        A veremben m marad
EDA6 C3F9F9  JP    F9F9      (esetleg: 0)

```

Ismét egy önálló, nem a szimbólumláncból elérhető szubrutin következik.

ORD A megadott szöveg első karakterének kódját hagyja a veremben

EDA9 CD43FC	CALL FC43	Lehívja a következő utasítást
EDAC CD75EA	CALL EA75	és kiértékeli a megadott
EDAF FD05	PUSH IY	szövegkifejezést, majd a
EDB1 CD21FA	CALL FA21	mutatóval az elejére állva
EDB4 E1	POP HL	érvényteleníti
EDB5 23	INC HL	
EDB6 7E	LD A, (HL)	Ezután veszi a szöveg hosszát,
EDB7 B7	OR A	amely
EDB8 CA14FB	JP Z, FB14	ha 00: hiba, egyébként a
EDBB C310ED	JP ED10	következő karakter kódját a
		verembe teszi

PEEK

EDBE 17ED	Az előző szimbólum címe:	ED17
EDC0 04	Az azonosító hossza:	04
EDC1 50 45 45 4B	neve:	PEEK
EDC5 0A	típusa:	beépített numerikus

EDC6 CD68EA	CALL EA68	Feldolgozza a kifejezést
EDC9 CD02FD	CALL FD02	Ellenőrzi a címet
EDCC CDE7FF	CALL FFE7	VID címzés esetén a cím
EDCF 3803	JR C, EDD4	átalakítása és a
EDD1 F3	DI	videomemória belapozása
EDD2 D302	OUT (02), A	
EDD4 6E	LD L, (HL)	PEEK
EDD5 3E70	LD A, 70	Visszaállítja az eredeti
EDD7 D302	OUT (02), A	memóriakonfigurációt és az
EDD9 FB	EI	értéket konvertálja a BASIC
EDDA C312ED	JP ED12	verembe

PI A BASIC verembe tölti a PI konstanst

EDDD BEED	Az előző szimbólum címe:	EDBE
EDDF 02	Az azonosító hossza:	02
EDE0 50 49	neve:	PI
EDE2 0A	típusa:	beépített numerikus

EDE3 DF	RST 18	
EDE4 8522	: D34	PI értékét a verembe tölti és
EDE6 C9	RET	kész

RND

EDE7 DDED	Az előző szimbólum címe:	EDDD
EDE9 03	Az azonosító hossza:	02
EDEA 52 4E 44	neve:	RND
EDED 0A	típusa:	beépített numerikus

EDEE FE96	CP 96	Ha utána "("-es paraméterezés
EDFO 2810	JR Z,EE02	áll: ugrás előre
EDF2 CDD8E6	CALL E6D8	Az RND törzsrutinja
EDF5 CD2BFA	CALL FA2B	HL-t verembe viszi: x
EDF8 DF	RST 18	
EDF9 0526	: D38	x; 32768 (dec)
EDFB 00	: +	x+32768
EDFC 0526	: D38	x+32768; 32768
EDFE 0C	: DUP	x+32768; 32768; 32768
EDFF 00	: +	x+32768; 65536
EE00 81	: /	(x+32768)/65536
EE01 C9	RET	

Ekkor a stackben egy nyilvánvalóan 0...1 közötti érték lehet. Ha hívásnál egy zárójeles kifejezést is megadtak, akkor a rutin az előbbieket helyett a következő:

EE02 CD68EA	CALL EA68	Értékeli a kifejezést
EE05 CDC3FA	CALL FAC3	
EE08 CB7C	BIT 7,H	HL-be konvertálja a számot,
EE0A CZ14FB	JP NZ,FB14	amely 7FFFH-nál nem lehet
EE0D 7C	LD A,H	nagyobb és 0 sem, különben
EE0E B5	OR L	hibajelzést kapunk
EE0F CA14FB	JP Z,FB14	
EE12 E5	PUSH HL	
EE13 AF	XOR A	Ezután HL-t addig duplázza, amíg
EE14 3C	INC A	túlcsordulás nem lesz, közben
EE15 29	ADD HL,HL	számlálja a duplázások számát
EE16 30FC	JR NC,EE14	
EE18 3D	DEC A	
EE19 E1	POP HL	
EE1A 2B	DEC HL	A kapott paraméterekkel fog
EE1B E5	PUSH HL	dolgozni
EE1C F5	PUSH AF	
EE1D CDD8E6	CALL E6D8	HL-ben egy RND érték
EE20 F1	POP AF	

EE21 47	LD B,A	Most visszafelé: a kapott
EE22 CB3C	SRL H	értéket 2-vel osztja addig,
EE24 CB1D	RR L	ahányszor az előbb szorozta
EE26 10FA	DJNZ EE22	
EE28 EB	EX DE,HL	Ha az így kapott érték mégis
EE29 E1	POP HL	nagyobb, mint az eredeti szám
EE2A B7	OR A	-1, akkor megismétli az
EE2B ED52	SBC HL,DE	eljárást
EE2D 19	ADD HL,DE	
EE2E 38EB	JR C,EE1B	
EB3E 0C	: DUF	d;d
EE30 EB	EX DE,HL	Egyébként a nyert érték a
EE31 C32BFA	JP FA2B	verembe kerül és kész

SGN A kifejezés előjelét teszi a stackbe

EE34 E7ED	Az előző szimbólum címe:	ED7E
EE36 03	Az azonosító hossza:	03
EE37 53 47 4E	neve:	SGN
EE3A 0A	típusa:	beépített numerikus

EE3B CD68EA	CALL EA68	Kiértékeli a kifejezést
EE3E FD7E06	LD A,(IY+06)	Ha a szám értéke eleve zérus,
EE41 B7	OR A	akkor kész
EE42 C8	RET Z	
EE43 FD7E08	LD A,(IY+08)	Az exponens,
EE46 F5	PUSH AF	ezt elteszi
EE47 CD08FA	CALL FA08	A szám helyére 1.0-t ír
EE4A F1	POP AF	
EE4B E680	AND 80	
EE4D FDB608	OR (IY+08)	Bemaszkolja az előjelet
EE50 FD7708	LD (IY+08),A	és kész
EE53 C9	RET	

SIN

EE54 34EE	Az előző szimbólum címe:	EE34
EE56 03	Az azonosító hossza:	03
EE57 53 49 4E	neve:	SIN
EE5A 0A	típusa:	beépített numerikus

EE5B CD68EA	CALL EA68	Kiértékeli a kifejezést,
EE5E DF	RST 18	legyen ez: x
EE5F 0522	: D34	x; D34=PI
EE61 0522	: D34	x; PI; PI
EE63 80	: +	x; 2*PI

EE64 CD8BED	CALL ED8B	Veszi az $x \cdot 2 \cdot \pi$ -vel való osztási maradékát: s
EE67 FD7E08	LD A, (IY+08)	Az s exponense
EE6A B7	OR A	
EE6B 1F	RRA	
EE6C F5	PUSH AF	
EE6D DF	RST 18	
EE6E 8A	: SXD	Az s-t az X-be viszi
EE6F DF	RST 18	
EE70 06	: X	s
EE71 8523	: D35	s; D35= $\pi/2$
EE73 F1	POP AF	
EE74 F5	PUSH AF	
EE75 FC26F7	CALL M, F726	*(-1)
EE78 CD93F6	CALL F693	Összehasonlítás
EE7B C1	POP BC	
EE7C C5	PUSH BC	A további végrehajtás a két érték viszonyától függ
EE7D 281B	JR Z, EE9A	
EE7F A8	XOR B	
EE80 FA93EE	JP M, EE93	
EE83 DF	RST 18	
EE84 06	: X	s;
EE85 8522	: D34	s; π
EE87 F1	POP AF	
EE88 EE40	XOR 40	
EE8A F5	PUSH AF	
EE8B F426F7	CALL P, F726	*(-1)
EE8E DF	RST 18	
EE8F 00	: +	s+ π , vagy s- π
EE90 8A	: SXD	Tárolja X-ben, legyen: z
EE91 18DC	JR EE6F	
EE93 F1	POP AF	Még egy előjelvizsgálat, ill. beállítás
EE94 EE80	XOR 80	
EE96 F5	PUSH AF	
EE97 FA6FEE	JP M, EE6F	

Most már az X regiszterben közvetlen feldolgozásra alkalmas szög van.

EE9A DF	RST 18	
EE9B 06	: X	z
EE9C 0C	: DUP	z; z
EE9D 0C	: DUP	z; z; z
EE9E 0C	: DUP	z; z; z; z
EE9F 02	: *	z; z; z * z = w
EEA0 08	: SX	A w-t X-be másolja
EEA1 051D	: D29	z; z; w; D29
EEA3 02	: *	z; z; w * D29
EEA4 051C	: D28	z; z; w * D29; D28
EEA6 00	: +	z; z; D29 * w + D28
EEA7 06	: X	z; z; D29 * w + D28; w

EEA8 02	:	*	z; z; (D29*w+D28)*w
EEA9 051B	:	D27	z; z; (D29*w+D28)*w; D27
EEAB 00	:	+	z; z; (D29*w+D28)*w+D27
EEAC 06	:	X	z; z; (D29*w+D28)*w+D27; w
EEAD 02	:	*	z; z; ((D29*w+D28)*w+D27)*w
EEAE 051A	:	D26	z; z; ((D29*w+D28)*w+D27)* *w; D26
EEB0 00	:	+	z; z; ((D29*w+D28)*w+D27)* *w+D26
EEB1 06	:	X	z; z; ((D29*w+D28)*w+D27)* *w+D26; w
EEB2 02	:	*	z; z; (((D29*w+D28)*w+D27) *w+D26)*w
EEB3 0519	:	D25	
EEB5 00	:	+	Az előbbiekkal megegyező
EEB6 06	:	X	műveleteket végzünk a D25-tel
EEB7 02	:	*	is, majd
EEB8 02	:	*	és
EEB9 80	:	+	után végül a veremben:

(((((D29*w+D28)*w+D27)*w+D26)*w+D25)*w*z+z

EEBA F1	POP	AF	
EEBB 87	ADD	A, A	
EEBC F0	RET	P	Ha az előjel pozitív volt: kész,
EEBD C326F7	JP	F726	egyébként az eredménynek is a (-1)-szerese kell

SQR

EEC0 54EE	Az előző szimbólum címe:	EE54
EEC2 03	Az azonosító hossza:	03
EEC3 53 51 52	neve:	SQR
EEC6 0A	típusa:	beépített numerikus

EEC7 CD68EA	CALL	EA68	Kiértékeli a kifejezést
EECA FDCB087E	BIT	7, (IY+08)	A szám exponense
EECE C214FB	JP	NZ, FB14	Negatív szám: hiba!
EED1 FD7E06	LD	A, (IY+06)	Az első jegyek
EED4 B7	OR	A	Ha a szám 0, akkor az eredmény
EED5 C8	RET	Z	is 0
EED6 CDC3EA	CALL	EAC3	A számot az X-be tölti
EED9 21C619	LD	HL, 19C6	Az X exponense
EEDC 7E	LD	A, (HL)	
EEDD D63F	SUB	3F	

EEDF F5	PUSH AF	
EEEE 363F	LD (HL),3F	Így az X-ben 0...1 közötti szám van: x
EEE2 DF	RST 18	
EEE3 06	: X	x
EEE4 051F	: D31	x; D31
EEE6 02	: *	D31*x
EEE7 051E	: D30	D31*x; D30
EEE9 00	: +	D31*x+D30
EEEE 8B	: SYD	Ezt menti Y-ba, a verem üres
EEEB 0604	LD B,04	Egy ciklus kezdődik
EEED C5	PUSH BC	
EEEE DF	RST 18	
EEEF 07	: Y	y
EEF0 06	: X	y;x
EEF1 07	: Y	y;x;y
EEF2 01	: /	y; x/y
EEF3 07	: Y	y; x/y; y
EEF4 03	: -	y; x/y-y
EEF5 0500	: D0	y; x/y-y; 0.5
EEF7 02	: *	y; (x/y-y)*0.5
EEF8 00	: +	y+0.5*(x/y-y)
EEF9 8B	: SYD	Ez lesz az Y új értéke
EEFA C1	POP BC	
EEFB 10F0	DJNZ EEED	A közelítést ismételve
EEFD F1	POP AF	
EEFE CB47	BIT 0,A	
EF00 2809	JR Z,EF0B	Ha páros, ugrás előre
EF02 F5	PUSH AF	
EF03 DF	RST 18	
EF04 07	: Y	y, az előbb kapott érték
EF05 0520	: D32	y; D32
EF07 02	: *	y*D32
EF08 8B	: SYD	Ez lesz a végső Y
EF09 F1	POP AF	
EF0A 3C	INC A	
EF0B CB3F	SRL A	Gyökvonás az eredeti exponensben (osztás 2-vel)
EF0D 21CD19	LD HL,19CD	Y exponense
EF10 86	ADD A,(HL)	Az eredmény exponense
EF11 E67F	AND 7F	
EF13 77	LD (HL),A	
EF14 C39AEA	JP EA9A	Az eredményt Y-ból a verembe másolja és kész

HATVÁNYOZÁS

Önálló szubrutin, nincs a szimbólumtáblába láncolva, saját tokenje alapján éri el a rendszer (9FH)

EF17 FD7E06	LD A, (IY+06)	A kitevő első jegyei
EF1A B7	OR A	
EF1B 2006	JR NZ, EF23	Ha a kitevő 0, akkor az
EF1D CD1BFA	CALL FA1B	eredményként 1.0-t készít elő a
EF20 C308FA	JP FA08	veremben
EF23 FD7E0F	LD A, (IY+0F)	Az alap első jegyei
EF26 B7	OR A	
EF27 CA1BFA	JP Z, FA1B	Ha az alap 0, akkor az eredmény
		is 0 lesz
EF2A FD7E08	LD A, (IY+08)	A kitevő exponense
EF2D E67F	AND 7F	
EF2F FE43	CP 43	Ha kitevő 1000-nél nagyobb:
EF31 304A	JR NC, EF7D	ugrás előre
EF33 CD92FA	CALL FA92	DUP
EF36 CDC3FA	CALL FAC3	HL-be konvertálja
EF39 E5	PUSH HL	
EF3A CD92FA	CALL FA92	DUP
EF3D CD67F7	CALL F767	Kerekítés a szám végén
EF40 E1	POP HL	
EF41 E5	PUSH HL	
EF42 CD2BFA	CALL FA2B	HL-t visszateszi a verem
		tetejére
EF45 CD93F6	CALL F693	Összehasonlítás
EF48 E1	POP HL	
EF49 2032	JR NZ, EF7D	Ha nem egyenlők: ugrás
EF4B CB7C	BIT 7, H	
EF4D F5	PUSH AF	
EF4E C486FC	CALL NZ, FC86	HL-t komplementálja
EF51 E5	PUSH HL	
EF52 DF	RST 1B	
EF53 0A	: SxD	
EF54 0A	: SxD	Az alapot X-be teszi
EF55 8501	: D1	1.0
EF57 E1	POP HL	
EF58 E5	PUSH HL	
EF59 CB45	BIT 0, L	
EF5B 2806	JR Z, EF63	
EF5D CD9FEA	CALL EA9F	x
EF60 CD12F5	CALL F512	Szorzás
EF63 E1	POP HL	
EF64 CB3C	SRL H	
EF66 CB1D	RR L	
EF68 7C	LD A, H	Az alap hatványozása,
EF69 B5	OR L	amíg HL=0000 lesz
EF6A 2808	JR Z, EF74	
EF6C E5	PUSH HL	

EF6D DF	RST	18	
EF6E 06	:	X	
EF6F 06	:	X	
EF70 02	:	*	
EF71 8A	:	SXD	
EF72 18E3	JR	EF57	
EF74 F1	POP	AF	Pozitív egész kitevő esetén
EF75 C8	RET	Z	kész is
EF76 DF	RST	18	
EF77 0B	:	SYD	Y-ba menti
EF78 0501	:	D1	1.0
EF7A 07	:	Y	1/Y: negatív egész kitevő
EF7B 81	:	/	esetén az előbbi érték
EF7C C9	RET		reciproka kell

Ha a kitevő nem egész szám, akkor a számolás a logaritmusművelet felhasználásával történik:

EF7D CDBEEA	CALL	EABE	A kitevőt Y-ba tölti
EF80 CD21ED	CALL	ED21	Veszi az alap logaritmusát,
EF83 CD9AEA	CALL	EA9A	a verembe visszateszi a
			kitevőt,
EF86 CD12F5	CALL	F512	ezeket összeszorozza, és ez
EF89 C3B8EB	JP	EBB8	lesz a kitevő, amellyel
			elvégzi az EXP műveletet

STR\$

EF8C C0EE	Az előző szimbólum címe:	EEC0
EF8E 04	Az azonosító hossza:	04
EF8F 53 54 52 24	neve:	STR\$
EF93 08	típusa:	beépített szöveges

EF94 CD68EA	CALL	EA68	Kiértékeli a megadott
			számkifejezést
EF97 CD0EF8	CALL	F80E	A kapott számot ASCII kódo-
			lással a 1930H pufferbe teszi,
EF9A CD1BF8	CALL	FA1B	a veremből törli
EF9D C37DF8	JP	FA7D	Végül a pufferből szöveggént a
			stackbe viszi

STRING\$

Ez a függvény kétféleképpen adható meg:

- STRING\$(n,k): n darab k ASCII kódú karaktert jelent
- STRING\$(n,x\$): n darab olyan karakter, mellyel x\$ kezdődik

```
EFA0 8CEF      Az előző szimbólum címe:    EF8C
EFA2 07        Az azonosító hossza:     07
EFA3 53 54 52 49 4E 47 24      neve:    STRING$
EFAA 08        típusa:    beépített szöveges
```

```
EFAB 3E96      LD    A,96      A kulcsszó után először csak
EFAD CD54FD    CALL FD54      "("-t fogad el
EFB0 CD1BFB    CALL FB1B      A kért ismétlési számot (n) az
EFB3 F5        PUSH AF        A-ba konvertálja
EFB4 3EA4      LD    A,A4      A következő parancsbyte csak
EFB6 CD54FD    CALL FD54      "," karakter lehet
EFB9 FE02      CP    02        Ha a "," után szám áll,
EFBB 3013      JR    NC,EFD0   ugrás előre, ha nem:
EFBD CD94F2    CALL F294      feldolgozza a szöveget
EFC0 FD7E01    LD    A,(IY+01)
EFC3 B7        OR    A        Ha a megadott szöveg üres: ugrás
EFC4 2807      JR    Z,EFCD   előre
EFC6 FD7E02    LD    A,(IY+02)
EFC9 CD21FA    CALL FA21      Az első karakter
EFCC 01C1F5    LD    BC,F5C1  A szöveget a veremből törli
                (érvényteleníti)
```

Ez itt hatástalan utasítás, de ha a megadott szöveg üres, akkor a 0EFCDH címre ugrik a CPU, s ott a C1-nek és F5-nek megfelelő utasításokat hajtja végre:

```
C1    POP  BC
F5    PUSH AF
```

Ez azt jelenti, hogy üres szöveg esetén a megadott ismétlési számot is 0-ra írja át!

```
EFCF 37        SCF
EFDO D41BFB    CALL NC,FB1B   Ha eredetileg magát a kódot adták
                meg, azt előállítja A-ban

EFD3 F5        PUSH AF
EFD4 CD70EA    CALL EA70      Ezután csak ")" állhat
EFD7 D1        POP  DE        D=az ASCII kód
EFD8 FDE5      PUSH IY
EFDA E1        POP  HL
EFDB 2B        DEC  HL
EFD4 F1        POP  AF        A=az ismétlési szám
EFD0 3C        INC  A
```

EFDE CA12F9	JP	Z,F912	Ha a hossz 0FFH: hiba!
EFE1 3D	DEC	A	
EFE2 2805	JR	Z,EFE9	üres szöveg lesz
EFE4 47	LD	B,A	
EFE5 72	LD	(HL),D	Leteszi a kódot n-szer
EFE6 2B	DEC	HL	
EFE7 10FC	DJNZ	EFE5	
EFE9 77	LD	(HL),A	végül a szöveg hossza és
EFEA 2B	DEC	HL	
EFEB 3601	LD	(HL),01	az azonosító kerül a verembe
EFED E5	PUSH	HL	
EFEE FDE1	POP	IY	
EFF0 C9	RET		

Az utolsó trigonometrikus függvény: a tangens következik. Ennek számolása a szinusz- és koszinusz-függvénnyel fennálló kapcsolata alapján történik, tehát a korábbiakra való egyszerű hivatkozásokkal megoldható.

TAN

EFF1 A0EF	Az előző szimbólum címe:	EFA0
EFF3 03	Az azonosító hossza:	03
EFF4 54 41 4E	neve:	TAN
EFF7 0A	típusa:	beépített numerikus

EFF8 CD68EA	CALL	EA68	Kiértékeli a kifejezést
EFFB CDCDEA	CALL	EACD	Y-ba tölti, legyen: y
EFFE CD5EEE	CALL	EE5E	SIN(y)
F001 DF	RST	18	
F002 07	:	Y	SIN(y); y
F003 0523	:	D35	SIN(y); y; PI/2
F005 80	:	+	SIN(y); y+PI/2
F006 CD5EEE	CALL	EE5E	SIN(y); SIN(y+PI/2)
F009 C3FBF5	JP	F5FB	Elvégzi az osztást

USR

F00C F1EF	Az előző szimbólum címe:	EFF1
F00E 03	Az azonosító hossza:	03
F00F 55 53 52	neve:	USR
F007 0A	típusa:	beépített numerikus

F013 3E96	LD A,96	Az USB után csak "("-t fogad el
F015 CD54FD	CALL FD54	Kiértékeli a kifejezést ")"?
F018 CDA7F0	CALL FOA7	
F01B FE95	CP 95	
F01D 280A	JR Z,F029	Ha igen: ugrás előre
F01F 3EA4	LD A,A4	Ha még nem volt ")", akkor csak
F021 CD54FD	CALL FD54	","-t fogad el
F024 CDC4FA	CALL FAC4	A-ban a megadott érték
F027 3E95	LD A,95	Ezután már mindenképpen csak
F029 CD54FD	CALL FD54	"," lehet!
F02C 112BFA	LD DE,FA2B	
F02F D5	PUSH DE	
F030 E5	PUSH HL	
F031 CD02FD	CALL FD02	Ellenőrzi a címet
F034 E3	EX (SP),HL	
F035 C9	RET	Mehet tovább

VAL

F036 0CF0	Az előző szimbólum címe:	F00C
F038 03	Az azonosító hossza:	03
F039 56 41 4C	neve:	VAL
F03C 0A	típusa:	beépített numerikus

F03D CD75EA	CALL EA75	Feldolgozza a szövegkifejezést
F040 FDE5	PUSH IY	
F042 E1	POP HL	
F043 23	INC HL	
F044 E5	PUSH HL	
F045 4E	LD C,(HL)	A szöveg hossza
F046 0600	LD B,00	
F048 09	ADD HL,BC	A szöveg vége
F049 23	INC HL	
F04A 7E	LD A,(HL)	Az előző elemazonosító
F04B 70	LD (HL),B	00: a string vége jelzés
F04C E3	EX (SP),HL	
F04D F5	PUSH AF	
F04E 23	INC HL	
F04F CD14F9	CALL F914	Elvégzi az ASCII karakterekkel
F052 CD7FEC	CALL EC7F	megadott szám konverzióját
F055 F1	POP AF	
F056 E1	POP HL	
F057 77	LD (HL),A	Helyreállítja az előző elem
F058 2B	DEC HL	azonosítóját
F059 EB	EX DE,HL	
F05A FDE5	PUSH IY	
F05C E1	POP HL	
F05D 010900	LD BC,0009	A számmá konvertált szöveget
F060 09	ADD HL,BC	bemozgatja a korábbi szöveg
		helyére

F061 2B	DEC HL	
F062 EDB8	LDDR	
F064 13	INC DE	
F065 D5	PUSH DE	Beállítja a veremmutatót és
F066 FDE1	POP IY	kész
F068 C9	RET	

VARPTR Az verembe a megadott változó memóriacímét teszi

F069 36F0	Az előző szimbólum címe:	F036
F06B 06	Az azonosító hossza:	06
F06C 56 41 52 50 54 52	neve:	VARPTR
F072 0A	típusa:	beépített numerikus

F073 3E96	LD A,96	A kulcsszó után csak "(" lehet,
F075 CD54FD	CALL FD54	a zárójelben pedig egy változó
F078 E6FD	AND FD	nevének kell állnia, különben
F07A FE01	CP 01	hiba
F07C C214FB	JP NZ,FB14	
F07F D9	EXX	
F080 79	LD A,C	A keresett változó típusát
F081 E67F	AND 7F	leteszi,
F083 320817	LD (1708),A	azonban nem lehet beépített
F086 E60C	AND 0C	függvény, vagy DEF utasítás
F088 20F2	JR NZ,F07C	azonosítója
F08A D9	EXX	
F08B CD2EF4	CALL F42E	Megkeresi az adat címét a
F08E CD2BFA	CALL FA2B	szimbólumtáblában
F091 C370EA	JP EA70	A címet a verembe konvertálja
		Végül csak ")"-t fogad el

VERNUM Az alkalmazott BASIC rendszer verziószámát adja

F094 69F0	Az előző szimbólum címe:	F069
F096 06	Az azonosító hossza:	06
F097 56 45 52 4E 55 4D	neve:	VERNUM
F09D 0A	típusa:	beépített numerikus

F09E 210C00	LD HL,000C	HL-t konvertálja a verem
F0A1 C32BFA	JP FA2B	tetejére és kész

Mégpedig annyira kész, hogy ezzel a fejezet ezen részének is a végére értünk.

3.6.7 Alapműveletek és konverziók

Lassan ott tartunk, hogy a BASIC elvi működéséről szinte minden fontos dolgot tudunk -- sőt a legtöbb részletekérdésen is túl vagyunk.

Mi van még?

Azok -- a BASIC szempontjából alacsony szintű -- rutinok, amelyekre az eddigi programrészek oly sokszor hivatkoztak. Az egész számítógép működésének program szintű vezérlésében a hierarchikus felépítés érvényesül. Figyelemre méltó körülmény, hogy az alsóbb szinteken -- távolabb a BASIC bonyolult és speciális struktúrát mutató vezérlő programjaitól -- olyan szubrutinok találhatóak, amelyek csak kis mértékben igénylik ezt a BASIC környezetet, ill. ez a környezet könnyen imitálható! Ezáltal az ilyen szubrutinok már jórészt attól függetlenül, tetszőleges felhasználói környezetben is jól használhatók. A szépen kidolgozott szubrutinok beépítése pedig nagyon megkönnyítheti a programozói munkát!

3.6.7.1 Alacsony szintű rutinok

Kifejezések kiértékelése

A következő, meglehetősen összetett struktúrájú programok a BASIC utasítások végrehajtásában igen fontos szerepet játszanak. Segítségükkel értelmezi és értékeli a rendszer a BASIC programjainkba beírt, tetszőlegesen bonyolult számolási és szövegekkel kapcsolatos műveleteket.

Figyeljük meg, hogy az ismert műveleti sorrend (függvények kiértékelése -- zárójeles kifejezések -- hatványozás -- szorzás, osztás -- összeadás, kivonás -- NOT -- AND -- OR és XOR) hogyan valósul meg az "egymásba ágyazott", egymást hívó szubrutinok segítségével.

Számkifejezések

Belépési pont utasításle hívással

A kifejezéskezelő programrészt akkor hívhatjuk meg ezen a címen, ha biztosak vagyunk benne, hogy a forrás sorban (az aktuális BASIC utasítássorban) csakis egy kifejezés megadása következhet.

FOA4 CD43FC CALL FC43 Lehívja a következő utasítást

Belépési pont lehívott utasítás esetén

Ha itt lépünk be, a korábban lehívott utasítás alapján azonnal elkezdődik a kifejezés feldolgozása.

FOA7 D9	EXX	A jelzőbyte-ot a másodlagos regiszterkészletből A-ba hozzuk
FOA8 78	LD A,B	
FOA9 D9	EXX	
FOAA CDD7F0	CALL F0D7	Először a nagyobb prioritású műveleteket nézzük (AND és efölött)
FOAD D9	EXX	
FOAE 78	LD A,B	Ezután -- tehát legutoljára kerül sor az OR és XOR végrehajtására
FOAF D9	EXX	
FOB0 FEBF	CP BF	OR esetén ugrás a végrehajtó rutinra
FOB2 2804	JR Z,FOB8	
FOB4 FEB2	CP B2	XOR esetén ugyanott kell folytatni, míg ha egyik sincs: visszatér
FOB6 C0	RET NZ	
FOB7 37	SCF	Az XOR-t CY=1 jelzi

OR és XOR végrehajtása

FOB8 F5	PUSH AF	A műveletazonosító byte tárolása után lehívjuk a következő utasítást és szükség esetén elvégezzük az újabb, magasabb rendű műveleteket
FOB9 CD43FC	CALL FC43	
FOBC CDD7F0	CALL F0D7	
FOBF CDBFFA	CALL FABF	A két értéket a BASIC veremből HL-be és DE-be konvertálja
FOC2 F1	POP AF	
FOC3 3807	JR C,FOCC	XOR esetén ugrás előre
FOC5 7D	LD A,L	Itt az OR végrehajtásával folytatja, mint látjuk, a szokásos bitművelet zajlik, külön az alsó és felső byte-tal
FOC6 B3	OR E	
FOC7 6F	LD L,A	
FOC8 7C	LD A,H	
FOC9 B2	OR D	
FOCA 1805	JR F0D1	A befejezés a másik művelettel közös

FOCC 7D	LD A,L	
FOCD AB	XOR E	Itt történik az XOR végrehajtása
FOCE 6F	LD L,A	
FOCF 7C	LD A,H	
FOD0 B2	OR D	Sajnos, így csak az alsó byte
FOD1 67	LD H,A	lett XOR szerinti!
FOD2 CD2BFA	CALL FA2B	Végül az eredményt a HL-ből a
		verembe viszi és
FOD5 18D6	JR FOAD	mehet tovább

AND

FOD7 CDF4F0	CALL FOF4	Előbb a NOT és a magasabb
FODA D9	EXX	prioritású műveleteket végzi el
FODB 78	LD A,B	
FODC D9	EXX	
FODD FEC9	CP C9	Az AND tokenje. Ha nem kell,
FODF C0	RET NZ	visszatér
FOE0 CD43FC	CALL FC43	Lehívja a következő utasítást és
FOE3 CDF4F0	CALL FOF4	ha van, elvégzi az így adódó
FOE6 CDBFFA	CALL FABF	újabb, magasabb rendű
		műveleteket. A OFABFH rutin a
FOE9 7D	LD A,L	BASIC veremből HL-be és DE-be
FOEA A3	AND E	töltötte az adatokat, ezután
FOEB 6F	LD L,A	byte-onként elvégzi az AND
FOEC 7C	LD A,H	műveletet
FOED A2	AND D	
FOEE 67	LD H,A	Az eredményt HL-ből a verembe
FOEF CD2BFA	CALL FA2B	tölti, s mehet
		tovább
FOF2 18E6	JR FODA	

NOT

FOF4 FEC2	CP C2	A NOT tokenje. Ha NOT nincs,
FOF6 2012	JR NZ,F10A	ugrik a relációkhoz
FOF8 CD43FC	CALL FC43	Először itt is továbbmegyünk és
FOFB CDF4F0	CALL FOF4	elvégezzük a többi,
FOFE CDC3FA	CALL FAC3	magasabbrendű műveletet. Utána a
F101 7D	LD A,L	veremből HL-be töltjük a kapott
F102 2F	CPL	értéket és bitjeit megfordítjuk
F103 6F	LD L,A	
F104 7C	LD A,H	
F105 2F	CPL	Visszatérés a HL-t a BASIC
F106 67	LD H,A	verembe konvertáló szubrutinon
F107 C32BFA	JP FA2B	át

Relációk (vezérlőrutin)

F10A FE02	CP	02	Szám -- ha nem, ugrás a
F10C 3815	JR	C,F123	szövegek vizsgálatához
F10E CD55F1	CALL	F155	Először a +,- és a nagyobb
F111 FE99	CP	99	prioritású műveleteket végzi
F113 D8	RET	C	el
F114 FE9F	CP	9F	Ha relációjel nincs, akkor
F116 D0	RET	NC	viSSzatér
F117 F5	PUSH	AF	Ha van: lehívja a következő
F118 CD43FC	CALL	FC43	utasítást, s a magasabb rendű
F11B CD55F1	CALL	F155	műveleteket elvégezve
			előkészíti a két értéket
F11E CD93F6	CALL	F693	Elemi összehasonlítás,
F121 1817	JR	F13A	majd ugrás az értékeléshez

A 0F693H szubrutin a CY, S és Z jelzőbitekben adja vissza az összehasonlítás eredményét. Ha n1 és n2 a BASIC verembe elsőként és másodikként betöltött két szám, akkor CY=1 és S=1, ha n1 kisebb, míg Z=1, ha n1=n2.

Relációk karakterláncokkal (vezérlés)

F123 CD94F2	CALL	F294	Feldolgozza a szövegkifejezést
F126 FE99	CP	99	
F128 DA5DF3	JP	C,F35D	Ha utána nem relációjel
F12B FE9F	CP	9F	áll: "típushiba"
F12D D25DF3	JP	NC,F35D	
F130 F5	PUSH	AF	Utána a következő utasítás
F131 CD43FC	CALL	FC43	lehívásával, majd
F134 CD94F2	CALL	F294	feldolgozásával előkészíti, s
F137 CDD7F6	CALL	F6D7	összehasonlítja a két szöveget

A 0F6D7H rutin ugyanúgy működik, mint a számokat összehasonlító előbbi 0F693H című rutin.

Az eredmény értékelése

F13A E1	POP	HL	A-ba az előírt reláció tokenje
F13B 7C	LD	A,H	kerül, míg CY és Z az
			összehasonlított adatok
F13C CD42F1	CALL	F142	viszonyát mutatja
			Ezek alapján a HL-ben
			előállítjuk az eredményt
			(-1: igaz, 0: hamis),
F13F C32BFA	JP	FA2B	majd áttöltjük a verembe
			és kész

A relációk értékelésének segédrutinja

Most ismét meggyőződhetünk a tokenek átgondolt, következetes megválasztásáról: a relációknak megfeleltetett szimbolikus kódok rövid és szellemes értékelést tesznek lehetővé.

Reláció	Token	B i t e k							
kisebb	99	1	0	0	1	1	0	0	1
egyenlő	9A	1	0	0	1	1	0	1	0
kisebb vagy egyenlő	9B	1	0	0	1	1	0	1	1
nagyobb	9C	1	0	0	1	1	1	0	0
kisebb vagy nagyobb	9D	1	0	0	1	1	1	0	1
nagyobb vagy egyenlő	9E	1	0	0	1	1	1	1	0

nagyobb = kisebb

Mint látjuk, az alsó 3 bit vizsgálatával -- melyre a Z 80-as CPU egyszerű lehetőségeket biztosít -- a jelzőbite alapján könnyen eldönthető a megadott reláció igaz vagy hamis volta.

Nézzük, itt hogyan történik,

```
F142 21FFFF    LD    HL,FFFF    Az eredmény lehet (-1)
F145 0F        RRCA
```

Ezzel az A 0. bitje ("kisebb" jelzése) a CY-ba kerül, míg a korábbi összehasonlítás eredményét a Z és az S mutatja -- tehát mindkettő vizsgálható.

```
F146 3001      JR    NC,F149      Ha nem a "kisebb" reláció volt:
                    ugrás előre
F148 F8        RET   M            Igaz! (Ennyire egyszerű)
F149 0F        RRCA              Az "=" bitje
F14A 3001      JR    NC,F14D      Ha nem ez kell: tovább!
F14C C8        RET   Z            Ha a reláció "=" volt és Z=1:
                    igaz
F14D 0F        RRCA              A "nagyobb" jel volt?
F14E 3003      JR    NC,F153      Ha nem, vagy Z=1, akkor az
F150 2801      JR    Z,F153       állítás hamis, míg S=0 esetén
F152 F0        RET   P            igaz
F153 23        INC  HL            HL=0000:      hamis      állítás
F154 C9        RET                                     beállítása és kész
```

A számolási műveletek vezérlése

```
F155 CD6CF1    CALL F16C          A "+" és "-" vizsgálata
F158 C481F1    CALL NZ,F181      Ha egyik sincs, a magasabb
                    rendű műveleteket intézi el
```

F15B 1C	INC E	E=01: kivonás lesz
F15C C426F7	CALL NZ,F726	Megfordítja az előjelet
F15F CD6CF1	CALL F16C	Újabb utasítások lehívásával és a magasabbrendű műveletek elvégzésével előkészíti a számokat
F162 C0	RET NZ	Ha már nincs több: visszatér
F163 1C	INC E	Az E értéke szerint összeadást vagy
F164 CC93F4	CALL Z,F493	kivonást végez
F167 C48EF4	CALL NZ,F48E	Tovább!
F16A 18F3	JR F15F	

Az összeadás és kivonás előkészítése

F16C D9	EXX	Az utoljára lehívott utasítás jelzőbyte-ját áttölti A-ba
F16D 78	LD A,B	
F16E D9	EXX	
F16F 1EFF	LD E,FF	"Összeadás" jelzése
F171 FE98	CP 98	Ha a token "+": előkészíti a műveletet
F173 2804	JR Z,F179	A "-" esetében szintén
F175 FEA2	CP A2	Ha egyik sem: visszatér
F177 C0	RET NZ	Ha nem "+" volt, akkor most E=00; kivonás kell!
F178 1C	INC E	
F179 CD43FC	CALL FC43	A következő utasítás
F17C CD81F1	CALL F181	lehívásával, majd a magasabb
F17F AF	XOR A	rendű műveletek elvégzésével
F180 C9	RET	előkészíti a számokat az összevonásra

A szorzás és osztás vezérlése

F181 CDA2F1	CALL F1A2	Először a nagyobb prioritású műveleteket nézi
F184 D9	EXX	
F185 78	LD A,B	
F186 D9	EXX	
F187 FEA8	CP A8	A "*" tokenje
F189 2803	JR Z,F18E	Ha ez kell, ugrás előre
F18B FEA1	CP A1	Az "/" tokenje
F18D C0	RET NZ	Ha ez sem: visszatér
F18E 57	LD D,A	D-ben a művelet tokenje
F18F CD43FC	CALL FC43	A következő utasítás
F192 CDA2F1	CALL F1A2	lehívásával és a magasabb
F195 D5	PUSH DE	rendű műveletek elvégzésével
F196 7A	LD A,D	előkészíti a számokat

F197 FEAB	CP	A8	Ezután következik a token
F199 CC12F5	CALL	Z,F512	szerinti szorzás, vagy osztás
F19C C4FBF5	CALL	NZ,F5FB	elvégzése
F19F D1	POP	DE	Az E visszatöltése után mehet
F1A0 18E2	JR	F184	tovább

Hatványozás vezérlése

F1A2 D5	PUSH	DE	Először ismét a nagyobb
F1A3 CDBBF1	CALL	F1BB	prioritású műveletekkel
F1A6 D1	POP	DE	foglalkozunk (változók,
F1A7 D9	EXX		függvények, zárójelek)
F1A8 78	LD	A,B	
F1A9 D9	EXX		
F1AA FE9F	CP	9F	A hatványozás tokenje
F1AC C0	RET	NZ	Ha nem kell: visszatér
F1AD D5	PUSH	DE	
F1AE CD43FC	CALL	FC43	A magasabbrendű műveletek után
F1B1 CDBBF1	CALL	F1BB	meghívja a hatványozás
F1B4 CD17EF	CALL	EF17	rutinját
F1B7 18ED	JR	F1A6	Tovább!

Ezután már csak a rangsorban a legfelső szintű műveletek maradtak hátra. A felhasználói függvények kiértékelése miatt ez a szubrutin meglehetősen hosszadalmas.

Hibakezelés

F1B9 CF	RST	8	
F1BA 03		hibakód:03	Argument missing

Az ORD, ATN, zárójeles kifejezések és változók feldolgozása.

A két beépített függvény, mint láttuk, nem került be a szimbólumláncba, így ezek kiértékelésére a saját tokenjük alapján itt kerül sor.

F1BB F5	PUSH	AF	Először is ellenőrizzük, hogy
F1BC CD8EFC	CALL	FC8E	van-e elegendő memóriaterület
F1BF F1	POP	AF	
F1C0 FEC0	CP	C0	Az ORD tokenje. Ha ez kell,
F1C2 CAA9ED	JP	Z,EDA9	ugrás az ORD rutin belépési
			pontjára
F1C5 FEB1	CP	B1	Az ATN tokenje. Ha ez kell,
F1C7 CAF1EA	JP	Z,EAF1	ugrás az ATN-hez

F1CA FE02	CP	02	Ha	szöveg	következik:
F1CC DA5DF3	JP	C, F35D	"típushiba"		
F1CF CA43FC	JP	Z, FC43	Ha	szám:	visszatérés a
			következő	utasítás	lehívásával
F1D2 FEFD	CP	FD	Ha	a	forrássorban vége az
F1D4 30E3	JR	NC, F1B9	utasításnak:	hiba	
F1D6 FE96	CP	96	Ha	nem is a "("	tokenje: ugrás
F1D8 2008	JR	NZ, F1E2	előre,	míg ha az:	feldolgozza
F1DA CDA4F0	CALL	FOA4	a	zárójelen	belüli kifejezést
F1DD FE95	CP	95	Ha	ezután ")"	jön, akkor veszi
F1DF CA43FC	JP	Z, FC43	a	következő	utasítást
F1E2 FE03	CP	03	Most	már	csak számváltozó
F1E4 C25AFD	JP	NZ, FD5A	lehet,	különben	hiba!

Számváltozók feldolgozása

F1E7 CD2EF4	CALL	F42E	Ez	a	rutin HL-ben az adat
			szimbólumtáblabeli	címét,	C-
			ben a	típusbyte-ot	adja vissza
F1EA CB59	BIT	3, C	Ha	beépített	függvény,
F1EC C2ADDB	JP	NZ, DBAD	végrehajtja		
F1EF CB51	BIT	2, C	A	számváltozó	értékét a
F1F1 CA63FA	JP	Z, FA63	verembe	tölti	

Itt a DEF-fel meghatározott felhasználói függvények értékének kiszámításával folytatódik a végrehajtás.

DEF függvények kiértékelése

F1F4 DD5601	LD	D, (IX+01)	BASIC	változó,	típusbyte
F1F7 D5	PUSH	DE			
F1F8 E5	PUSH	HL	Az	adat	címe a
			szimbólumtáblában		
F1F9 D696	SUB	96	Egy	jelzést	állítunk elő arra,
F1FB D601	SUB	01	hogy	a	függvény neve után van-
F1FD 9F	SBC	A, A	e	zárójelben	megadatt
F1FE F5	PUSH	AF	paraméter		
F1FF 79	LD	A, C	A	típusbyte	
F200 2814	JR	Z, F216	Ha	a	név után nincs zárójel:
			ugrás	előre	
F202 08	EX	AF, AF	Itt	feldolgozza	a zárójeles
F203 CD43FC	CALL	FC43	kifejezést		

```

F206 B7      OR      A
F207 FA5AFD  JP      M,FD5A
F20A DD7701  LD      (IX+01),A
F20D CD8DF2  CALL   F28D
F210 3E95    LD      A,95
F212 CD54FD  CALL   FD54
F215 09      EX      AF,AF
F216 08      EX      AF,AF
F217 F1      POP    AF
F218 2A2617  LD      HL,(1726)
F21B E3      EX      (SP),HL
F21C E5      PUSH   HL
F21D 2A2417  LD      HL,(1724)
F220 E3      EX      (SP),HL

```

Ha egy új tokent talál:
hiba (not understood)!

Kiértékeli a kifejezést
A ")" tokenje, most csak ezt
fogadja el

```

F221 D9      EXX
F222 E5      PUSH   HL

```

TOP, az első szabad byte címét
a verembe, az utolsó elem alá
tölti

CHAIN, az utolsó szimbólum
címét a verembe, onnan a DEF
szimbólum első adatbyte-jának
címét a HL-be tölti

```

F223 2A0C17  LD      HL,(170C)
F226 E5      PUSH   HL
F227 C5      PUSH   BC
F228 D9      EXX
F229 5E      LD      E,(HL)
F22A 23      INC    HL
F22B 56      LD      D,(HL)
F22C 23      INC    HL
F22D ED530C17 LD      (170C),DE
F231 5E      LD      E,(HL)
F232 23      INC    HL
F233 56      LD      D,(HL)
F234 EB      EX      DE,HL

```

A következő utasítás címe a
forrásorban

```

F223 2A0C17  LD      HL,(170C)
F226 E5      PUSH   HL
F227 C5      PUSH   BC
F228 D9      EXX
F229 5E      LD      E,(HL)
F22A 23      INC    HL
F22B 56      LD      D,(HL)
F22C 23      INC    HL
F22D ED530C17 LD      (170C),DE
F231 5E      LD      E,(HL)
F232 23      INC    HL
F233 56      LD      D,(HL)
F234 EB      EX      DE,HL

```

Az aktuális BASIC sor
kezdőcíme

Ezután a DEF szimbólum első
adatbyte-jait tölti a START
változóba: ez lesz az aktuális
sor, ez tartalmazza a
függvénydefiníciót

```

F235 D9      EXX

```

Majd DE-be, s végül HL-be a
soron belül a függvény neve
utáni első utasítás címét
teszi, így itt folytatódik a
végrehajtás, melyre magát a
BASIC mechanizmust használjuk

```

F236 4F      LD      C,A
F237 CD43FC  CALL   FC43
F23A D696    SUB    96
F23C D601    SUB    01
F23E 9F      SBC    A,A
F23F A9      XOR    C
F240 C25AFD  JP      NZ,FD5A

```

A paraméterre vonatkozó
jelzést áttöltve, a következő
utasítás lehívásával itt is
megvizsgáljuk, hogy van-e a
név után zárójel. Ha a két
jelzés nem azonos, akkor a
függvényhivatkozás hibás!

```

F243 A9      XOR    C
F244 2827    JR      Z,F26D

```

Ha jó, helyreállítjuk a
jelzést
Ha nincs zárójeles paraméter:
ugrás előre

F246	CD43FC	CALL	FC43	A következő utasítás
F249	E681	AND	81	A zárójelek között most csak
F24B	FE01	CP	01	egy szimbólum lehet, különben:
F24D	C25AFD	JP	NZ,FD5A	hiba
F250	D9	EXX		Típusbyte; ha még nincs ilyen
F251	CB79	BIT	7,C	szimbólum, átmenetileg létre-
F253	CC0BF4	CALL	Z,F40B	hozza
F256	79	LD	A,C	Ellenőrzi a definiált és az
F257	DDAE01	XOR	(IX+01)	utasításban szereplő formális
F25A	E602	AND	02	változók típusát. Ha eltérő:
F25C	C25DF3	JP	NZ,F35D	hiba!
F25F	D5	PUSH	DE	Az adatbyte címe
F260	D9	EXX		
F261	E1	POP	HL	A változó értékét a veremből
F262	CD3BFB	CALL	FB3B	bemásolja a szimbólumtáblába
F265	CD43FC	CALL	FC43	Lehívja a következő utasítást,
F268	3E95	LD	A,95	ez csak egy ")" lehet
F26A	CD54FD	CALL	FD54	
F26D	3E9A	LD	A,9A	Utána csak az "=" jelet fogadja
F26F	CD54FD	CALL	FD54	el
F272	08	EX	AF,AF	Ezután a függvény definí-
F273	DD7701	LD	(IX+01),A	ciójában előírt utasításokat
F276	CD8DF2	CALL	F28D	hajtja végre
F279	C1	POP	BC	Végül csak a korábbi
F27A	E1	POP	HL	programparaméterek teljes
F27B	220C17	LD	(170C),HL	visszaállítása maradt hátra. A
F27E	E1	POP	HL	másodlagos regiszterkészletben
F27F	D9	EXX		BC, az aktuális sor címe, ezen
F280	E1	POP	HL	belül az aktuális utasítás
F281	222417	LD	(1724),HL	címe, a szimbólumtábla TOP és
F284	E1	POP	HL	CHAIN változójának visszaál-
F285	222617	LD	(1726),HL	lítása és a 1701H cím eredeti
F288	F1	POP	AF	értéke
F289	320117	LD	(1701),A	
F28C	C9	RET		

Tetszőleges kifejezések feldolgozása

F28D	DDCB014E	BIT	1,(IX+01)	A kifejezés típusa
F291	C2A7F0	JP	NZ,FOA7	Ha szám: ugrás a megfelelő
				belépesi pontra
				Itt a szövegek feldolgozásával
				folytatja

Szövegkifejezések (vezérlés)

F294 D9	EXX	
F295 78	LD A,B	Az aktuális utasításflag
F296 D9	EXX	
F297 CDE0F2	CALL F2E0	Először itt is a zárójeles kifejezéseket és a változókat dolgozza fel
F29A D9	EXX	
F29B 78	LD A,B	
F29C D9	EXX	Utolsóként kerül sorra a konkatenáció
F29D FE97	CP 97	
F29F C0	RET NZ	Ha nem kell: visszatér

Konkatenáció, karakterláncok összefűzése

Ez egyszerűen azt jelenti, hogy az első szöveg után hozzámásoljuk a másodikat, s az eredményt egyetlen karakterláncnak vesszük.

F2A0 CD43FC	CALL FC43	A következő utasítás
F2A3 CDE0F2	CALL F2E0	Elvégzi a magasabb rendű szövegműveleteket
F2A6 FDE5	PUSH IY	
F2A8 E1	POP HL	A BASIC veremben közönséges átmozgatásokat fog végrehajtani
F2A9 23	INC HL	
F2AA 5D	LD E,L	HL és DE a 2. lánc hosszára mutat
F2AB 54	LD D,H	
F2AC 4E	LD C,(HL)	
F2AD 0600	LD B,00	
F2AF 09	ADD HL,BC	A hossz alapján megkeresi az első szöveg azonosító byte-ját
F2B0 23	INC HL	
F2B1 E5	PUSH HL	
F2B2 FDE1	POP IY	
F2B4 23	INC HL	
F2B5 4E	LD C,(HL)	Ezután az első szöveg hosszára áll és memóriaterület ellenőrzés után a verembe elsőként lerakott szöveget a másodikként tárolt szöveg elé másolja. Előtte a második szöveg hosszát A-ba tölti
F2B6 CD8EFC	CALL FC8E	
F2B9 E5	PUSH HL	
F2BA 09	ADD HL,BC	
F2BB 79	LD A,C	
F2BC B7	OR A	
F2BD 1A	LD A,(DE)	
F2BE 2802	JR Z,F2C2	
F2C0 EDB8	LDDR	
F2C2 E1	POP HL	
F2C3 86	ADD A,(HL)	Az összevont karakterlánc hossza a két hossz összege lesz: leteszi
F2C4 12	LD (DE),A	
F2C5 DA12F9	JP C,F912	Ha túl hosszú az összevont szöveg: hiba (overflow)
F2C8 FEFF	CP FF	
F2CA CA12F9	JP Z,F912	
F2CD 4E	LD C,(HL)	Ismét az 1. karakterlánc hossza

F2CE 5D	LD E,L	Ennek címéből DE-vel rááll a
F2CF 54	LD D,H	2. szöveg végére -- amely
F2D0 1B	DEC DE	egyben az összevont
F2D1 1B	DEC DE	karakterlánc vége -- HL-lel
F2D2 09	ADD HL,BC	pedig az 1. szöveg végére. Ide
F2D3 EB	EX DE,HL	kell végül elmozgatni az
F2D4 4F	LD C,A	összevont szöveget
F2D5 03	INC BC	
F2D6 EDB8	LDDR	
F2D8 EB	EX DE,HL	Az így kapott karakterlánc
F2D9 3601	LD (HL),01	elejére írjuk az azonosítót,
F2DB E5	PUSH HL	beállítjuk az új veremhatárt
F2DC FDE1	POP IY	és kész,
F2DE 18BA	JR F29A	mehet tovább!

Stringváltozók és zárójeles kifejezések

F2E0 CD2DF3	CALL F32D	A változók feldolgozása
F2E3 D9	EXX	
F2E4 78	LD A,B	
F2E5 D9	EXX	
F2E6 FE96	CP 96	A "(" zárójel tokenje
F2E8 C0	RET NZ	Ha nincs: kész

Ezután már csak a szöveg-szeletre vonatkozó utasítás végrehajtása maradt hátra.

Szelet képzés

F2E9 CD27FD	CALL FD27	Szeletparaméterek:
F2EC FDE5	PUSH IY	E=kezdő, D=záró karakter
F2EE E1	POP HL	sorszám a láncban
F2EF 23	INC HL	
F2F0 7E	LD A,(HL)	A szöveg teljes hossza
F2F1 4F	LD C,A	
F2F2 0600	LD B,00	
F2F4 BA	CP D	Ha a D-beli záróérték ennél
F2F5 3801	JR C,F2F8	nagyobb, visszaállítja:
F2F7 7A	LD A,D	legfeljebb az utolsó karak-
F2F8 57	LD D,A	terig mehet
F2F9 B7	OR A	Ha D=0, üres lánc lesz:
F2FA 280D	JR Z,F309	ugrás előre
F2FC 79	LD A,C	Az eredeti hossz
F2FD BB	CP E	Ha a kezdőérték nagyobb,
F2FE 3809	JR C,F309	akkor üres string lesz
F300 1C	INC E	A kezdőérték 0 volt?
F301 1D	DEC E	

F302 2001	JR	NZ,F305	Ha kezdőértékként 00-t adtak meg, átállítjuk 1-re, a szeletképzést minimum az első karakterrel kezdjük
F304 1C	INC	E	
F305 7A	LD	A,D	Ha a kezdőérték nem nagyobb, mint a záró, ugrás előre.
F306 93	SUB	E	Egyébként üres szöveg paramétereket állítunk be, s ugrás a végére
F307 3004	JR	NC,F30D	
F309 09	ADD	HL,BC	
F30A AF	XOR	A	
F30B 180B	JR	F318	
F30D 3C	INC	A	A szelet hossza +1
F30E E5	PUSH	HL	Az eredeti szöveg eleje, majd a vége
F30F 09	ADD	HL,BC	Ide fogjuk a szeletet is másolni (DE=célcím)
F310 EB	EX	DE,HL	A szelet záróértéke és az ennek megfelelő karakter címe (HL=másolási kezdőcím). Végül a hossz alapján a szeletet átmozgatjuk
F311 4C	LD	C,H	
F312 E1	POP	HL	
F313 09	ADD	HL,BC	
F314 4F	LD	C,A	
F315 EDB8	LDDR		
F317 EB	EX	DE,HL	
F318 77	LD	(HL),A	Elé beírjuk a szelet hosszát és az elemazonosítóbyte-ot, majd beállítjuk a verem új határát és kész,
F319 2B	DEC	HL	
F31A 3601	LD	(HL),01	
F31C E5	PUSH	HL	
F31D FDE1	POP	IY	
F31F 18C2	JR	F2E3	mehet tovább

Segédrutin a szeletképzéshez

Ezt a pársoros rutint a programozók kétszer is benne hagyták a ROM-ban. Egyszer itt, egyszer a 0FD45H címen. A rendszer itt sohasem hívja, szeletképzésnél a másik rutin az aktív.

F321 CDC4FA	CALL	FAC4	Kiszámítja a megadott kifejezést és értékét a HL-be konvertálja
F324 1600	LD	D,00	
F326 24	INC	H	
F327 C8	RET	Z	
F328 15	DEC	D	Ha H nem 00, akkor D=FF lesz, H=FF esetén D=00, míg H=00 esetén D=L-lel tér vissza a rutin
F329 25	DEC	H	
F32A C0	RET	NZ	
F32B 55	LD	D,L	
F32C C9	RET		

Karakterláncok feldolgozása (vezérlés)

F32D CD8EFC	CALL FC8E	Ellenőrzi, hogy van-e hely a
F330 D9	EXX	memóriában
F331 C5	PUSH BC	Utasításflag és típusbyte
F332 D9	EXX	
F333 C1	POP BC	
F334 78	LD A,B	
F335 FE01	CP 01	Ha szövegkonstans, mehet
F337 DA43FC	JP C,FC43	tovább!
F33A 280B	JR Z,F347	Stringváltozó esetén ugrás
		előre!
F33C FE04	CP 04	Ha szám, vagy számszimbólum:
F33E 381D	JR C,F35D	"típushiba"!
F340 FEFD	CP FD	Ha utasítás vége:
F342 DA5AFD	JP C,FD5A	hiba (not understood)!

Hibakezelés

F345 CF	RST 8	
F346 03	hibakód:03	Argument missing

Stringváltozó feldolgozása

F347 79	LD A,C	Típusbyte
F348 FEC5	CP C5	Ha INKEY\$ token: ugrás a
F34A CA89EC	JP Z,EC89	végrehajtó rutinra
F34D CD2EF4	CALL F42E	A szimbólumtáblabeli adat
F350 CB59	BIT 3,C	címét HL-be teszi, s ha
F352 C2ADDB	JP NZ,DBAD	beépített függvény, azt
		hajtja végre
F355 CB51	BIT 2,C	Ha a szimbólum egy DEF
F357 C2F4F1	JP NZ,F1F4	utasítást azonosít, ezt a
		függvényt értékeli ki
F35A C37CFA	JP FA7C	Az egyszerű változóhoz
		tartozó stringet a BASIC
		verembe másolja. A
		visszatérés e rutinok lezáró
		RET-jére történik

Hibakezelés

F35D CF	RST 8	
F35E 03	hibakód:0E	Type mismatch

Ezzel, kedves Olvasó, a kifejezéseket kiértékelő rutinok végére értünk. Látjuk, hogy a számok közötti műveletek alacsony szintű rutinjaira még nem került sor, ezzel a jelen fejezet egy további részében foglalkozunk majd.

A szimbólumtáblát kezelő rutinok

A következő programrészek a BASIC verem és a láncolt szimbólumok kapcsolatát biztosítják, ill. ezzel kapcsolatos adminisztratív munkát látnak el. Általában egyszerűbbek, mint pl. az előző rész rutinjai, közülük talán a tömbök kezelése a legfigyelemreméltóbb.

Új szimbólum láncolása

Belépéskor a 1728H címtől a forrássor következő utasításának címe van tárolva. A rutin bejegyzi a szimbólum nevét, elvégzi a megfelelő adminisztrációt, s visszatéréskor HL a név utáni típusbyte címét tartalmazza.

F35F	CD8EFC	CALL	FC8E	Ellenőrzi, hogy van-e elég memóriaterület
F362	ED5B2417	LD	DE, (1724)	CHAIN
F366	2A2617	LD	HL, (1726)	TOP
F369	222417	LD	(1724), HL	Először elvégzi a láncolást
F36C	73	LD	(HL), E	azzal, hogy CHAIN-ba az
F36D	23	INC	HL	utolsó elem címeként az
F36E	72	LD	(HL), D	eddig TOP-értéket írja,
F36F	23	INC	HL	valamint az új szimbólum
				elejére bejegyzni az előző
				címét
F370	0E00	LD	C, 00	A névhossz mérése
F372	E5	PUSH	HL	A hosszbyte címe
F373	ED5B2817	LD	DE, (1728)	A forrássorból olvassuk a név
F377	1A	LD	A, (DE)	karaktereit
F378	FEFD	CP	FD	Ha utasítás végét talál:
F37A	3027	JR	NC, F3A3	kilép a ciklusból
F37C	FE20	CP	20	A 20H alá konvertált
F37E	381C	JR	C, F39C	karaktereket elfogadja (l. a
				3.6.7.2 szakaszt)
F380	FE24	CP	24	A "\$" jelet elfogadja --
F382	2819	JR	Z, F39D	lezáró karakterként
F384	FE2E	CP	2E	A névben "." karakter is
F386	2814	JR	Z, F39C	lehet!
F388	FE30	CP	30	20H és 2FH között azonban más
F38A	3817	JR	C, F3A3	már nem lehet!
F38C	FE3A	CP	3A	
F38E	380C	JR	C, F39C	Számjegyeket elfogad
F390	FE3F	CP	3F	3A és 3EH közötti karakter
F392	380F	JR	C, F3A3	viszont nem lehet!
F394	FE60	CP	60	Sőt ezeken kívül a szimbólum
F396	3804	JR	C, F39C	nevében már csak a 3FH...5FH
F398	FEA9	CP	A9	kódú
F39A	3807	JR	C, F3A3	karakterek
				szerepelhetnek

F39C 37	SCF		"Folytatni kell" jelzés és a
F39D 13	INC	DE	név hossz növelése után
F39E 0C	INC	C	mindkét címmel továbblépünk a
F39F 23	INC	HL	memóriában, letesszük az új
F3A0 77	LD	(HL),A	karaktert, s ha lehet,
F3A1 38D4	JR	C,F377	folytatjuk a névbeolvasást
F3A3 7E	LD	A,(HL)	Ha kész, a záró karaktert A-
F3A4 23	INC	HL	ba, a következő címet a TOP-
F3A5 222617	LD	(1726),HL	ba töltjük (típusbyte), a név
F3A8 E3	EX	(SP),HL	elé pedig beírjuk a hosszát
F3A9 71	LD	(HL),C	és kész
F3AA E1	POP	HL	
F3AB C9	RET		

Területfelszabadítás új számszimbólumnak

F3AC CD8EFC	CALL	FC8E	Ellenőrzi, hogy van-e még
F3AF 2A2617	LD	HL,(1726)	hely a memóriában
F3B2 0606	LD	B,06	
F3B4 AF	XOR	A	Majd a TOP-ban tárolt címtől
F3B5 77	LD	(HL),A	kezdve 0-val tölt fel 6 byte-
F3B6 23	INC	HL	ot
F3B7 10FC	DJNZ	F3B5	Visszatérés a TOP beállítá-
F3B9 1813	JR	F3CE	sával a következő rutin végén

Területfelszabadítás új szimbólumoknak, vezérlés

F3BB DDCB014E	BIT	1,(IX+01)	Ha szám-szimbólum, ugrás az
F3BF 20EB	JR	NZ,F3AC	előző rutinra, szöveg esetén
			itt folytatja

Területfelszabadítás új szövegszimbólumnak

F3C1 CD8EFC	CALL	FC8E	A memória ellenőrzése után az
F3C4 2A2617	LD	HL,(1726)	első adatbyte TOP-ban tárolt
F3C7 71	LD	(HL),C	címére a maximális hosszt
F3C8 23	INC	HL	tölti, a következő címre,
F3C9 0600	LD	B,00	aktuális hosszként pedig most
F3CB 70	LD	(HL),B	0-t. Rááll a szöveg végét
F3CC 09	ADD	HL,BC	követő címre
F3CD 23	INC	HL	
F3CE 222617	LD	(1726),HL	Végül beállítja a TOP értékét
F3D1 C9	RET		és kész

Szimbólumkeresés

A szubrutin feladatát a megnevezés mutatja. Hívásakor HL az aktuális BASIC sorban egy szimbólum nevének első karakterére mutat, s meg kell vizsgálni, hogy ez a név szerepel-e már a szimbólumtáblában.

Ha a megadott szimbólum már létezik, a rutin a DE regiszterpárban visszaadja az első adatbyte címét, a C regiszterben a típusbyte-ot és CY=1-et állít be. Sikertelen keresés után pedig DE=0000, Z=1 és CY=0 lesz.

F3D2 222817	LD	(1728),HL	Az aktuális utasítás címe.
F3D5 212517	LD	HL,1725	Először a CHAIN
F3D8 3EE1	LD	A,E1	rendszerváltozóból, majd
(de: F3D9 E1	POP	HL)	ismételt végrehajtáskor a
F3DA 56	LD	D,(HL)	vizsgált szimbólum első 2
F3DB 2B	DEC	HL	byte-jából DE-be tölti a
F3DC 5E	LD	E,(HL)	következő szimbólum címét
F3DD 7B	LD	A,E	
F3DE B2	OR	D	Ha a cím 0000, akkor a lánc
F3DF 2A2817	LD	HL,(1728)	végére ért, a keresés
F3E2 C8	RET	Z	sikertelen!
F3E3 EB	EX	DE,HL	
F3E4 23	INC	HL	A címet további kereséshez
F3E5 E5	PUSH	HL	tárolja, majd a név
F3E6 23	INC	HL	azonosításához a vizsgált
F3E7 46	LD	B,(HL)	szimbólum névhosszával ciklust
F3E8 1A	LD	A,(DE)	szervez
F3E9 13	INC	DE	
F3EA 23	INC	HL	A karakterek összehasonlítása.
F3EB BE	CP	(HL)	Ha bárhol eltérés van, áttér a
F3EC 20EB	JR	NZ,F3D9	következő szimbólumra
F3EE 10F8	DJNZ	F3E8	
F3F0 FE24	CP	24	Külön vizsgálja a végét
F3F2 2810	JR	Z,F404	"\$" esetén rendben van, ez
F3F4 1A	LD	A,(DE)	egyértelmű lezárást jelent a
F3F5 FE20	CP	20	név után
F3F7 280B	JR	Z,F404	
F3F9 FEFD	CP	FD	Ezen kívül csak a "szóközt", az
F3FB 3007	JR	NC,F404	"utasítás vége", valamint a
F3FD FEA9	CP	A9	80...A8H kódú karaktereket
F3FF 30D8	JR	NC,F3D9	fogadja el
F401 17	RLA		
F402 30D5	JR	NC,F3D9	
F404 F1	POP	AF	A cím már nem kell
F405 23	INC	HL	A megtalált szimbólumban
F406 CBBE	RES	7,(HL)	"régi" jelzést állít be
F408 37	SCF		Sikeres keresés jelzése
F409 181D	JR	F428	Ugrás a befejezéshez, a lezáró
			utasítások a következő
			rutinéval közösek

Új szimbólum létrehozása

A rutin teljes munkát végez. Elvégzi a láncolást, a BASIC utasításor HL mutatta címéről a szimbólumtáblába másolja az új nevet, kitölti a típusbyte-ot és ennek megfelelő területet foglal le a memóriában. Visszatéréskor HL már a következő utasításra mutat, DE az első adatbyte címét, a C regiszter pedig a típusbyte-ot tartalmazza.

F40B CD5FF3	CALL F35F	Láncolás és névbejegyzés
F40E D624	SUB 24	A "\$" kódja
F410 2802	JR Z,F414	Ha szöveg: A=00, ha nem: A=02
F412 3E02	LD A,02	-- számszimbólum
F414 77	LD (HL),A	Beírja a típusbyte-ot
F415 E5	PUSH HL	
F416 23	INC HL	Szöveg esetén a következő cím
F417 222617	LD (1726),HL	a TOP új értéke
F41A 2006	JR NZ,F422	Ha szám: ugrás előre
F41C 0E12	LD C,12	Szövegnek lefoglal 12H byte-ot
F41E CDC1F3	CALL F3C1	a memóriában,
F421 AF	XOR A	
F422 C4ACF3	CALL NZ,F3AC	számok esetén pedig 06 byte-
		ot, ez a rutin a TOP változót
F425 E1	POP HL	is állítja
F426 CBFE	SET 7,(HL)	A típusbyte-ban új szimbólum
		jelzése
F428 4E	LD C,(HL)	Betölti a típusbyte-ot, majd
F429 23	INC HL	HL és DE megfelelő értékeivel
F42A EB	EX DE,HL	visszatér
F42B C9	RET	

Hibakezelés

F42C CF	RST 8	
F42D 05	hibakód:05	Bad subscript

Az adat címe a szimbólumtáblában

F42E D9	EXX	
F42F D5	PUSH DE	Az első adatbyte címe
F430 CB41	BIT 0,C	Tömb?
F432 79	LD A,C	A típusbyte-ot átmenti
F433 D9	EXX	
F434 E1	POP HL	
F435 4F	LD C,A	Ha nem tömbről van szó:
F436 CA43FC	JP Z,FC43	kész, mehet tovább!

Jelentősebb munkát jelent a tömbelemek elérése. A program ezzel folytatódik.

Tömbelem keresése

```

F439 3E96      LD  A,96
F43B CD51FD    CALL FD51
F43E AF        XOR  A
F43F 5F        LD  E,A
F440 57        LD  D,A

F441 7E        LD  A,(HL)
F442 23        INC HL
F443 EB        EX  DE,HL
F444 C5        PUSH BC
F445 E5        PUSH HL
F446 F5        PUSH AF
F447 3EA4      LD  A,A4
F449 C454FD    CALL NZ,FD54
F44C CDC4FA    CALL FAC4
F44F EB        EX  DE,HL
F450 4E        LD  C,(HL)
F451 23        INC HL
F452 46        LD  B,(HL)
F453 23        INC HL
F454 EB        EX  DE,HL
F455 B7        OR  A
F456 ED42      SBC HL,BC
F458 30D2      JR  NC,F42C
F45A 09        ADD HL,BC
F45B F1        POP AF
F45C F5        PUSH AF
F45D D5        PUSH DE
F45E 3D        DEC A
F45F 280E      JR  Z,F46F
F461 EB        EX  DE,HL
F462 4E        LD  C,(HL)
F463 23        INC HL
F464 46        LD  B,(HL)
F465 23        INC HL
F466 E5        PUSH HL
F467 F5        PUSH AF
F468 CDB3FC    CALL FCB3
F46B F1        POP AF
F46C D1        POP DE
F46D 18EF      JR  F45E
F46F D1        POP DE
F470 F1        POP AF
F471 C1        POP BC
F472 09        ADD HL,BC
F473 3D        DEC A
F474 20CF      JR  NZ,F445

F476 3E95      LD  A,95
F478 CD54FD    CALL FD54

```

A "("-t fogadva el, lehívja a következő utasítást. DE=0000, ez lesz a 2 byte-os számláló kezdőértéke

Az első adatbyte a dimenziószám, utána az indexek maximális értékei 2-2 byte-on A számláló és a dimenziószám Az "," után további indexeket kell hívni A kért indexet HL-be tölti, onnan DE-be, így betöltheti a szimbólumtáblából annak maximális értékét. HL a következő index címére áll

Ha a kért index nagyobb, mint a maximális, akkor hiba!

Ismét a dimenziószám

A következő index címe Ha végigért, kilép az alábbi számolási ciklusból: a vizsgált aktuális és a még utána álló összes index maximális értékét össze-szorozva jutunk el a kért indexhez tartozó elemekhez. Ehhez ismételten meghívjuk a HL = BC *DE funkciójú rutint

A számolás végén a kapott elemszámot hozzáadjuk a számláló korábbi értékéhez, s így járunk el addig, amíg a tömb végére nem érünk

A ")" tokenje. Végül már csak ezt fogadja el

F47B EB	EX	DE,HL	DE=a kért elem sorszáma a tömbben
F47C C1	POP	BC	C-ben a típusbyte
F47D CB49	BIT	1,C	Szám? -- ha igen: a tömb
F47F 010600	LD	BC,0006	elemeinek hossza: 06, szövegek
F482 2003	JR	NZ,F487	esetén pedig a hosszt az
F484 4E	LD	C,(HL)	indexek utáni byte mutatja a
F485 03	INC	BC	szimbólumtáblában
F486 03	INC	BC	
F487 E5	PUSH	HL	Az első elem kezdőcíme
F488 CDB3FC	CALL	FCB3	HL=DC*DE a kért elem
F48B D1	POP	DE	távolsága, s végül ezekből
F48C 19	ADD	HL,DE	az elem memóriacíme
F48D C9	RET		

Alapműveletek lebegőpontos számokkal

Ismét egy meglehetősen bonyolult, de igen nagy jelentőségű és izgalmas ponthoz érkeztünk. Az itt következő rutinok hajtják végre a tényleges számolási feladatokat a BASIC veremben korábban elhelyezett számokkal.

A rutinok értelmezése során, a rövid és egyértelmű leírás érdekében, gyakran fogjuk használni az alábbi egyszerű jelöléseket:

a, ai	a BASIC veremben elsőként elhelyezett (nagyobb memóriacímmű) szám, ill. annak i-edik jegye
b, bi	az utolsóként verembe töltött (kisebb memóriacímmű) szám, ill. annak i-edik jegye
e1, e2	a számok 40H-val megnövelt kitevője (tehát ahogyan a memóriában szerepel), de a számok előjelét mutató 7. bit nélkül
exp1, exp2	a számok teljes exponense (tehát az előbbi), de az előjelbittel együtt
t1, t1	a két számhoz tartozó, ún. túlcscordulásbyte (az exponens előtti memóriacímen)

Kivonás

F48E F5	PUSH	AF	Az exp2-ben az előjelbitet
F48F CD26F7	CALL	F726	megfordítja, ezután a művelet
F492 F1	POP	AF	összeadással helyettesíthető.

A program azzal folytatódik

Összeadás

F493 F5	PUSH AF	
F494 FDE5	PUSH IY	y azonosítójának címe
F496 110900	LD DE,0009	x azonosítójának címe
F499 FD19	ADD IY,DE	IY ismét y-ra mutat
F49B FDE3	EX (SP),IY	exp2
F49D FD7E08	LD A,(IY+08)	e2
F4A0 E67F	AND 7F	Az e2-t tárolja C-ben
F4A2 4F	LD C,A	exp1
F4A3 FD7E11	LD A,(IY+11)	e1
F4A6 E67F	AND 7F	e1-e2
F4A8 91	SUB C	Ha nem negatív: ugrás
F4A9 F2BFF4	JP P,F4BF	Most IY az x-re mutat
F4AC FDE3	EX (SP),IY	e1-e2
F4AE F5	PUSH AF	Ezzel a "nagyságrendek"
F4AF ED44	NEG	különbsége pozitív szám
F4B1 CD90F7	CALL F790	A nagyságrendek kiegyenlítése: a jegyek eltolása
F4B4 C1	POP BC	B-ben e1-e2 (negatív)
F4B5 FDE3	EX (SP),IY	IY ismét y-ra mutat
F4B7 FD7E11	LD A,(IY+11)	exp1
F4BA 90	SUB B	exp1-(e1-e2), így: e1=e2
F4BB FD7711	LD (IY+11),A	lett (a 7. bit: maradt)
F4BE AF	XOR A	Átlépi a következő sort
F4BF C490F7	CALL NZ,F790	A másik esetben is elvégzi a
F4C2 FD7E08	LD A,(IY+08)	nagyságrendek kiegyenlítését
F4C5 FDAE11	XOR (IY+11)	(F4A9)!
F4C8 07	RLCA	Azonos előjelű számok?
F4C9 D4F4F4	CALL NC,F4F4	Ha igen: összeadás
F4CC FDE3	EX (SP),IY	IY x-re mutat, s ha azonos
F4CE 301E	JR NC,F4EE	előjelűek voltak, ugrás a
F4D0 CD1CF7	CALL F71C	befejezéshez
F4D3 FDE3	EX (SP),IY	A negatív számot megszorozza
F4D5 CD1CF7	CALL F71C	(-1)-gyel, akár mindkettőt, s
F4D8 CDF4F4	CALL F4F4	utána
F4DB FDE3	EX (SP),IY	elvégzi az összeadást
F4DD FD7E07	LD A,(IY+07)	IY x-re mutat
F4E0 B7	OR A	t
F4E1 280B	JR Z,F4EE	Ha nincs túlcscordulás,
F4E3 CD0CF7	CALL F70C	ugrás előre
F4E6 3E80	LD A,80	Az n jegyeit (-1)-szeresének
F4E8 FDAE08	XOR (IY+08)	megfelelően írja át és az
F4EB FD7708	LD (IY+08),A	előjelbitet is megfordítja
F4EE F1	POP AF	
F4EF CD34F7	CALL F734	Normalizálás után kész
F4F2 F1	POP AF	
F4F3 C9	RET	

F4F4 F5	PUSH AF	A jelzőbitekét tárolja
F4F5 FDE5	PUSH IY	IY az y-ra mutat,
F4F7 D1	POP DE	DE szintén,
F4F8 210900	LD HL,0009	
F4FB 19	ADD HL,DE	HL pedig az x-re
F4FC 0607	LD B,07	Az összeadást ciklusban 7
F4FE 23	INC HL	byte-tal, tehát t1 és t2-vel is
F4FF 13	INC DE	elvégezzük
F500 1A	LD A,(DE)	Az y egy jegye,
F501 8E	ADC A,(HL)	+ x egy jegye + átvitel
F502 27	DAA	és decimális "igazítás"
F503 77	LD (HL),A	Az eredményt az x helyén
F504 10F8	DJNZ F4FE	tároljuk
F506 F1	POP AF	A végén visszatölti a
F507 C9	RET	jelzőbitekét és kész

Kilépési pont a szorzásból, ha az eredmény: n=0

F508 110900	LD DE,0009	
F50B FD19	ADD IY,DE	A veremben 0 számot tesz az
F50D CDF9F9	CALL F9F9	eredmény helyére
F510 F1	POP AF	
F511 C9	RET	

Szorzás

F512 F5	PUSH AF	
F513 FD7E0F	LD A,(IY+0F)	a1,2
F516 B7	OR A	a=0?
F517 28EF	JR Z,F508	Ha igen: n=0 és kész!
F519 FD7E06	LD A,(IY+06)	b1,2
F51C B7	OR A	b=0?
F51D 28E9	JR Z,F508	Ha igen: n=0 és kész!
F51F FD7E11	LD A,(IY+11)	exp1
F522 FD6E08	LD L,(IY+08)	exp2
F525 AD	XOR L	
F526 87	LD H,A	A vizsgálat eredménye
F527 AD	XOR L	A-ban ismét: exp1
F528 E67F	AND 7F	
F52A 6F	LD L,A	L-ben és A-ban: e1
F52B FD7E08	LD A,(IY+08)	exp2
F52E E67F	AND 7F	e2
F530 85	ADD A,L	e1+e2

F531 D640	SUB 40	A-ban a helyes, most már csak 40H-val növelt kitevő
F533 38D3	JR C,F508	Ha túl kicsi: n=0 lesz,
F535 FA12F9	JP M,F912	ha túl nagy: hiba
F538 3C	INC A	
F539 FA12F9	JP M,F912	Sőt még e=7FH sem lehet!
F53C 3D	DEC A	
F53D 6F	LD L,A	
F53E E5	PUSH HL	
F53F D9	EXX	Kezdődik a számolás!
F540 E3	EX (SP),HL	HL'-be az előbbi HL-t tölti,
F541 D5	PUSH DE	majd az összes regisztert
F542 C5	PUSH BC	tárolja
F543 E5	PUSH HL	
F544 FDE5	PUSH IY	
F546 E1	POP HL	HL is y-ra mutat,
F547 5D	LD E,L	
F548 54	LD D,H	DE szintén,
F549 011000	LD BC,0010	
F54C 09	ADD HL,BC	majd HL-ben t1 címe
F54D CDE8F5	CALL F5E8	Az x-t jegyekre bontja
F550 D5	PUSH DE	A DE a szétbontott x elé mutat
F551 77	LD (HL),A	
F552 2B	DEC HL	A=00, így törli az x
F553 77	LD (HL),A	azonosítóját és exponensét
F554 2B	DEC HL	
F555 CDE8F5	CALL F5E8	Az y jegyeire bontása, a memóriában x elé

E műveletek eredményeként az eredeti x és y törlődött, helyettük, az általuk korábban lefoglalt memóriaterületek elé az x és y egy-egy jegyét tartalmazó byte-ok kerültek, mégpedig az x jegyei vannak a nagyobb memóriacímen.

Ezután lényegileg úgy történik a jegyek szorzása, mint amikor ezt papíron, írásban végezzük. Az egyetlen, ám nagyon lényeges eltérés az, hogy az egymás alá kerülő rész-szorzatokat most nem csak a legvégén adjuk össze, hanem egy-egy szorzás elvégzése után azonnal, s a memóriában mindig az addig elvégzett műveletek eredményét tároljuk.

Rögtön látni fogjuk, hogy a nagy pontosságú műveletvégzés érdekében a szorzást 16 helyiértéken végezzük. A részösszegeket és a végeredményt is az y és x eredeti helyén képezzük a memóriában. A jegyek szorzásához a ROM elején, a 0C003H...0C066H címeken található táblázatot használjuk fel (51. old). Az eredményben az első két helyi értéken a két túlcsonduló számjegyet, utána pedig 14 értékes számjegyet tárolunk. A 13. és 14. jegyet persze

nem fogjuk tárolni, ezek a kerekítéshez, a még pontosabb számoláshoz kellene.

F558 210A00	LD HL,000A	A 16-jegyű eredményhez a
F55B 19	ADD HL,DE	szorzást csak y3-mal kezdjük
F55C 4D	LD C,L	(y12 helyett)
F55D 44	LD B,H	Az y3-ra mutat BC
F55E 111300	LD DE,0013	HL'-t az eredmény utolsó (16.)
F561 19	ADD HL,DE	jegyére állítjuk, DE' pedig az
F562 D1	POP DE	első szorzás előtt x12-re
F563 13	INC DE	mutat
F564 E5	PUSH HL	Az A-ban y jegyeit követjük:
F565 F602	OR O2	A+1 az y aktuális jegyének
F567 08	EX AF,AF	sorszámára
F568 D9	EXX	A'-ben számolunk, C pedig a
F569 0E0C	LD C,0C	ciklusváltozó
F56B D9	EXX	
F56C E1	POP HL	HL' n16-ra mutat: az eredmény
F56D E5	PUSH HL	legkisebb helyiértékű jegye
F56E C5	PUSH BC	Az y aktuális kezdő jegyének
		címe
F56F 1A	LD A,(DE)	A'=xi, az aktuális szorzó
F570 D9	EXX	
F571 08	EX AF,AF	A+1=az y aktuális jegyének
F572 F5	PUSH AF	sorszámára
F573 08	EX AF,AF	
F574 87	ADD A,A	A szorzótábla használatához az
F575 2839	JR Z,F5B0	A'-ben xi-t megszorozzuk 10-
F577 57	LD D,A	zel (ez adja az eltolást az xi-
F578 87	ADD A,A	nek megfelelő sorhoz), s ezt D-
F579 87	ADD A,A	ben tároljuk. Ha a szorzó 0:
F57A 82	ADD A,D	ugrás előre
F57B 57	LD D,A	
F57C 26C0	LD H,C0	Rááll a szorzótáblára
F57E 1E00	LD E,00	E-ben lesznek az átvitelek
F580 0610	LD B,10	
F582 08	EX AF,AF	Ha még nem ért el y1-ig,
F583 F28EF5	JP P,F58E	folytatja, míg
F586 1C	INC E	az y1-gyel való szorzást is
F587 1D	DEC E	elvégezve, már csak az esetleg
F588 2826	JR Z,F5B0	fennmaradt átviteleket kell a
F58A 08	EX AF,AF	korábbi összeghez hozzáadni
F58B AF	XOR A	
F58C 180D	JR F59B	
F58E FE0C	CP 0C	A 13. és 14. jegynél pedig a
F590 30F8	JR NC,F58A	szorzat eleve 0
F592 08	EX AF,AF	Az y jegyének sorszámát ismét
		tároljuk

```

F593 D9      EXX
F594 0A      LD  A,(BC )
F595 D9      EXX

F596 82      ADD  A,D
F597 C603    ADD  A,03
F599 6F      LD  L,A
F59A 7E      LD  A,(HL)

F59B 83      ADD  A,E
F59C 27      DAA
F59D D9      EXX
F59E 86      ADD  A,(HL)
F59F 27      DAA
F5A0 3600    LD  (HL),00
F5A2 ED6F    RLD
F5A4 23      INC  HL
F5A5 03      INC  BC
F5A6 D9      EXX
F5A7 07      RLCA
F5A8 07      RLCA
F5A9 07      RLCA
F5AA 07      RLCA
F5AB 5F      LD  E,A
F5AC 08      EX  AF,AF
F5AD 3D      DEC  A
F5AE 10D3    DJNZ F583

F5B0 F1      POP  AF
F5B1 3C      INC  A
F5B2 08      EX  AF,AF
F5B3 0D      DEC  C
F5B4 D9      EXX
F5B5 C1      POP  BC
F5B6 0B      DEC  BC
F5B7 13      INC  DE
F5B8 20B2    JR  NZ,F56C

F5BA E1      POP  HL
F5BB 010F00  LD  BC,000F
F5BE 09      ADD  HL,BC
F5BF 5D      LD  E,L
F5C0 54      LD  D,H
F5C1 CDDAF5  CALL F5DA
F5C4 E5      PUSH HL
F5C5 FDE1    POP  IY
F5C7 E1      POP  HL
F5C8 7C      LD  A,H
F5C9 E680    AND  80
F5CB E5      OR   L
F5CC FD7709  LD  (IY+08),A

```

A'-be pedig betöltjük az y aktuális jegyét (járulékos eltolás a szorzótábla kijelölt sorában)

Végül a kapott eltolások és a 0C003H kezdőcím alapján a táblázatból kiolvasható a két jegy szorzata, amihez még hozzájön a korábbi átvitel. A decimális jegykorrekció után a kapott eredmény hozzáadható a korábban nyert részösszeghez, s az új érték már tárolható

Utána a következő helyi értékre léptetjük a címmutatókat (y és n jegyeire) Végül a keletkezett átvitelt az A' alsó 4 bitjébe forgatjuk,

áttöltjük E-be, az A-ban y jegymutatóját továbbléptetjük, s a szorzás mehet tovább

Ha xi-vel y minden jegyét megszoroztuk, akkor x, y, és az eredmény mutatóit az aktuálisan következő kezdőértékekre állítjuk, a ciklusváltozót csökkentjük, s ha még nincs teljesen kész, folytatjuk a szorzást

Ha a végére értünk, akkor az eredmény legnagyobb helyiértékű jegyére állunk és visszaállítjuk a szokásos, tömörített tárolási formát (2-jegy byte-onként)

Az új BASIC veremhatárt most már IY-ba töltve,

beállítjuk az eredmény előjelét,

F5CF	FD360009	LD	(IY+00),09	a szám elé írjuk a "09"
F5D3	C1	POP	BC	azonosítót, visszatöltjük a
F5D4	D1	POP	DE	regiszterek eredeti értékét, s
F5D5	E1	POP	HL	a feladatot a normalizálás
F5D6	D9	EXX		rutinjával fejezzük be
F5D7	C3EFF4	JP	F4EF	

Kétségtelen, hogy egy meglehetősen bonyolult struktúrájú, a CPU szinte valamennyi regiszterét igénybevevő, de mindenképp nagyon tömör utasítássorozatot jártunk át. Igazi profi munka, amellyel nagyjából azonos súlyú lesz az osztás szubrutinja. Ezek a programozás igazi gyöngyszemei, tanulmányozásuk fölöttébb hasznos, a komoly, nagyszabású feladatokat kedvelőknek pedig különös csemegét jelentenek.

Előtte azonban át kell még futnunk két rövid és egyszerű szubrutinon.

Segédrutin a szorzáshoz: BCD tömörítés

F5DA	0607	LD	B,07	
F5DC	1A	LD	A,(DE)	Az egyesével 14 (dec.) byte-
F5DD	1B	DEC	DE	on tárolt számjegyeket
F5DE	ED6F	RLD		(ezekre mutat DE) kettesével
F5E0	1A	LD	A,(DE)	1-1 byte-ba tömöríti (az új
F5E1	1B	DEC	DE	tárolási címet tartalmazza
F5E2	ED6F	RLD		HL)
F5E4	2B	DEC	HL	
F5E5	10F5	DJNZ	F5DC	
F5E7	C9	RET		

Segédrutin a szorzáshoz

A tömörített BCD kódú számot felbontja különálló jegyeire és azokat byte-onként tárolja a memóriában

F5E8	0607	LD	B,07	7 byte-on kell a bontást
F5EA	AF	XOR	A	elvégezni
F5EB	ED6F	RLD		A HL mutatta címről először a
F5ED	12	LD	(DE),A	nagyobb helyi értékű jegyet
F5EE	1B	DEC	DE	forgatjuk be az A-ba és
F5EF	AF	XOR	A	tároljuk a DE szerinti
F5F0	ED6F	RLD		memóriacímre.
F5F2	12	LD	(DE),A	Majd ismételt eljárással a
F5F3	1B	DEC	DE	kisebb című memóriarekeszbe
F5F4	2B	DEC	HL	az alacsonyabb helyi értékű
F5F5	10F3	DJNZ	F5EA	jegy kerül
F5F7	AF	XOR	A	
F5F8	C9	RET		

Hibakezelés

F5F9 CF	RST 8	Cannot divide by 0 (nem lehet
F5FA 0B	hibakód:0B	0-val osztani)

Osztás

Ezt a műveletet is elvileg ugyanúgy végezzük, mint amikor többjegyű számokat papíron osztunk. Akkor azonban az eredmény egy-egy jegyét "fejben" tippeljük meg, amit itt nagy biztonsággal azzal helyettesítünk, hogy az osztandóból ismételt művelettel annyiszor vonjuk ki az osztót, amennyiszer lehet és ezt a számot tároljuk. Az osztás többi része lényegileg ugyanaz: az előbbi kivonások után kapott maradékot egy helyi értékkel léptetjük (szorozzuk tízzel), s a következő osztást már ezzel végezzük. Az eljárást addig folytatjuk, amíg a hányados jegyeinek számával a szükséges pontosságot (itt 12 jegy) el nem érjük.

Az osztás programja is több érdekes segédrutint használ. Ezeket a fejezet egy későbbi részében ismerjük meg.

F5FB F5	PUSH AF	
F5FC FD7E06	LD A, (IY+06)	
F5FF B7	OR A	Ha az osztó (y) nulla,
F600 28F7	JR Z, F5F9	hibajelzéssel visszatér
F602 FD7E0F	LD A, (IY+0F)	Ha az osztandó (x) értéke 0,
F605 B7	OR A	akkor az eredmény is az, csak
F606 CA1AFA	JF Z, FA1A	IY-t kell visszaállítani
F609 D9	EXX	
F60A E5	PUSH HL	Kezdődik a munka: először itt
F60B D5	PUSH DE	is mentjük a regisztereket
F60C C5	PUSH BC	
F60D FDE5	PUSH IY	Majd IY-t az osztandóra, x-re
F60F 110900	LD DE, 0009	állítjuk és betöltjük HL-be az
F612 FD19	ADD IY, DE	
F614 FD6EFF	LD L, (IY+FF)	exp2 és
F617 FD6608	LD H, (IY+08)	exp1 exponenseket
F61A E3	EX (SP), HL	Ezeket és az x-re mutató címet
F61B 221C17	LD (171C), HL	a verembe tesszük, az eredeti
F61E E5	PUSH HL	IY-t pedig a 171CH-ra is. A
F61F FDE3	EX (SP), IY	veremben az y előtti helyen
F621 CDF1F9	CALL F9F1	kapjuk az eredményt (n), a
		helyét most nullázzuk, s ezzel
		n1-re állunk
F624 0604	LD B, 04	Beállítjuk az alábbi
F626 2A1C17	LD HL, (171C)	mutatókat:
F629 D9	EXX	

```

F62A 01F7FF LD BC,FFF7
F62D 2A1C17 LD HL,(171C)
F630 5D LD E,L
F631 54 LD D,H
F632 B7 OR A
F633 ED42 SBC HL,BC
F635 EB EX DE,HL
F636 09 ADD HL,BC
F637 09 ADD HL,BC
F638 0607 LD B,07
F63A B7 OR A
F63B 13 INC DE
F63C 1A LD A,(DE)
F63D D9 EXX
F63E 23 INC HL
F63F 9E SBC A,(HL)
F640 D9 EXX
F641 27 DAA
F642 23 INC HL
F643 77 LD (HL),A
F644 10F5 DJNZ F63B

```

-- HL' az osztó (y) jegyeit címzi;
 -- DE az osztandóra mutat (eredetileg x);
 -- HL pedig egy -- a memóriában az eredmény előtti számra, itt tároljuk majd a kivonási maradékokat

Elvégezzük a 7-jegyű kivonást: az osztandó jegyeiből sorra kivonjuk az osztó megegyező helyiértékű jegyeit, s az eredményt a szükséges decimális igazítás után a memória külön helyén (a hányados előtt) tároljuk

```

F646 E6F0 AND F0
F648 2013 JR NZ,F65D
F64A D9 EXX
F64B CDD2F7 CALL F7D2
F64E 3C INC A
F64F CDE3F7 CALL F7E3
F652 2A1C17 LD HL,(171C)
F655 D9 EXX
F656 010700 LD BC,0007
F659 EDB8 LDDR
F65B 18DB JR F638

```

Ha már túlcsondulás is lett: a kivonás kész
 Ha még nincs túlcsondulás: a kivonásokat az eredmény helyén, a kijelölt helyi értéken számláljuk, visszaállítjuk a címmutatókat, a maradékot az osztandó helyére másoljuk és folytatjuk a kivonást

```

F65D FDE3 EX (SP),IY
F65F CD84F7 CALL F784
F662 FDE3 EX (SP),IY
F664 D9 EXX
F665 04 INC B
F666 78 LD A,B
F667 FE10 CP 10
F669 38BB JR C,F626

```

Amikor egy kivonás kész, a maradékot megszorozzuk 10-zel, az eredményben ráállunk a következő helyi értékre és ha még nincs 12 (dec.) jegy, folytatjuk az eljárást (az aktuális jegy sorszáma: B-3)

```

F66B D9 EXX
F66C 2A1C17 LD HL,(171C)
F66F 2B DEC HL
F670 2B DEC HL
F671 010800 LD BC,0008
F674 EDB8 LDDR
F676 FDE1 POP IY
F678 E1 POP HL

```

Ha a 12 jegy megvan, akkor felmásoljuk az eredeti osztó (x) helyére, (hiszen itt kell visszaadni az eredményt)

F679 7C	LD	A,H	A H-ban és L-ben tárolt eredeti
F67A AD	XOR	L	exponensek alapján, a 7. bitek
F67B E680	AND	80	összehasonlításával képezzük
F67D 4F	LD	C,A	az n előjelét. Utána az eredeti
F67E CBBC	RES	7,H	előjeleket törölve kapjuk e1-
F680 CBBD	RES	7,L	et és e2-t, ezekből pedig az e
F682 7C	LD	A,H	= e1 - e2 + 40H művelettel,
F683 95	SUB	L	majd az előjelbit beírásával
F684 C640	ADD	A,40	kapjuk az eredmény exponensét
F686 A9	XOR	C	
F687 FD7708	LD	(IY+08),A	
F68A A9	XOR	C	Ha e kisebb 80H-nál, a
F68B F2D3F5	JP	P,F5D3	nagyságrend jó, visszatérés a
			normalizálás rutinján át
F68E CDF9F9	CALL	F9F9	Ha nagyobb: az eredmény 0 lesz
F691 18F8	JR	F68B	(ami bizony hiba, hiszen
			helyesen "Overflow" jelzés
			kellene!)

A tömör és professzionális kidolgozásra mindenesetre itt sem panaszkodhatunk, a fenti utasítássorozat jól átgondolt, szép munka.

A segédrutinok

A ROM következő részében az előbbi műveleti rutinokból és természetesen más helyekről is hívott néhány, különböző terjedelmű és bonyolultságú szubrutin található. Ezek jellegüknél fogva már kevesebb magyarázatot igényelnek.

Két szám összehasonlítása

Ha x és y jelöli a BASIC veremben elsőként és másodikként elhelyezett számokat, akkor az x-y kivonás logikának megfelelően:

```
-- ha x kisebb, mint y      CY=1,  S=1  és  Z=0;
-- ha x és y egyenlők      CY=0,  S=0  és  Z=1;
-- ha x nagyobb, mint y    CY=0,  S=0  és  Z=0;
```

```

F693 FDE5    PUSH IY
F695 E1      POP  HL
F696 010900 LD   BC,0009
F699 09      ADD  HL,BC
F69A 5D      LD   E,L
F69B 54      LD   D,H
F69C 1B      DEC  DE
F69D 09      ADD  HL,BC
F69E E5      PUSH HL
F69F FDE1    POP  IY

```

Az IY az y azonosítóját címzi,

a HL ezzel x-re mutat,

a DE pedig exp2-re

Utána HL-t, majd IY-t az x elé állítva, a számokat érvénytelenítjük

```

F6A1 2B      DEC  HL
F6A2 7E      LD   A,(HL)
F6A3 E680    AND  80
F6A5 0F      RRCA
F6A6 4F      LD   C,A
F6A7 1A      LD   A,(DE)
F6A8 E680    AND  80
F6AA 0F      RRCA
F6AB 91      SUB  C
F6AC C0      RET  NZ

```

Az exp1-et betöltve, nézzük az x előjelét

Az x előjelbitjét tároljuk C-ben

Ugyanígy dolgozzuk fel y előjelét, majd ezeket kivonva, eltérő előjelek esetén már készen is vagyunk

```

F6AD 81      ADD  A,C
F6AE 2801    JR   Z,F6B1
F6B0 EB      EX  DE,HL
F6B1 1A      LD   A,(DE)
F6B2 1B      DEC  DE
F6B3 E67F    AND  7F
F6B5 4F      LD   C,A
F6B6 7E      LD   A,(HL)
F6B7 2B      DEC  HL
F6B8 E67F    AND  7F
F6BA 91      SUB  C
F6BB C0      RET  NZ

```

Azonos előjelű számok esetén, ha negatívak, akkor először felcseréljük a címmutatókat, hiszen a nagyobb abszolút értékű negatív szám a kisebb. Ezután az előjelektől megszabadulva, az előbbi séma szerint összehasonlítjuk a két kitevőt

Ha különbözők: kész

```

F6BC 0607    LD   B,07
F6BE 1A      LD   A,(DE)
F6BF E6F0    AND  F0
F6C1 0F      RRCA
F6C2 4F      LD   C,A
F6C3 7E      LD   A,(HL)
F6C4 E6F0    AND  F0
F6C6 0F      RRCA
F6C7 91      SUB  C
F6C8 C0      RET  NZ
F6C9 1A      LD   A,(DE)
F6CA E60F    AND  0F
F6CC 4F      LD   C,A
F6CD 7E      LD   A,(HL)
F6CE E60F    AND  0F
F6D0 91      SUB  C
F6D1 C0      RET  NZ

```

Végül a 7 byte-on tárolt 14 (dec.) jegyet kell külön-külön összehasonlítani

Ehhez minden byte-ot kétszer kell nézni, különválasztva először a nagyobb, majd kisebb helyi értékű számjegyet.

Az A-ban elvégzett kivonások rögtön beállítják a jelzőbiteket is, tehát ha valamelyik két jegy eltérő, akkor azonnal meg is szakíthatók a további összehasonlítások

F6D2 1B	DEC DE	Ha pedig a két szám minden
F6D3 2B	DEC HL	jegye is egyenlő, akkor ennek
F6D4 10E8	DJNZ F6BE	jelzésével történik a vissza-
F6D6 C9	RET	térés

A szubrutin végén a két szám változatlanul a BASIC veremben marad, tehát még felhasználhatók. Mivel azonban IY-t az x elé állítottuk, így a legközelebbi beírási műveletnél már x, de esetleg y is átíródik.

A következő szubrutin a fenti mintára, két string összehasonlítását végzi el.

Két karakterlánc összehasonlítása

Az eredményt a CY, S és Z jelzőbitek mutatják, pontosan úgy, mint ha számokat hasonlítanánk össze.

F6D7 FDE5	PUSH IY	
F6D9 E1	POP HL	
F6DA 23	INC HL	A második szöveg hosszbyte-
F6DB E5	PUSH HL	jára mutat
F6DC 4E	LD C, (HL)	A második szöveg hossza
F6DD 0600	LD B, 00	
F6DF 09	ADD HL, BC	
F6E0 23	INC HL	
F6E1 23	INC HL	Az első szöveg hosszára mutat
F6E2 E5	PUSH HL	
F6E3 7E	LD A, (HL)	Az első szöveg hossza
F6E4 B9	CP C	
F6E5 3801	JR C, F6E8	Ezután A-ba a rövidebb, C-be az
F6E7 79	LD A, C	1. szöveg hossza kerül
F6E8 4E	LD C, (HL)	
F6E9 09	ADD HL, BC	Ennek segítségével IY-t a
F6EA 23	INC HL	szövegek elé állíthatjuk, így
F6EB E5	PUSH HL	azokat érvénytelenítjük
F6EC FDE1	POP IY	
F6EE 4F	LD C, A	Az összehasonlítás a rövidebb
F6EF 03	INC BC	szöveg szerinti hosszon
F6F0 D1	POP DE	történik
F6F1 E1	POP HL	A hosszakkal kezdjük és addig
F6F2 1A	LD A, (DE)	folymatjuk, amíg eltérő byte-
F6F3 13	INC DE	okat nem találunk, vagy a
F6F4 EDA1	CPI	rövidebb szöveg végére nem
F6F6 E2FBF6	JP PO, F6FB	érünk
F6F9 28F7	JR Z, F6F2	

F6FB 2B	DEC HL	
F6FC 4E	LD C, (HL)	Végül a jelzőbitek beállítása
F6FD 6F	LD L, A	végett, kivanással összehason-
F6FE 60	LD H, B	lítjuk az utolsóként vizsgált
F6FF ED42	SBC HL, BC	karaktereket, s az A-ban még
F701 7C	LD A, H	külön jelzést is adva (0,
F702 C8	RET Z	negatív szám vagy 1),
F703 F8	RET M	visszatérés
F704 3E01	LD A, 01	
F706 C9	RET	

A két szöveg az összehasonlítás után változatlanul megvan a BASIC veremben, ám az IY veremhatár mutató visszaállítása miatt a legközelebbi veremírási művelet már elrontja. A két szöveg jelenléte a továbbiakban már érvénytelen.

Ezután egyszerű és rövid segédrutinok következnek.

Negatív szám jegyeinek szorzása (-1)-gyel

A funkció megnevezése úgy értendő, hogy az adott szám jegyeit 0-ból kivonjuk -- az átvitelek megfelelő figyelembevételével és természetesen azt ezt követően szükségessé váló BCD kódú jegykorrekcióval. A kivonás a túlcscordulásbyte-ot is felhasználva, 14 helyi értéken történik. Ezért -- eredetileg nem 0 számok esetén -- a túlcscordulásbyte tartalma mindig 99H lesz.

F707 FDCB087E	BIT 7, (IY+08)	Ha az előjel pozitív, nem
F70B C8	RET Z	csinálunk semmit
F70C FDE5	PUSH IY	
F70E E1	POP HL	
F70F 110700	LD DE, 0007	A D szolgáltatja a kivonáshoz
F712 B7	OR A	a 0-t, E pedig a ciklusszámláló
F713 23	INC HL	
F714 7A	LD A, D	
F715 9E	SBC A, (HL)	Kivonás 0-ból, majd decimális
F716 27	DAA	igazítás után az eredmény
F717 77	LD (HL), A	tárolása
F718 1D	DEC E	
F719 20F8	JR NZ, F713	Ismétlés 7 byte-on át
F71B C9	RET	

Negatív szám jegyeinek szorzása (-1)-gyel, a szám előjelének átállításával

F71C FDCB087E	BIT 7, (IY+08)	Ha a szám pozitív, nem
F720 C8	RET Z	csinálunk semmit

```

F721 C00CF7    CALL F70C
F724 1805      JR    F72B

```

A jegyek (-1)-szeresét vesszük, majd visszatérés az előjelbit átállításával

Nem 0 szám előjelét megfordítja

```

F726 FD7E06    LD    A,(IY+06)
F729 B7        OR    A
F72A C8        RET   Z
F72B 3E80      LD    A,80
F72D FDAE08    XOR   (IY+08)
F730 FD7708    LD    (IY+08),A
F733 C9        RET

```

Ha a szám értéke 0: nem csinálunk semmit

Egyébként az előjelbitet megfordítjuk

Számok normalizálása

```

F734 C0DFF9    CALL F9DF
F737 C8        RET   Z

F738 FD7E07    LD    A,(IY+07)
F73B B7        OR    A
F73C 2818      JR    Z,F751
F73E 3E01      LD    A,01
F740 CD90F7    CALL F790
F743 FD7E08    LD    A,(IY+08)
F746 E67F      AND   7F
F748 FE7E      CP    7E
F74A CA12F9    JP    Z,F912
F74D FD3408    INC  (IY+08)
F750 C9        RET

```

Ha minden számjegy 0, akkor az exponenst is nullázza és kész

Ha a túlcscordulásbyte üres: ugrás előre
Ha korábban egy túlcscordulást okozó műveletvégzés volt, akkor a jegyek 1-szeri balraléptetésével osztunk 10-zel és 1-gyel növeljük az exponenst. (Ha így túl nagy szám lenne: Overflow!)

```

F751 FD7E06    LD    A,(IY+06)
F754 E6F0      AND   F0
F756 C0        RET   NZ
F757 CD84F7    CALL F784
F75A FD7E08    LD    A,(IY+08)
F75D FD3508    DEC  (IY+08)
F760 E67F      AND   7F
F762 20ED      JR    NZ,F751

F764 C3F9F9    JP    F9F9

```

Ha korábban nem volt túlcscordulás és az első jegy nem 0: kész

A jegyeket jobbra léptetve szorzunk 10-zel, ezzel együtt az exponenst csökkentjük

Ismét megnézzük az első jegyet

Ha közben az exponens 0 lett, akkor az egész érték is 0

Az utolsó két jegy kerekítése

```

F767 E5      PUSH HL
F768 FDE5    PUSH IY
F76A E1      POP HL
F76B 23      INC HL
F76C 7E      LD A,(HL)
F76D FE50    CP 50
F76F 3600    LD (HL),00
F771 380F    JR C,F782
F773 0606    LD B,06
F775 23      INC HL
F776 7E      LD A,(HL)
F777 C601    ADD A,01
F779 27      DAA
F77A 77      LD (HL),A
F77B 3002    JR NC,F77F
F77D 10F6    DJNZ F775
F77F CD34F7  CALL F734
F782 E1      POP HL
F783 09      RET

```

Az utolsó két jegyet tartalmazó byte-ot nézzük. Ha 50H-nál nem nagyobb, akkor helyére 00 kerül és kész

Ha az utolsó két jegyet tartalmazó byte 50H-nál nagyobb, akkor a helyére ugyan 00 kerül, de az előtte álló jegyet megnöveljük 1-gyel. Ha átvitel van, akkor a többi jegy is változik

Végül normalizálás és kész

Eltolás egy jeggyel jobbra (szorzás 10-zel)

```

F784 FDE5    PUSH IY
F786 E1      POP HL
F787 0607    LD B,07
F789 AF      XOR A
F78A 23      INC HL
F78B ED6F    RLD
F78D 10FB    DJNZ F78A
F78F 09      RET

```

A szám utolsó (12.) jegye 0 lesz, a többi eggyel magasabb helyi értékre, az eredeti első jegy pedig a túlcsoordulásbyte-ba lép

Két szám nagyságrendjének kiegyenlítése

Belépéskor az A tartalmazza a nagyságrendben megkívánt korrekciót. A rutin ennek megfelelően a számjegyeket ismételt balra léptetve (osztás 10-zel) állítja be a tárolt szám új alakját. Az exponenst nem változtatja meg.

```

F790 FE0E    CP 0E
F792 3030    JR NC,F7D4
F794 01FF00  LD BC,00FF
F797 FDE5    PUSH IY
F799 E1      POP HL
F79A E5      PUSH HL

```

Ha a kért korrekció 14 (dec.) nagyságrendnél is nagyobb, az eredmény 0

C-ben a léptetések számát tartjuk nyilván

F79B 5D	LD	E,L	Először a byte-os léptetések
F79C 54	LD	D,H	érdekében, amíg lehet 2-esével
F79D 23	INC	HL	változtatjuk a jegyek helyi
F79E D602	SUB	O2	értékét
F7A0 0C	INC	C	
F7A1 30FA	JR	NC,F79D	Ha C=00 (mert pl. A=01 volt),
F7A3 2812	JR	Z,F7B7	mehet előre
F7A5 13	INC	DE	
F7A6 F5	PUSH	AF	Ha C nem 0, akkor lehetőség van
F7A7 3E07	LD	A,07	a jegyek byte-os eltolására,
F7A9 91	SUB	C	ezt
F7AA 4F	LD	C,A	
F7AB EDB0	LDIR		végrehajtjuk,
F7AD EB	EX	DE,HL	
F7AE D608	SUB	O8	majd az eltolás miatt
F7B0 2F	CPL		érvénytelenné vált korábbi
F7B1 70	LD	(HL),B	számjegyek helyére 00-t írunk
F7B2 23	INC	HL	
F7B3 3D	DEC	A	
F7B4 20FB	JR	NZ,F7B1	
F7B6 F1	POP	AF	
F7B7 E1	POP	HL	Ha a szükséges nagyságrend
F7B8 3C	INC	A	korrekció páros szám volt:
F7B9 C0	RET	NZ	kész
F7BA 0E07	LD	C,07	Páratlan számú korrekció
F7BC 09	ADD	HL,BC	esetén pedig a még fennmaradt
F7BD 41	LD	B,C	egy helyiértékkel való
F7BE ED67	RRD		eltolást végezzük el a
F7C0 2B	DEC	HL	memóriában
F7C1 10FB	DJNZ	F7BE	
F7C3 C9	RET		
F7C4 FD4E08	LD	C,(IY+08)	13-nál nagyobb (dec.)
F7C7 CDF9F9	CALL	F9F9	nagyságrend kiegyenlítés ese-
F7CA FD7108	LD	(IY+08),C	tén az exponenst változatlanul
F7CD C9	RET		hagyva, a számjegyeket
			nullázzuk

A következő két utasítást a rendszer nem használja, a fejlesztés során maradhatott a ROM-ban:

```

F7CE 21B6FE    LD    HL,FEB6
F7D1 E5      PUSH HL

```

Segédrutin: egy számjegy beolvasása a BASIC veremből

Ez a szubrutin az IY által címzett szám (B)-3 sorszámú jegyét tölti az A-ba, míg az A-ban hozott értéket a C regiszterbe menti. (B): a B regiszter értéke.

F7D2 E5	PUSH HL	HL-ben előállítjuk a kívánt
F7D3 C5	PUSH BC	jegy memóriacímét és CY=1, ha
F7D4 CDFDF7	CALL F7FD	(B) páratlan volt. Az adott
F7D7 3805	JR C,F7DE	címen tárolt két jegy közül
F7D9 7E	LD A,(HL)	annak megfelelően választjuk a
F7DA 07	RLCA	nagyobb vagy kisebb helyi
F7DB 07	RLCA	értékűt, hogy (B) értéke
F7DC 07	RLCA	páratlan vagy páros
F7DD 07	RLCA	
F7DE E60F	AND OF	
F7E0 C1	POP BC	
F7E1 E1	POP HL	
F7E2 C9	RET	

Segédrutin: egy számjegy átírása a BASIC veremben

Belépéskor A tartalmazza a számjegy új értékét, B-ben pedig a jegy sorszámát adjuk meg, 3-mal növelve, tehát: sorszám=(B)-3 alakban.

F7E3 E5	PUSH HL	HL-be a kijelölt jegy
F7E4 C5	PUSH BC	memóriacíme kerül; CY=1, ha
F7E5 CDFDF7	CALL F7FD	(B) páratlan, A-ban a régi, C-
F7E8 06F0	LD B,F0	ben az új érték
F7EA 380B	JR C,F7F7	
F7EC CB21	SLA C	Ezek alapján az új értéket C-
F7EE CB21	SLA C	ben szükség esetén a megfelelő
F7F0 CB21	SLA C	4 bitre mozgatva, az adott
F7F2 CB21	SLA C	memóriacíme beírjuk a számjegy
F7F4 7E	LD A,(HL)	új értékét
F7F5 060F	LD B,OF	
F7F7 A0	AND B	
F7F8 B1	OR C	
F7F9 77	LD (HL),A	
F7FA C1	POP BC	
F7FB E1	POP HL	
F7FC C9	RET	

Végül az itt tárgyalt utolsó szubrutin az előbbi kettőt szolgálja ki. A HL regiszterpárban adja a B-vel meghatározott számjegy memóriacímét, A-ban az adott byte-on

tárolt mindkét számjegyet, CY=1 beállítással jelzi, ha B-ben páratlan szám volt, végül C-ben megőrzi az A eredeti értékét. Mindehhez képest a rutin meglehetősen rövid!

F7FD	FD E5	PUSH	IY	
F7FF	E1	POP	HL	
F800	CB38	SRL	B	B-ben most már a jegyet
F802	F5	PUSH	AF	tartalmazó byte-hoz való
F803	3E08	LD	A,08	eltolás van
F805	90	SUB	B	Ezt 8-ból kivonva, már
F806	4F	LD	C,A	közvetlenül kaphatjuk a kívánt
F807	0600	LD	B,00	számjegy címét a
F809	09	ADD	HL,BC	memóriában
F80A	F1	POP	AF	
F80B	4F	LD	C,A	A-t áttöltjük C-be, majd
F80C	7E	LD	A,(HL)	beolvassuk az adott címen
F80D	C9	RET		tárolt számjegyeket és ezzel
				kész!

A ROM ezután következő részében túlnyomórészt BASIC verem és más pufferek közötti adatmozgatást megvalósító szubrutinok kaptak helyet. Itt találhatóak ugyanakkor egyéb konverziós rutinok is, amelyeknek ismerete felhasználói szempontból nagyon hasznos lehet.

A BASIC verem és más pufferek kapcsolata

A ROM-nak ez a területe eléggé hosszú szubrutinokkal kezdődik, amelyek azonban logikailag nem túlságosan bonyolultak. Méretük nem annyira a funkció sokféleségéből, hanem elsősorban a vizsgálandó feltételek nagy számából adódik.

Konverziós rutin: a veremben levő számot ASCII kódú számlánccá alakítva a 1930H kezdőcímű pufferbe tölti.

F80E	FD7E06	LD	A,(IY+06)	Ha a szám értéke 0, akkor a
F811	B7	OR	A	1931H címre leteszi a 0 ASCII
F812	213119	LD	HL,1931	kódját, s máris ugrás a rutin
F815	3630	LD	(HL),30	végére
F817	287E	JR	Z,F897	Pozitív szám esetén mehet
F819	FDCB087E	BIT	7,(IY+08)	tovább, negatív számok elé
F81D	2806	JR	Z,F825	vizsont kirakja a "-" előjelet
F81F	362D	LD	(HL),2D	(2DH)
F821	23	INC	HL	

F822 CD26F7	CALL F726	A veremben ezután pozitív értékként szerepel
F825 CD67F7	CALL F767	Kerekítés
F828 060E	LD B,0E	Ezután a 10. (dec) jegytől visszafelé haladva megkeressük az első nem 0 számjegyet (sorszám=B-3), aminek sorszáma egyben a kiírandó értékes jegyek száma (C=04...0DH)
F82A 05	DEC B	
F82B 78	LD A,B	
F82C FE04	CP 04	
F82E D4D2F7	CALL NC,F7D2	
F831 28F7	JR Z,F82A	
F833 48	LD C,B	
F834 0604	LD B,04	
F836 FD5E08	LD E,(IY+08)	
F839 7B	LD A,E	
F83A D63B	SUB 3B	
F83C 57	LD D,A	
F83D 7B	LD A,E	
F83E FE4A	CP 4A	
F840 3016	JR NC,F858	
F842 FE40	CP 40	e=40...49H esetén egyszerű kiírás lesz
F844 300A	JR NC,F850	
F846 91	SUB C	Törtek esetén az értékes jegyeket figyelembevéve, a nagyságrend szerint lesz normálalak vagy egyszerű kiírás. Ekkor: B=FB...04 (-5...+4 dec.)
F847 3810	JR C,F859	
F849 FE32	CP 32	
F84B 380C	JR C,F859	
F84D 42	LD B,D	
F84E 180B	JR F85B	
F850 79	LD A,C	Az e=40...49H esetben D=05...0EH, az értékes jegyek száma szerint kell vagy sem a szám végére 0-kat írni Egyszerű kiírás lesz
F851 BA	CP D	
F852 3007	JR NC,F85B	
F854 4A	LD C,D	
F855 B7	OR A	
F856 1804	JR F85C	
F858 37	SCF	Normálalak jelzése. Ekkor a tizedespont előtt egy számjegy lesz
F859 1605	LD D,05	
F85B 0C	INC C	Ezután a B-beli érték segítségével sorra vesszük a számjegyeket Amikor kell, lerakjuk a "." tizedespont kódját, s ha a B már a verembeli szám egy jegyét címzi,
F85C F5	PUSH AF	
F85D 78	LD A,B	
F85E BA	CP D	
F85F 2003	JR NZ,F864	
F861 362E	LD (HL),2E	
F863 23	INC HL	
F864 78	LD A,B	
F865 D604	SUB 04	


```

F867 3E00      LD   A,00
F869 F4D2F7    CALL P,F7D2
F86C C630      ADD  A,30
F86E 77        LD   (HL),A
F86F 23        INC  HL
F870 04        INC  B
F871 78        LD   A,B
F872 B9        CP   C
F873 20E9     JR   NZ,F85E
F875 F1        POP  AF
F876 3020     JR   NC,F898

F878 3645     LD   (HL),45
F87A 23        INC  HL
F87B 362B     LD   (HL),2B
F87D 7B        LD   A,E
F87E D640     SUB  40
F880 3004     JR   NC,F886
F882 362D     LD   (HL),2D
F884 ED44     NEG
F886 23        INC  HL

F887 06FF     LD   B,FF
F889 04        INC  B
F88A 4F        LD   C,A
F88B D60A     SUB  0A
F88D 30FA     JR   NC,F889
F88F 78        LD   A,B
F890 C630     ADD  A,30
F892 77        LD   (HL),A
F893 23        INC  HL
F894 90        SUB  B
F895 81        ADD  A,C
F896 77        LD   (HL),A

F897 23        INC  HL
F898 113019   LD   DE,1930
F89B 37        SCF
F89C ED52     SBC  HL,DE
F89E EB        EX  DE,HL
F89F 73        LD   (HL),E
F8A0 C9        RET

```

akkor azt töltjük be, egyébként A-ban 0 marad
Képezzük a számjegy kódját és letesszük a puffer aktuális címére
Vesszük a következő jegyet, s ha még van további értékes jegy, folytatjuk.
Ha nem normálalakban kell kiírni, kész is

Az "E" kódját tároljuk, majd utána egy "+" jelet,
ha azonban az exponens 40H-nál kisebb, akkor az előbbi címre a "-" előjel kódját írjuk, s a kitevő abszolút értékét vesszük

Ezután a kitevőt ismételten 10-zel csökkentve, s a kivonásokat számolva előállítjuk és átkódolás után tárol először a kitevő 10-es helyi értékű számjegyét, majd a maradékból az egyes helyi értékűt

Végül az első puffercímből kivonással megállapítjuk a képzett számlánc hosszát, ezt a lánc előtt, a 1930H címen tároljuk és kész

A HL kezdőcímű memóriaterületről szöveg konverziója szabványos szövegelemként a BASIC verembe (pl. a COMMAND, parancsbeviteli pufferből)

```

F8A1 D5        PUSH DE
F8A2 C5        PUSH BC
F8A3 1EFF     LD   E,FF

```

A szöveghossz számolása

F8A5 0E00	LD	C,00	A kezdő " jel figyelése
F8A7 7E	LD	A, (HL)	
F8A8 23	INC	HL	Megkeressük az első nem szóköz karaktert
F8A9 FE20	CP	20	
F8AB 28FA	JR	Z,F8A7	
F8AD FE22	CP	22	
F8AF 2003	JR	NZ,F8B4	
F8B1 0C	INC	C	Ha ", megjegyezzük
F8B2 7E	LD	A, (HL)	A következő karakter
F8B3 23	INC	HL	
F8B4 1C	INC	E	Számoljuk
F8B5 0C	INC	C	
F8B6 0D	DEC	C	Ha nem idézőjellel kezdődött,
F8B7 280E	JR	Z,F8C7	ugrás előre
F8B9 FEFF	CP	FF	Ha vége a sornak, befejezzük a
F8BB 2816	JR	Z,F8D3	vizsgálatot,
F8BD FE22	CP	22	új " jel esetén pedig
F8BF 20F1	JR	NZ,F8B2	megnézzük, hogy egymás után 2
F8C1 BE	CP	(HL)	" jel van-e. Ha igen:
F8C2 23	INC	HL	folytatjuk, egy " pedig a
F8C3 28ED	JR	Z,F8B2	szöveg végét jelenti
F8C5 180C	JR	F8D3	
F8C7 FE2C	CP	2C	Ha nem volt kezdő idéző jel,
F8C9 2808	JR	Z,F8D3	akkor egy vessző, vagy "!"
F8CB FE21	CP	21	jelig, ezek híján az utasítást
F8CD 2804	JR	Z,F8D3	lezáró tokenig folytatjuk a
F8CF FEFD	CP	FD	leltározást
F8D1 38DF	JR	C,F8B2	
F8D3 2B	DEC	HL	Amikor ennek vége, akkor az
F8D4 E5	PUSH	HL	utolsó karakter címét és a
F8D5 D5	PUSH	DE	hosszt tároljuk
F8D6 43	LD	B,E	A BASIC verembe való
F8D7 FDE5	PUSH	IY	átmozgatást a hossz szerinti
F8D9 D1	POP	DE	ciklusban végezzük el, ha van
F8DA CD8EFC	CALL	FC8E	elég memória
F8DD 04	INC	B	
F8DE 05	DEC	B	Csökkentjük a számlálót,
F8DF 1B	DEC	DE	vesszük a következő
F8E0 2816	JR	Z,F8F8	veremcímét, s a pufferből --
F8E2 2B	DEC	HL	visszafelé haladva -- a
F8E3 7E	LD	A, (HL)	következő karaktert
F8E4 FE22	CP	22	Az idézőjeleket külön meg kell
F8E6 2006	JR	NZ,F8EE	vizsgálni! Ha a szöveg nem "
F8E8 0C	INC	C	jellel kezdődött, akkor ezt is
F8E9 0D	DEC	C	tároljuk, ha viszont igen,
F8EA 2802	JR	Z,F8EE	akkor a dupla idézőjelek közül
F8EC 2B	DEC	HL	csak egyet, míg az elsőt és
F8ED 7E	LD	A, (HL)	utolsót nem
F8EE 12	LD	(DE),A	

F8EF FE20	CP	20	A parancsbeolvasáskor 20H alá
F8F1 30EB	JR	NC, F8DE	konvertált kódú karaktereket
F8F3 F680	OR	80	helyreállítjuk
F8F5 12	LD	(DE), A	
F8F6 18E6	JR	F8DE	
F8F8 EB	EX	DE, HL	Amikor az átmásolás befeje-
F8F9 D1	POP	DE	ződött, a szöveg elé beírjuk
F8FA 73	LD	(HL), E	még a hosszát, az elé pedig a
F8FB 2B	DEC	HL	szöveget azonosító byte-ot
F8FC 3601	LD	(HL), 01	Végül IY-t az új veremhatárra
F8FE E5	PUSH	HL	állítjuk,
F8FF FDE1	POP	IY	Visszatöltjük az elmentett
F901 E1	POP	HL	regisztereket és kész
F902 C1	POP	BC	
F903 D1	POP	DE	
F904 C9	RET		

A HL kezdőcímmű pufferből stringet vagy számot tesz a BASIC verembe (vezérlés)

F905 7E	LD	A, (HL)	Mégnézzük az első karaktert:
F906 FE22	CP	22	ha idézőjel, szöveggént dol-
F908 2897	JR	Z, F8A1	gozzuk fel
F90A CD14F9	CALL	F914	Különben számként, s ha ez
F90D D25AFD	JP	NC, FD5A	sikertelen: "Not understood"
F910 17	RLA		hiba. Ha pedig túl nagy:
F911 D0	RET	NC	"Overflow"

Hibakezelés

F912 CF	RST	8	
F913 0D		hibakód:0DH	Overflow

ASCII kódú számlánc konverziója

Belépéskor HL a számláncot tartalmazó puffer kezdőcíme mutat. A rutin szabványos lebegőpontos számmá alakítja és betölti a BASIC verembe.

F914 C5	PUSH	BC	
F915 D5	PUSH	DE	
F916 CDF4F9	CALL	F9F4	Először 0-át töltünk a verembe
F919 E5	PUSH	HL	
F91A FDE5	PUSH	IY	
F91C E1	POP	HL	
F91D 110700	LD	DE, 0007	A túlcsordulásbyte címére
F920 19	ADD	HL, DE	állunk

F921 E3	EX	(SP),HL	Azt tároljuk, s helyette HL-be a puffercím kerül
F922 1E3F	LD	E,3F	Az exponens kezdőértéke 11 (dec.) jegyig nézzük
F924 060B	LD	B,0B	
F926 7E	LD	A,(HL)	
F927 23	INC	HL	Megkeressük az első nem szóköz karaktert
F928 FE20	CP	20	
F92A 28FA	JR	Z,F926	
F92C FE2B	CP	ZB	Ha ez "+" jel, mehet tovább,
F92E 2806	JR	Z,F936	a "-" előjelet azonban D-ben külön megjegyezzük
F930 FE2D	CP	2D	
F932 2004	JR	NZ,F938	
F934 CBCA	SET	1,D	
F936 7E	LD	A,(HL)	Nézzük a következő karaktert
F937 23	INC	HL	
F938 FE2E	CP	2E	Ha tizedespontot találunk: ugrás előre
F93A 282D	JR	Z,F969	
F93C D63A	SUB	3A	Ha nem számjegy: ugrás az exponens vizsgálatához, ahol már csak az "E" kódját fogadjuk el
F93E 3031	JR	NC,F971	A számjegyet tároljuk és beállítjuk a "szám" jelzést, majd a "." bitjét a CY-ba mozgatjuk
F940 C60A	ADD	A,0A	Ha a számjegy 0, akkor: ha volt már értékes jegy is vagy ".", tároljuk, különben nem. Ha volt tizedespont, akkor 0 beolvasásakor csökkentjük az exponenst
F942 302D	JR	NC,F971	
F944 4F	LD	C,A	
F945 CBFA	SET	7,D	
F947 7A	LD	A,D	
F948 0F	RRCA		
F949 2009	JR	NZ,F954	
F94B CB52	BIT	2,D	
F94D 2007	JR	NZ,F956	
F94F 30E5	JR	NC,F936	
F951 1D	DEC	E	
F952 18E2	JR	F936	
F954 CBD2	SET	2,D	Nem 0 számjegy esetén "értékes jegy" jelzés, s ha nem volt tizedespont, növeljük az exponenst
F956 3801	JR	C,F959	
F958 1C	INC	E	A jegyek száma
F959 78	LD	A,B	A 11. jegy után beolvasott szám elveszik!
F95A 3D	DEC	A	
F95B 28D9	JR	Z,F936	
F95D 47	LD	B,A	Addig azonban betöltjük a BASIC verembe. Az aktuális címet attól függően léptetjük tovább, hogy páros vagy páratlan sorszámú beolvasás volt-e
F95E E3	EX	(SP),HL	A cím tárolása után folytatjuk a beolvasást
F95F 0F	RRCA		
F960 3801	JR	C,F963	
F962 2B	DEC	HL	
F963 79	LD	A,C	
F964 ED6F	RLD		
F966 E3	EX	(SP),HL	
F967 18CD	JR	F936	

F969 CBFA SET 7,D
 F96B CB42 BIT 0,D
 F96D CBC2 SET 0,D
 F96F 28C5 JR Z,F936

A tizedespont feldolgozása:
 beállítjuk a "szám lehet" és
 "tizedespont" jelzést, s ha
 még ez az első: mehet tovább

F971 E3 EX (SP),HL
 F972 05 DEC B
 F973 CB40 BIT 0,B
 F975 2803 JR Z,F97A
 F977 AF XOR A
 F978 ED6F RLD
 F97A E1 POP HL
 F97B 2B DEC HL
 F97C 7E LD A,(HL)
 F97D CDC8FB CALL FBC8
 F980 FE45 CP 45
 F982 7B LD A,E
 F983 2040 JR NZ,F9C5

A beolvasás kész

Ha páratlan sorszám
 fejeződött be, akkor a szám
 végére még egy 0 számjegyet
 írunk

Ezután nézzük az utolsó
 beolvasott karaktert
 Kódkonverzió után csak E
 kódját fogadjuk el
 A-ban az exponens
 Ha nem E következett:
 ugrás előre

F985 CBFA SET 7,D
 F987 23 INC HL
 F988 7E LD A,(HL)
 F989 FE98 CP 98
 F98B 280E JR Z,F99B
 F98D FE2B CP 2B
 F98F 280A JR Z,F99B
 F991 FEA2 CP A2
 F993 2804 JR Z,F999
 F995 FE2D CP 2D
 F997 2003 JR NZ,F99C
 F999 CBEA SET 5,D
 F99B 23 INC HL
 F99C 0600 LD B,00
 F99E 2B DEC HL
 F99F 23 INC HL
 F9A0 7E LD A,(HL)
 F9A1 D63A SUB 3A
 F9A3 3012 JR NC,F9B7
 F9A5 C60A ADD A,0A

E esetén "szám" jelzés a D
 regiszterben
 Ellenőrizzük az E utáni
 karaktereket:

ha a "+" tokenje vagy kódja,
 mehet tovább

Ha a "-" tokenje vagy kódja,
 akkor ezt feljegyezzük a D
 regiszterben és azután mehet
 tovább

Itt készítjük el a kitevőt

Beolvassuk a következő
 karaktert és ellenőrizzük,
 hogy számjegy-e?

F9A7 300E JR NC,F9B7
 F9A9 4F LD C,A
 F9AA 78 LD A,B
 F9AB 87 ADD A,A
 F9AC 47 LD B,A
 F9AD 87 ADD A,A
 F9AE 87 ADD A,A
 F9AF 80 ADD A,B
 F9B0 81 ADD A,C
 F9B1 47 LD B,A
 F9B2 F29FF9 JP P,F99F

Ha nem számjegy, ugrás előre

(B)*2

Végül B korábbi értékének 10-
 szereséhez adjuk C-t, s ezt
 tároljuk tovább. Ha a kitevő
 még megfelelő: folytatjuk

F9B5	CBF2	SET	6,D	A "kitevő nagy" jelzés
F9B7	7B	LD	A,D	A kitevő értéke
F9B8	CB6A	BIT	5,D	Ha előtte "-" előjel volt,
F9BA	2802	JR	Z,F9BE	akkor vesszük a (-1)-szeresét,
F9BC	ED44	NEG		s végül A-ban a teljes exponens
F9BE	83	ADD	A,E	
F9BF	FE7F	CP	7F	
F9C1	3802	JR	C,F9C5	Ha az exponens 7EH-nál
F9C3	CBF2	SET	6,D	nagyobb, feljegyezzük
F9C5	D5	PUSH	DE	
F9C6	FD7708	LD	(IY+08),A	Az exponens beírása
F9C9	CB4A	BIT	1,D	Ha negatív szám, az exponens
F9CB	C426F7	CALL	NZ,F726	előjelbitjét beállítja
F9CE	CDDFF9	CALL	F9DF	Ha pedig 0, akkor az exponens
				is 00 lesz
F9D1	D1	POP	DE	Végül vizsgáljuk a D-be írt
F9D2	7A	LD	A,D	feljegyzéseket
F9D3	1F	RRA		E műveletekből az derül ki,
F9D4	A2	AND	D	hogy túlságosan nagy negatív
F9D5	E620	AND	20	kitevő volt-e
F9D7	7A	LD	A,D	Majd pedig az, hogy szám volt-
F9D8	17	RLA		e egyáltalán (CY=1)
F9D9	C4F9F9	CALL	NZ,F9F9	Nagy negatív kitevő esetén a
F9DC	D1	POP	DE	szám 0 lesz
F9DD	C1	POP	BC	
F9DE	C9	RET		

Ebben a részben ez a szubrutin volt a legbonyolultabb és leghosszabb. Láthatjuk, hogy itt sem maga a munka volt igazán bonyolult, hanem egyszerűen aprólékos és egymástól független, szerteágazó körülményt kellett figyelembe venni. Most már lényegesen rövidebb és egyszerűbb szerkezetű programrészek következnek.

0 értékű szám exponensének nullázása

F9DF	E5	PUSH	HL	
F9E0	FDE5	PUSH	IY	
F9E2	E1	POP	HL	
F9E3	0607	LD	B,07	Vesszük sorra a jegyeket
F9E5	AF	XOR	A	
F9E6	23	INC	HL	Ha bármelyik érték nem nulla,
F9E7	B6	OR	(HL)	akkor a ciklusból kilépve
F9E8	2005	JR	NZ,F9EF	abbahagyjuk a további
F9EA	10FA	DJNZ	F9E6	vizsgálatot. Ha végig 0 volt,
F9EC	FD7708	LD	(IY+08),A	az exponens is 00 lesz
F9EF	E1	POP	HL	
F9F0	C9	RET		

0 értéket tesz a BASIC verembe

```
F9F1 CD8EFC    CALL FC8E
F9F4 11F7FF    LD  DE,FFF7
F9F7 FD19      ADD  IY,DE
F9F9 FDE5      PUSH IY
F9FB E3        EX  (SP),HL
F9FC 3609      LD  (HL),09
F9FE 0608      LD  B,08
FA00 AF        XOR  A
FA01 23        INC  HL
FA02 77        LD  (HL),A
FA03 10FC      DJNZ FA01
FA05 E1        POP  HL
FA06 37        SCF
FA07 C9        RET
```

Ellenőrzi, hogy van-e elegendő memóriaterület

Ha igen, akkor beállítja az új veremhatárt, beírja a számok azonosítóját, majd elhelyez 8 db 00 byte-ot a memóriában

A sikeres működés jelzésével tér vissza

Az utolsó érték átírása 1-re

```
FA08 CDF9F9    CALL F9F9
FA0B FD360840  LD  (IY+08),40
FA0F FD360610  LD  (IY+06),10
FA13 C9        RET
```

Először nullázza a jegyeket, majd az exponensbe 40H-t, az első két jegy helyére 10H-t ír és visszatér

Az alábbi utasításokat nem használja a rendszer, a fejlesztés során maradhatott a ROM-ban.

```
FA14 21B6FE    LD  HL,FEB6
FA17 E5        PUSH HL
FA18 18DF      JR  F9F9
```

Egy szám törlése (érvénytelenítése) a stackben

```
FA1A F1        POP  AF
FA1B 110900    LD  DE,0009
FA1E FD19      ADD  IY,DE
FA20 C9        RET
```

Egyik belépési pont, de a következő sor is tehet A veremhatár mutató átállításával a szám elveszettnek mondható

Egy szöveg törlése (érvénytelenítése) a stackben

```
FA21 FD5E01    LD  E,(IY+01)
FA24 1600      LD  D,00
```

FA26 13	INC DE	A veremhatár mutatót a szöveg
FA27 13	INC DE	hosszának megfelelően állítja
FA28 FD19	ADD IY,DE	vissza, így a szöveg
FA2A C9	RET	elveszettnek mondható

HL értékének betöltése a BASIC stackbe

FA2B CDF1F9	CALL F9F1	Először egy 0 számot helyezünk
FA2E FD23	INC IY	el
FA30 FD221C17	LD (171C),IY	Az utolsó jegyek címét
FA34 FD2B	DEC IY	tároljuk
FA36 E5	PUSH HL	
FA37 CD83FC	CALL FC83	Ha HL negatív, szorozza (-1)-
FA3A 37	SCF	gyel, majd bitjeit a
FA3B ED6A	ADC HL,HL	legmagasabb helyi értékű
FA3D 30FC	JR NC,FA3B	helyre mozgatja. Sőt azon túl
		úgy, hogy a legnagyobb értékű
		bit CY-ba kerül
FA3F 0E01	LD C,01	Itt a veremben felhasznált
FA41 180F	JR FA52	jegyeket számlálja
FA43 04	INC B	Az átvitel miatt új jegy
FA44 0C	INC C	írására is szükség lett
FA45 3E41	LD A,41	Így nincs funkciója, de itt
(FA46 LD B,C)		beállítja a ciklusszámlálót
FA47 13	INC DE	
FA48 1A	LD A,(DE)	HL bitjeit balra léptetve,
FA49 8F	ADC A,A	ezzel szinkronban a verembeli
FA4A 27	DAA	sámjegyeket címzi, mindig
FA4B 12	LD (DE),A	megszorozza 2-vel, s ha a HL-
FA4C 10F9	DJNZ FA47	ből CY-ba kilépő bit értéke 1,
FA4E 38F3	JR C,FA43	akkor még 1-gyel növeli. Ezzel
FA50 ED6A	ADC HL,HL	a kitűzött feladatot oldja meg
FA52 ED5B1C17	LD DE,(171C)	
FA56 Z0EE	JR NZ,FA46	
FA58 F1	POP AF	
FA59 E680	AND 80	
FA5B F649	OR 49	Végül beállítja az exponenst
FA5D FB7708	LD (IY+08),A	és visszatérés a normalizálás
FA60 C334F7	JP F734	rutinján át

A számváltozó HL címtől tárolt értéket szabványos formátumban a BASIC verembe tölti.

FA63 CD8EFC	CALL FC8E	Területellenőrzés
FA66 010500	LD BC,0005	
FA69 09	ADD HL,BC	A szám exponense

FA6A FDE5	PUSH IY	
FA6C D1	POP DE	
FA6D 1B	DEC DE	Először a verembe másolja az
FA6E EDA8	LDD	exponenst, majd elé ír egy 00
FA70 AF	XOR A	byte-ot (túlcscordulás)
FA71 12	LD (DE),A	
FA72 1B	DEC DE	
FA73 03	INC BC	Utána sorra átmásolja a tárolt
FA74 EDB8	LDDR	számjegyeket, majd elé még egy
FA76 12	LD (DE),A	00 byte-ot, s végül az
FA77 3E09	LD A,09	azonosítót. A visszatérés a
FA79 1B	DEC DE	következő rutin végét
FA7A 1811	JR FASD	felhasználva történik

Egy szöveges változó HL címtől tárolt értékét a verembe tölti

FA7C 23	INC HL	
FA7D 4E	LD C,(HL)	A karakterlánc hossza
FA7E 0600	LD B,00	
FA80 CD8EFC	CALL FC8E	Ha van elegendő szabad
FA83 09	ADD HL,BC	memória, akkor HL-t a szöveg
FA84 03	INC BC	végére állítja,
FA85 FDE5	PUSH IY	
FA87 D1	POP DE	DE-t a verem következő szabad
FA88 1B	DEC DE	címére, s átmásolja
FA89 EDB8	LDDR	Elé írja a szöveg azono-
FA8B 3E01	LD A,01	sítóját,
FA8D 12	LD (DE),A	
FA8E D5	PUSH DE	beállítja az új veremhatárt és
FA8F FDE1	POP IY	kész
FA91 C9	RET	

Duplikál egy számot a BASIC stackben (kétszerez, újra megismétel)

FA92 CD8EFC	CALL FC8E	Területellenőrzés
FA95 FDE5	PUSH IY	
FA97 D1	POP DE	
FA98 1B	DEC DE	
FA99 210900	LD HL,0009	A duplikálást egyszerű átmoz-
FA9C 4D	LD C,L	gatással valósítjuk meg
FA9D 44	LD B,H	
FA9E 19	ADD HL,DE	
FA9F EDB8	LDDR	
FAA1 13	INC DE	
FAA2 D5	PUSH DE	Végül IY-ba töltjük az új
FAA3 FDE1	POP IY	veremhatárt
FAA5 C9	RET	

Két számot duplikál a BASIC verembe

FAA6 CD8EFC	CALL FC8E	Ha van elegendő memóriaterület,
FAA9 FDE5	PUSH IY	
FAAB E1	POP HL	
FAAC 2B	DEC HL	a szükséges paraméterek beállítása után a duplikáció egyszerű átmozgatással történik
FAAD 5D	LD E,L	
FAAE 54	LD D,H	
FAAF 011200	LD BC,0012	
FAB2 09	ADD HL,BC	
FAB3 EDB8	LDDR	
FAB5 13	INC DE	Beállítjuk az új veremhatárt és kész
FAB6 D5	PUSH DE	
FAB7 FDE1	POP IY	
FAB9 C9	RET	

A BASIC utasításokban következő kifejezés értékét a HL regiszterpárba tölti.

FABA CD43FC	CALL FC43	Lehívja a következő utasítást és a kifejezés kiértékelése után elvégzi a konverziót
FABD 1805	JR FAC4	

A következő szubrutint a BASIC rendszer változatos belépési pontokon használja, de felhasználói programjainkban mi magunk is többféle címen hívhatjuk. Ez lesz egyben a fejezet utolsó nagyobb lélegzetű segédrutinja.

A BASIC veremben elhelyezett x és y számokat a DE és a HL regiszterpárokba konvertálja

FABF CDC3FA	CALL FAC3	Először y-t HL-be, innen rögtön DE-be töltjük, majd a 0FAC3H rutint megismételve x-t HL-be konvertáljuk
FAC2 EB	EX DE,HL	

A BASIC veremben levő értéket HL-be konvertálja

FAC3 F637	OR 37	Ha itt lépünk be, akkor az OR művelettel beállított CY=0 miatt nem hívódik meg a kifejezést kiértékelő rutin, a feldolgozás nélkül megy tovább
FAC5 C5	PUSH BC	
FAC6 D5	PUSH DE	
FAC7 DCA7F0	CALL C,FOA7	

A BASIC kifejezés értéket HL-be konvertálja

FAC4 37	SCF	
FAC5 C5	PUSH BC	Most CY=1 beállítása miatt
FAC6 D5	PUSH DE	először kiértékeli a
FAC7 DCA7F0	CALL C,FOA7	kifejezést
FACA FD7E08	LD A,(IY+08)	Az exponens (exp)
FACD F5	PUSH AF	A 40H-val megnövelt kitevő (e)
FACE E67F	AND 7F	
FAD0 ED62	SBC HL,HL	HL=0000
FAD2 FE45	CP 45	Ha a verembeli szám túl nagy:
FAD4 303E	JR NC,FB14	hiba
FAD6 D640	SUB 40	Ha tört: törli a veremből és
FAD8 382C	JR C,FB06	kész
FADA C605	ADD A,05	A megengedett nagyságrend
FADC 4F	LD C,A	beállítása
FADD 0604	LD B,04	Az első jegyre állunk
FADF 7C	LD A,H	A 10-zel való szorzással
FAE0 E6E0	AND E0	minimum 3 biteltolás jár, ha az
FAE2 2030	JR NZ,FB14	eddig érték már túl nagy: hiba
FAE4 29	ADD HL,HL	
FAE5 5D	LD E,L	
FAE6 54	LD D,H	
FAE7 29	ADD HL,HL	
FAE8 29	ADD HL,HL	HL korábbi értékét szorozzuk
FAE9 19	ADD HL,DE	10-zel (dec.)
FAEA 3828	JR C,FB14	Ha túl nagy: hiba
FAEC CDD2F7	CALL F7D2	Beolvassuk a veremből a
FAEF 5F	LD E,A	kijelölt számjegyet és
FAF0 1600	LD D,00	hozzáadjuk HL eddigi értékéhez
FAF2 19	ADD HL,DE	
FAF3 381F	JR C,FB14	Ha túl nagy: hiba
FAF5 04	INC B	Vesszük a szám következő
FAF6 79	LD A,C	jegyét, s ha még nem értük el
FAF7 B8	CP B	az előírt nagyságrendet,
FAF8 20E5	JR NZ,FADF	folytatjuk
FAFA F1	POP AF	Ha kész, betöltjük ismét az
FAFB F5	PUSH AF	exponenst
FAFC B7	OR A	Ha a szám negatív előjelű volt,
FAFD FC86FC	CALL M,FC86	akkor HL-t komplementálja
FB00 F1	POP AF	
FB01 AC	XOR H	
FB02 FA14FB	JP M,FB14	Ha túl nagy szám: hiba
FB05 3EF1	LD A,F1	Így üres utasítás, de ha
(FB06 POP AF)		ideugrik, lényeges!
FB07 110900	LD DE,0009	Törli a számot a veremből
FB0A FD19	ADD IY,DE	
FB0C 7C	LD A,H	Az S jelzőbitben jelzi, ha a
FB0D B7	OR A	szám negatív
FB0E D9	EXX	

FB0F 78	LD	A,B	Végül A-ban visszaadja az
FB10 D9	EXX		utolsó utasításflaget és kész
FB11 D1	POP	DE	
FB12 C1	POP	BC	
FB13 C9	RET		

Hibakezelés

FB14 CF	RST	8	
FB15 04		hibakód:04	Bad argument

A BASIC veremben levő számot A-ba konvertálja

FB16 CD43FC	CALL	FC43	A következőt utasítás
FB19 3EF6	LD	A,F6	Ez itt érdektelen hatású CY=1-
FB1B 37	SCF		et állít be, de:
(FB1A F637	OR	37)	itt belépve CY=0-val mehet
			tovább! Ezért a belépési
FB1C E5	PUSH	HL	ponttól függően kétféleképpen
FB1D CDC5FA	CALL	FAC5	használjuk az értéket HL-be
			konvertáló rutint!
FB20 24	INC	H	
FB21 25	DEC	H	
FB22 C214FB	JP	NZ,FB14	Ha H nem üres: hiba
FB25 7D	LD	A,L	Egyébként L-t az A-ba töltve
FB26 E1	POP	HL	kész is
FB27 C9	RET		

Szám másolása a BASIC veremből a szimbólumtábla HL által mutatott területére

FB28 CD67F7	CALL	F767	Az utolsó jegyek kerekítése
FB2B EB	EX	DE,HL	
FB2C FDE5	PUSH	IY	
FB2E E1	POP	HL	
FB2F 23	INC	HL	
FB30 23	INC	HL	
FB31 010500	LD	BC,0005	Először a számjegyeket,
FB34 EDB0	LDIR		
FB36 23	INC	HL	majd az exponenst másolja át és
FB37 EDA0	LDI		kész. Visszatérés a
FB39 181C	JR	FB57	szövegmásoló rutin végét
			felhasználva

A BASIC veremben levő elem bemásolása a szimbólumtáblába (vezérlés)

FB3B DDCB014E	BIT	1,(IX+01)	Ha szám, az előbbi rutinra
FB3F 20E7	JR	NZ,FB28	ugrik, egyébként itt folytatja

Szöveg másolása a BASIC veremből a szimbólumtábla HL által mutatott területére

FB41 4E	LD	C,(HL)	A szöveg előírt hossza a
FB42 23	INC	HL	szimbólumtáblában
FB43 FDE5	PUSH	IY	
FB45 D1	POP	DE	
FB46 13	INC	DE	Ez pedig a tényleges hossz a
FB47 1A	LD	A,(DE)	veremben
FB48 0C	INC	C	
FB49 2001	JR	NZ,FB4C	A hossz legfeljebb 0FFH lehet
FB4B 0D	DEC	C	
FB4C B9	CP	C	Ha a verembeli szöveg az
FB4D D212F9	JP	NC,F912	előírtnál hosszabb: hiba
FB50 4F	LD	C,A	(overflow)
FB51 0600	LD	B,00	
FB53 03	INC	BC	Ha a szöveg hossza megfelelő,
FB54 EB	EX	DE,HL	elvégzi a bemásolást,
FB55 EDB0	LDIR		
FB57 E5	PUSH	HL	a veremmutatót visszaállítva
FB58 FDE1	POP	IY	az elemet törli és
FB5A C9	RET		kész

3.6.7.2 Még néhány apróság

Az U0 RAM első 48 byte-ja

A 0FB5B...0FB81H ROM-címeken az a terület található, amelyet az inicializálások -- a BASIC indítása -- alkalmával a 0008...002FH U0 RAM részére kell másolni. Itt található a hibakezelő rutinok, az RST 18 hívás belépési pontjai, de itt találjuk a funkcióhívásokat sokszor a CALL 001BH formában hívó szubrutin vezérlő részét is. Végül az USRTAB, az EXT utasítások kezdőcímeit tartalmazó táblázat is itt kapott helyet, természetesen 00-ra inicializálva.

Amikor az RST 08 hibakezelő rutint hívjuk, láttuk, hogy az utána következő byte tartalmazza a hibakódot. A CPU normál szubrutinhívásként hajtja végre az RST 8 utasítást, tehát a visszatérési címet megjegyzi és a 0008 memóriacíme ugrik, ahol a következő áll:

FB5B E1	POP	HL	Ez a visszatérési cím, amely
FB5C 7E	LD	A, (HL)	éppen a hibakódra mutat. Ha ez
FB5D B7	OR	A	00, akkor nincs hiba, erre a
FB5E E5	PUSH	HL	címre visszatérve NOP
FB5F 00	NOP		utasítást jelent, tehát a cím
FB60 00	NOP		visszatehető. Általában
FB61 00	NOP		azonban nyilvánvalóan nem fog
FB62 00	NOP		itt visszatérni
FB63 C8	RET	Z	

Ez már a 0010H címre kerül, s főleg a funkcióhívásoknál gyakori hibaellenőrzési belépési pont. Mint tudjuk, a funkcióhívások mindig A-ban adják vissza a hibakódot, így közvetlenül itt lehet belépni.

FB64 E1	POP	HL	Ha hiba van, akkor
FB65 C35CFD	JF	FD5C	természetesen el lehet dobni a
			visszatérési címet

Rövidesen látjuk, hogyan történik a kijelölt 0FD5CH címtől az A-beli hibakód feldolgozása.

FB68 C30000	JF	0000	Ezek a 0015...0017H címre
			kerülnek, s amikor használjuk,
			a 00-k helye fontos ugrócím

FB6B C382FB	JF	FB82	Az RST 18 utasítás ugrócíme.
			Itt állunk előtte

FB6E 321F00	LD	(001F), A	Ez pedig 001BH-ra kerül, a 00
FB71 F700	RST	30: 00	helyére töltődik A, s
FB73 C9	RET		lebonyolódik egy funkcióhívás

FB74 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Ezek inicializáláskor az USRTAB területet törlik.

Az RST 18 vezérlőrutinja

Az RST 18 használatával már sokszor találkoztunk, s láttuk, hogy pl. a szinusz függvény kiszámításánál egy-egy RST 18 hívás után a funkciószámok hosszú sorozata áll. Most tisztázódik ezek feldolgozásának módszere.

RST 18 Vezérlés

FB82 221E17	LD	(171E),HL	Menti az aktuális értékeket
FB85 320417	LD	(1704),A	
FB88 2A1817	LD	HL,(1718)	Ezt a rekeszt is használja,
FB8B E3	EX	(SP),HL	értékét a verembe tölti,
			egyben HL-be kerül a
			viisszatérési cím.
FB8C 7E	LD	A,(HL)	Ezen a címen épp a funkciószám
FB8D 23	INC	HL	van. Ráállunk a következő
FB8E 221817	LD	(1718),HL	címre és tároljuk
FB91 87	ADD	A,A	A funkciószám 2-szerese éppen
FB92 F5	PUSH	AF	az eltolás lehet a
			végrehajtási címhez

Egyúttal azonban arra is jó, hogy ellenőrizzük, ez az elem az utolsó volt-e a funkciószámok sorozatában. Ha igen: b7=1 miatt az összeadás után carry=1 lesz.

FB93 C6C7	ADD	A,C7	Hozzáadjuk a táblázat elejé-
FB95 6F	LD	L,A	hez, így a végrehajtási címhez
FB96 26C0	LD	H,C0	jutunk
FB98 7E	LD	A,(HL)	
FB99 23	INC	HL	
FB9A 66	LD	H,(HL)	Kiolvassuk e címet és
FB9B 6F	LD	L,A	letesszük a 0016H-ra, ezzel a
FB9C 221600	LD	(0016),HL	0015H-n elhelyezett 0C3H-val
FB9F 2A1E17	LD	HL,(171E)	egy ugróutasítást hozunk létre
FBA2 3A0417	LD	A,(1704)	
FBA5 CD1500	CALL	0015	Meghívjuk ezt a rutint
FBA8 320417	LD	(1704),A	Ezzel a feladat lényegében
FBAB 221E17	LD	(171E),HL	kész, csak még elő kell
FBAE 2A1817	LD	HL,(1718)	készíteni a folytatást
FBB1 F1	POP	AF	Az utolsó kód volt?
FBB2 30D8	JR	NC,FB8C	Ha nem: ugrás vissza és
			folytatja ugyanezeket
FBB4 E3	EX	(SP),HL	Az utolsó kód után pedig
FBB5 221817	LD	(1718),HL	helyreállítja az összes hasz-
FBB8 2A1E17	LD	HL,(171E)	nált regisztert, de még a 1718H
FBBB 3A0417	LD	A,(1704)	tartalmát is
FBBE C9	RET		és kész

Érdemes ezt a technikát alaposan tanulmányozni. A felhasználói programjaink szerteágazó rutinjai ilyen módszerrel biztonságosan kézben tartható, imponálóan jól szervezett struktúrát alkothatnak!

Adott esetben természetesen maga az RST 18-ra szervezett funkciósorozat is jól használható.

Segédrutinok

A következő szubrutint az editorral beolvasott BASIC sorok tokenizálásánál használja a rendszer, de szükség esetén más konverziós célokra is használható. Működése teljesen nyilvánvaló, így nem részletezzük.

Kódkonverzió, segédrutin

FBBF	LD	A, (DE)	FBCE	RET	C
FBC0	INC	B	FBCF	CP	7B
FBC1	DEC	B	FBD1	RET	NC
FBC2	RET	NZ	FBD2	AND	DF
FBC3	CP	FF	FBD4	RET	
FBC5	RET	Z	FBD5	CP	10
FBC6	AND	7F	FBD7	RET	C
FBC8	CP	20	FBD8	CP	19
FBCA	JR	C,FBDS	FBDA	RET	NC
FBCC	CP	61	FBD8	AND	EF
			FBDD	RET	

Sorszám keresés. Előállítja a BASIC veremben tárolt sorszámú programsor kezdőcímét

FBDE	FE02	CP	02	Ha a veremben nem szám
FBE0	C25AFD	JP	NZ,FD5A	van: hiba!
FBE3	CDC3FA	CALL	FAC3	A sorszám HL-be, majd a sor
FBE6	CD44DD	CALL	DD44	kezdőcíme is HL-be
FBE9	D0	RET	NC	Ha a keresés sikeres volt:
				visszatér
FBEA	CF	RST	8	Egyébként hiba:
FBEB	02	hibakód:	02	Line missing

Tudjuk, hogy a TVC BASIC több utasítása kiadható periféria megjelölésével, pl.:

```
GET #5: X$
PRINT #4:X$
```


Beolvas egy karaktert a magnetofonról, majd kiírja azt a nyomtatóra.

Az előbbi művelethez természetesen a számítógép és a magnetofon közötti kommunikációt megfelelően elő kell készíteni. (OPEN-nel olvasásra megnyitott puffert file)

Az elsődleges eszköz a ki- és beviteli műveletek során mindig a képernyő, az editor, ill. a billentyűzet. A felhasználó perifériahivatkozásait a következő program kezeli.

I/O perifériák kezelése (vezérlés)

FBEC 0E20	LD C,20	Elsődleges az editor
FBEE CDFDFB	CALL FBFD	Ellenőrzi és elvégzi a perifériakijelölést
FBF1 C0	RET NZ	Ha nem volt: visszatér
FBF2 FEFD	CP FD	Ellenőrzi a ":" meglétét és ha van: lehívja a következő utasítást
FBF4 284D	JR Z,FC43	
FBF6 FEFE	CP FE	Ha nincs ":", akkor csak sor vége lehet,
FBF8 D0	RET NC	egyébként értelmetlen:
FBF9 CF	RST 8	Not understood
FBFA 01	hibakód: 01	

A perifériahivatkozások teljesítése

FBFB 0E20	LD C,20	Érvényes funkcióosztályként
FBFD DD7105	LD (IX+05),C	először az editort jelöli ki
FC00 D9	EXX	
FC01 78	LD A,B	
FC02 D9	EXX	Az utasítás után szerepel a "#" jel?
FC03 FEA7	CP A7	
FC05 C0	RET NZ	Ha nem: visszatér, kész
FC06 CD16FB	CALL FB16	Ha igen: az utána álló számot A-ba konvertálja
FC09 E607	AND 07	
FC0B 87	ADD A,A	Ezután csak a b2...b0 biteket hagyva meg, ezeket a megfelelő b6...b4 bitekbe viszi, s ez lesz az érvényes funkcióosztály
FC0C 87	ADD A,A	
FC0D 87	ADD A,A	
FC0E 87	ADD A,A	
FC0F 320517	LD (1705),A	
FC12 4F	LD C,A	A C-ben adja át
FC13 AF	XOR A	Beállítja a Z=1 és CY=0 jelzéseket,
FC14 D9	EXX	
FC15 78	LD A,B	A-ba a következő utasításflaget tölti és kész
FC16 D9	EXX	
FC17 C9	RET	

Kisebbs inicializáló rutin következik. A BASIC sokszor hívja meg. A dolga az, hogy biztosítsa a képernyőre írt karakterek láthatóságát az INK és a PAPER egyszerű illesztésével -- nem viselvé el a két kód egyenlőségét.

INK és PAPER illesztése

FC18	LD	(IX+05),20	FC29	LD	A,(0B13)
FC1C	LD	A,(0B4E)	FC2C	BIT	1,A
FC1F	LD	C,A	FC2E	RET	Z
FC20	LD	A,(0B4D)	FC2F	LD	A,C
FC23	CP	C	FC30	CPL	
FC24	RET	NZ	FC31	LD	(0B4D),A
FC25	INC	A	FC34	RET	
FC26	LD	(0B4D),A			

A következő rutin a nyitott file-okat zárja le

FC35	BIT	3,(IX+00)	Ha nincs is nyitott file,
FC39	RET	Z	akkor visszatér
FC3A	RES	3,(IX+00)	Egyébként adminisztrálja és
FC3E	RST	30: 54	végrehajtja a zárást
FC40	RST	30: D4	
FC42	RET		

Most az egész BASIC rendszer talán leggyakrabban hívott rutinja következik. A 0FC43H címhez értünk, s az Olvasó, ha eddig követte a könyv előző fejezeteit, tudhatja, hogy szinte minden végrehajtási műveletben többször is leírtuk.

Mindig csak ezt írtuk mellé: "Lehívja a következő utasítást". Tekintettel arra, hogy a TV-Computer BASIC működése lényegének megértéséhez szükséges, most vizsgáljuk meg részletesebben a kérdést.

Az egyes BASIC utasítások végrehajtása közben a CPU másodlagos regiszterkészletében

- HL' mutat a következő végrehajtandó utasítás címére,
- B' az utasításflag, az aktuálisan kezelt utasításbyte-ról tartalmaz fontos információkat,
- C' szimbólumtáblában elérhető adat típusbyte-ját
- DE' a típusbyte memóriacímét tartalmazza

Egy-egy utasítás végrehajtásához a BASIC vermet is megfelelően elő kell készíteni.

Ezt a nagyszabású, összetett feladatot vállalja magára a most megtárgyalásra kerülő szubrutin. Munkája tehát röviden úgy összegezhető, hogy folyamatosan készíti elő a terepet a soron következő feladatok végrehajtásához.

A következő utasítás leihívása

FC43 D9	EXX		
FC44 7E	LD	A, (HL)	A következő parancsbyte
FC45 47	LD	B, A	
FC46 FEFE	CP	FE	Ha FE ("!" token), vagy
FC48 3030	JR	NC, FC7A	FF (sorvége): visszatér, carry=0, HL marad itt
FC4A 23	INC	HL	A többi esetben rááll a következő byte-ra
FC4B FEC5	CP	C5	Az INKEY\$ tokenje, speciális szöveg-parancs
FC4D 282F	JR	Z, FC7E	Ha ez a vizsgált byte, ugrás előre
FC4F FE80	CP	80	Ha b7=1, tehát tokenről van szó:
FC51 3027	JR	NC, FC7A	ugrás előre
FC53 FE20	CP	20	A betűközöket átlépi,
FC55 28ED	JR	Z, FC44	továbbmegy
FC57 2B	DEC	HL	Ha nem token: visszaáll az aktuális címre
FC58 0600	LD	B, 00	
FC5A FE22	CP	22	Idézőjel esetén a szöveg
FC5C 37	SCF		betöltése következik a BASIC
FC5D 2817	JR	Z, FC76	verembe
FC5F CBC8	SET	1, B	Megnézzük, hogy szám-e?
FC61 FE41	CP	41	Ha a kód a betűk előtt van:
FC63 3811	JR	C, FC76	ugrás, lehet szám
FC65 FE5B	CP	5B	
FC67 3F	CCF		A nagybetűk utáni kódok
FC68 380C	JR	C, FC76	elutasítása
FC6A CDD2F3	CALL	F3D2	A betűket szimbólumhivatkozásnak tekintjük és keressük a szimbólumtáblában
FC6D D40BF4	CALL	NC, F40B	Ha nincs: új szimbólumként adminisztráljuk
FC70 79	LD	A, C	
FC71 E602	AND	02	A típusbyte, amelyből csak b1-et használjuk

FC73 F601 OR 01 A b0-lal együtt jelezzük, hogy
FC75 47 LD B,A milyen változó volt:
 01 -- szöveges
 03 -- szám

Ez már a befejezés:

FC76 DC05F9 CALL C,F905 A szám- vagy szövegkonstansokat
 tölti a BASIC verembe

FC79 78 LD A,B
FC7A D9 EXX
FC7B FEFD CP FD
FC7D C9 RET
 Ez még egy végső jelzést tesz
 lehetővé: Z=0, ha az aktuális
 byte ":"

FC7E 4F LD C,A
FC7F 0601 LD B,01
FC81 18F6 JR FC79
 Erre csak az INKEY\$-nál kerül
 sor: C=C5H

Foglaljuk össze táblázatosan a működés legfontosabb paramétereit:

	kód	HL'	A=B'	Z	CY	C'	DE'
sor vége	FE, FF	marad	FE, FF	0	0		
kettőspont	FD	+1	FD	1	0		
tokenek	80...FC	+1	token	0	1		
de INKEY\$	C5	+1	01	0	1	C5	
szimbólum	41...5B	név után	1 string 3 szám	0 0	1 1	típus- byte	adat címe
szám	30...39	szám után	02	0	1		
szöveg	bármilyen	szöveg után	0	0	1		

A nem megfelelő paraméterezés általában Not understood hibát okoz.

Figyeljük meg, hogy az érdemi feldolgozást önmagában CY=1 jelzi, HL' csak a sor végén nem megy tovább. Figyelemre méltó a szimbólumok előkészítése is.

Ezekután az előbbi rutin gyakori hívása már érthető. A programfutás vezérlésében betöltött szerepe teljesen alapvető, munkája és információi teremtik meg a tevékenység gördülékeny végrehajtásának feltételeit.

Egy rövidke rutin ékelődött be a ROM következő részébe. Működése annyi, hogy ha HL negatív (a H 7. bitje 1), akkor komplementálja:

```

FC83 BIT 7,H
FC85 RET Z
FC86 DEC HL
FC87 LD A,L
FC88 CPL
FC89 LD L,A
FC8A LD A,H
FC8B CPL
FC8C LD H,A
FC8D RET

```

A komplementálások eredményeként a negatív előjelű szám abszolút értékét kapjuk

Végül néhány olyan rutin következik, amely részletezésre nem szorul, így ezeknek is csak rövidített formátumú listáját közöljük.

Területellenőrzés

```

FC8E PUSH HL
FC8F PUSH DE
FC90 LD DE,0100
FC93 CALL FC99
FC96 POP DE
FC97 POP HL
FC98 RET

FC99 LD HL,(1726)
FC9C ADD HL,DE
FC9D JR C,FCB1
FC9F EX DE,HL
FCA0 PUSH IY
FCA2 POP HL
FCA3 SBC HL,DE
FCA5 JR C,FCB1
FCA7 SBC HL,HL
FCA9 ADD HL,SP
FCAA LD DE,(0B17)
FCAE SBC HL,DE
FCB0 RET NC

```

Hibakezelés:

```

FCB1 RST 8
FCB2 06 No memory

```

Segédrutin a többelemek eléréséhez

```

FCB3 XOR A
FCB4 LD L,A
FCB5 LD H,A
FCB6 LD A,10
FCB8 DEC A
FCB9 RET M
FCBA ADC HL,HL
FCBC SLA C
FCBE RL B
FCC0 JR NC,FCB8
FCC2 ADD HL,DE
FCC3 JR FCB8

```

A következő utasítás keresése. Elmegy egy lezáró ":"-ig vagy a sor végéig

```

FCC5  CALL FCD1
FCC8  RET  NC
FCC9  CP   CA
FCCB  RET  NC
FCDC  CALL FCD5
FCDF  JR   FCC8

FCD1  EXX
FCD2  PUSH HL
FCD3  EXX
FCD4  POP  HL
FCD5  LD   A,(HL)
FCD6  CP   FE
FCD8  RET  NC
FCD9  INC  HL
FCDA  CP   FD
FCDC  JR   NZ,FCD5
FCDE  LD   A,(HL)
FCDF  INC  HL
FCE0  CP   20
FCE2  JR   Z,FCE4
FCE4  DEC  HL
FCE5  SCF
FCE6  RET

```

Az A-ban megadott azonosítójú területre áll a BASIC veremben

```

FCE7  LD   B,00
FCE9  PUSH IY
FCEB  POP  HL
FCEC  LD   C,(HL)
FCED  CP   C
FCEE  RET  Z
FCEF  INC  C
FCF0  DEC  C
FCF1  SCF
FCF2  RET  Z
FCF3  DEC  C
FCF4  JR   NZ,FCF3
FCF6  INC  HL
FCF7  LD   C,(HL)
FCF8  INC  C
FCF9  ADD  HL,BC

```

```

FCFA  PUSH HL
FCFB  POP  IY
FCFD  JR   FCEC

```

Segédrutin a POKE utasítás címének ellenőrzéséhez

```

FCFF  CALL F0A7
FD02  LD   HL,8000
FD05  PUSH HL
FD06  CALL FA2B
FD09  CALL F493
FD0C  CALL FAC3
FD0F  POP  DE
FD10  ADD  HL,DE
FD11  RET

```

Ugrótábalkezelés a SET és SOUND utasításokhoz

```

FD12  LD   C,A
FD13  LD   A,(HL)
FD14  OR   A
FD15  LD   A,C
FD16  RET  Z
FD17  SUB  (HL)
FD18  INC  HL
FD19  LD   E,(HL)
FD1A  INC  HL
FD1B  JR   NZ,FD13
FD1D  LD   D,A
FD1E  ADD  HL,DE
FD1F  CALL FC43
FD22  CALL DBAD
FD25  SCF
FD26  RET
FD1E  ADD  HL,DE
FD1F  CALL FC43
FD22  CALL DBAD
FD25  SCF
FD26  RET

```

Végül a karakterláncok szeleteinek előállításához használt segédrutin listáját közöljük.

FD27	CALL	FC43		A	hivatkozott	0FD45H-s
FD2A	LD	E,01		rutin	pedig	az alábbi:
FD2C	JR	Z,FD36				
FD2E	CALL	FD45	FD45	CALL	FAC4	
FD31	LD	E,D	FD48	LD	D,00	
FD32	CP	FD	FD4A	INC	H	
FD34	JR	NZ,FD40	FD4B	RET	Z	
FD36	CALL	FC43	FD4C	DEC	D	
FD39	LD	D,FF	FD4D	DEC	H	
FD3B	CP	95	FD4E	RET	NZ	
FD3D	CALL	NZ,FD45	FD4F	LD	D,L	
FD40	LD	A,95	FD50	RET		
FD42	JP	FD54				

Az utasítások formai ellenőrzésére igen gyakran használt módszer, hogy az A regiszterben megadjuk a következő utasításbyte-ként elfogadható egyedüli értéket, s csak akkor megyünk tovább, ha valóban ezt találjuk. Ennek rutinja:

FD51	CD43FC	CALL	FC43	Az ellenőrzés a 0FD54H
FD54	D9	EXX		címen belépve hatásos: ha a
FD55	B8	CP	B	lehívott parancsbyte nem
FD56	D9	EXX		egyezik a megadottal:
FD57	CA43FC	JP	Z,FC43	nem folytatja

Hanem ráfut az itt következő hibakezelő ágra!

3.6.8 Hibakezelés

3.6.8.1 Bevezetés

Amikor a BASIC a további működését lehetetlené tevő állapotot észlel, akkor félbeszakítja a munkát és a helyzetről igyekszik tájékoztatni a felhasználót.

Ez az ember és a gép közötti párbeszéd fontos része.

Az előforduló hibák minden esetben azt jelentik, hogy a számítógép nem képes az ember által előírt feladatot

teljesíteni. Ami persze sokszor nem a gép képességein múlik, hiszen egy rosszul megszervezett utasítássorozat lehet önmagában is értelmetlen, ellentmondásos, vagy pedig egyszerűen csak arról van szó, hogy -- esetleg szándékaink ellenére -- a gép szókincsében nem szereplő kifejezéseket használtunk.

A hibák egy része nyilvánvaló okokra vezethető vissza és bizonyos típusokba sorolható. Ezekről a BASIC hibakezelő programja szöveges információkat is közöl a felhasználóval. A legtöbb esetben azt a program- vagy parancssort is kiírja, amelynek végrehajtása közben a hibát észlelte. Ez természetesen sokszor nem azonos azzal, ami a hiba tényleges forrása. Azonban abból, hogy a gép mely ponton nem tudta tovább folytatni a működést, messzemenően következtetni lehet a hiba eredetére is.

A hibakezelő az RST 8, RST 10 vagy közvetlenül az 0FD5AH vagy a 0FD5C hívásokkal érhető el.

Ha a hiba oka ismert, célszerű az A-ban elhelyezett hibakóddal a 0FD5C címre ugrani, vagy a rövidebb RST 8 hívást használni. Utóbbi esetben az RST 8 utáni byte-ba kell írni a hibakódot. A funkcióhívások az RST 10 megoldást használják, eleve az A regiszterben visszaadva a hiba kódját. A 0FD5AH belépési pont 01-es hibakódot generál.

A funkcióhívásokat használó felhasználói programokban is célszerű lehet a hívások után az RST 10 használata, bár ez a BASIC hibakezelőjére ugrathatja a gépet, s ez nem mindig kívánatos.

Végül itt is felhívjuk a figyelmet arra, hogy a teljes hibakezelő vezérlő része az U0 RAM elején van, tehát átírható. Ez természetesen nagy megfontoltságot követel, hiszen a rendszer működése fölött gyakorolt saját felügyelete megváltoztatásáról van szó. A felhasználó céljait szolgáló változtatások azonban indokoltak lehetnek és főleg egyszerűen kivitelezhetők.

3.6.8.2 Hibakezelő alprogram

FD5A 3E01	LD	A,01	01-es hibakód generálása Not understond
FD5C FD2A1A17	LD	IY,(171A)	Általános belépési pont IY-t a legutóbbi, még helyes működésnél tárolt értékre állítja

FD60 FEF5	CP	F5
FD62 CAA3FF	JP	Z,FFA3
FD65 F5	PUSH	AF
FD66 FE06	CP	06
FD68 CCFDC	CALL	Z,DCFC
FD6B CD35FC	CALL	FC35
FD6E CD18FC	CALL	FC18
FD71 CD79FE	CALL	FE79
FD74 06 0D 0A 2A 2A 2A 20		
FD7B F1	POP	AF
FD7C CB7F	BIT	7,A
FD7E 2018	JR	NZ,FD98
FD80 21C6FD	LD	HL,FDC6
FD83 01FFFF	LD	BC,FFFF
FD86 03	INC	BC
FD87 09	ADD	HL,BC
FD88 4E	LD	C,(HL)
FD89 23	INC	HL
FD8A 0C	INC	C
FD8B 0D	DEC	C
FD8C 2804	JR	Z,FD92
FD8E B9	CP	C
FD8F 4E	LD	C,(HL)
FD90 20F4	JR	NZ,FD86
FD92 23	INC	HL
FD93 CDDDFE	CALL	FEDD
FD96 1818	JR	FDB0

CTRL+ESC esetén ugrás az ezt kezelő programra

Memóriaahiány esetén a BASIC program működési feltételeinek inicializálása

Lezárja a file-okat, az INK és PAPER illesztése és soremelés után kiírja: "*** "

Ha a hibakód b7 bitje 1 (funkcióhívások), akkor ugrás előre

A szöveges információk táblázata: hibakód+hossz+szöveg+0FFH formában

Eltolás a következő szöveghez
A kód az utolsó elemnél 00, a hiba azonosítása tehát legfeljebb itt megszakad. Ha a megadott kód egyikkel sem azonos, akkor ugrás előre

Az üzenet hossza
Ha nem egyezik: folytatja a keresést
A hibakód azonosítása után kiírja a megfelelő üzenetet, majd ugrás

Az operációs rendszerből eredő hibák feldolgozása:

FD98 6F	LD	L,A
FD99 CD79FE	CALL	FE79
FD9C 0D 53 79 73 74 65 6D 20 65 72 72 6F 72 20		
FDA A AF	XOR	A
FDAB 67	LD	H,A
FDAC 47	LD	B,A
FDAD CD1BFF	CALL	FF1B
FDB0 CD79FE	CALL	FE79
FDB3 04 2E 0B 0D 0A		
FDB8 DDCB00CE	SET	1,(IX+00)
FDBC 2A0C17	LD	HL,(170C)
FDBF AF	XOR	A
FDC0 CD2DDD	CALL	DD2D
FDC3 C30EE1	JP	E10E

L=a kód
Kiírja:
"System error "

HL=a hibakód
A kód decimális kiírása majd a végére:
".", törlés a sor végéig és "ok" üzenet nem kell!
Végül veszi az aktuális sor kezdőcímét és kilistázza a sort
Ezután ugrás az END rutinra

3.6.8.3 Hibaüzenetek

Mivel a hibaüzenetek kiírása is ugyanazzal a rutinnal történik, mint a BASIC soroké, ezért itt is találkozunk tokenekkel.

ROM	Hibakód	Hossz	Karakterek	üzenet
FDC6	01	0F	4E 6F 74 20 75 6E 64 65 72 73 74 6F 6F 64 FF	Not understood
FDD7	02	06	4C 69 6E 65 93 FF	Line missing
FDDF	03	04	41 92 93 FF	Argument missing
FDE5	04	04	91 61 92 FF	Bad argument
FDEB	05	0B	91 73 75 62 73 63 72 69 70 74 FF	Bad subscript
FDF8	06	08	90 6D 65 6D 6F 72 79 FF	No memory
FE02	07	03	90 FB FF	No DATA
FE07	08	03	90 F2 FF	No FOR
FE0C	09	03	90 F0 FF	No GSUB
FE11	0A	03	8F F8 FF	Cannot CONTINUE
FE16	0B	0D	8F 64 69 76 69 64 65 20 62 79 20 30 FF	Cannot divide by 0
FE25	0C	03	8F DB FF	Cannot READ
FE2A	0D	09	4F 76 65 72 66 6C 6F 77 FF	Overflow
FE35	0E	0E	54 79 70 65 20 6D 69 73 6D 61 74 63 68 FF	Type mismatch
FE45	0F	18	56 61 72 69 61 62 6C 65 20 64 65 63 6C 61 72 65 64 20 74 77 69 63 65 FF	Variable declared twice
FE5F	10	06	91 66 69 6C 65 FF	Bad file
FE67	00	0F	42 41 53 49 43 20 63 6F 72 72 75 70 74 65 64 FF	BASIC corrupted

3.6.9 Az utolsó segédrutinok

Szövegkiíró rutin

A szubrutin hívását követő byte-okra el kell helyezni a szöveget hossz+karakterlánc alakban.

FE79 E3	EX (SP),HL	A visszatérési címet a
FE7A CD7FFE	CALL FE7F	veremből kiveszi és ezt a
		területet szöveggé feldol-
		gozza
FE7D E3	EX (SP),HL	A szöveg utáni cím lesz a RET
FE7E C9	RET	cím

Praktikus!

Sok szöveg kiírását igénylő programokban használata, de legalábbis ehhez hasonló program alkalmazása indokolt.

Kiír egy szöveget. Belépéskor HL a hosszbyte-ra mutat

FE7F 7E	LD A, (HL)	
FE80 23	INC HL	
FE81 B7	OR A	Ha a szöveg hossza 0:
FE82 C8	RET Z	visszatérés.
FE83 C5	PUSH BC	
FE84 47	LD B,A	
FE85 7E	LD A, (HL)	
FE86 23	INC HL	
FE87 CD9AFE	CALL FE9A	Kiír egy karaktert
FE8A 10F9	DJNZ FE85	
FE8C C1	POP BC	
FE8D C9	RET	

A következő rutin különböző pontokon belépve sorrendben: töröl a sor végéig, soremelés, A-ban tárolt kódú karakter kiírása funkciókat lát el.

FE8E 3E0B	LD A,0B	Z=1 esetén törlés a sor
FE90 CC9AFE	CALL Z,FE9A	végéig
FE93 3E0D	LD A,0D	Soremelés belépési pont
FE95 CD9AFE	CALL FE9A	
FE98 3E0A	LD A,0A	
FE9A F5	PUSH AF	Belépési pont az A-ban tárolt
FE9B C5	PUSH BC	ASCII karakter kiírásához
FE9C D5	PUSH DE	

FE9D 4F	LD C,A	
FE9E CDA6FE	CALL FEA6	Kiírás az előírt eszközre, hibakezeléssel
FEA1 D7	RST 10	
FEA2 D1	POP DE	
FEA3 C1	POP BC	
FEA4 F1	POP AF	
FEA5 C9	RET	

Funkcióhívás lebonyolítása

Érdekes és nagyon változatosan használható megoldást mutat a funkcióhívások egységes kezelésére a következő kis rutin.

FEA6 117F00	LD DE,007F	D és E a funkciókód előállít- tásában döntő maszk
FEA9 3A0517	LD A,(1705)	Egy másik belépési pont, akkor DE input paraméter lehet.
FEAC F601	OR 01	
FEAE B2	OR D	
FEAF A3	AND E	
FEB0 C31B00	JP 001B	Leteszi az A-ban összeállít- tott funkciókódot és végre is hajtja

Kilépési pont az EXT ROM-ba

FEB3 CD9AFE	CALL FE9A	Egy karakter kiírása
FEB6 E1	POP HL	
FEB7 C3F0FF	JP FFF0	Ugrás: EXT ROM, HL címre

A BASIC veremben levő szám kiírása

FEBA FDCB087E	BIT 7,(IY+08)	Az előjel pozitív?
FEBE CCC7FE	CALL Z,FEC7	Ha igen: betűközt ír
FEC1 CD0EF8	CALL F80E	A számot a 1930H pufferbe, ASCII-ra konvertálja és
FEC4 CD7FFE	CALL FE7F	kiírja
FEC7 3E20	LD A,20	Egy betűköz kiírása
FEC9 18CF	JR FE9A	

Tokenizált sorok kiírása a HL mutatta címtől

FECB 87	ADD A,A	Az első 3 sor az A b7 bitjét
FECF FE40	CP 40	törli
FECE 1F	RRA	A osztása 2-vel (CY=0)

FEEF 0D9AFE	CALL FE9A	Az A-kódú karakter írása
FED2 FE22	CP Z2	Ha nem idézőjel volt,
FED4 2009	JR NZ, FEDE	ugrás előre
FED6 B9	CP C	Ha korábban is az volt,
FED7 2802	JR Z, FEDB	ugrás előre
FED9 41	LD B, C	
FEDA 0E78	LD C, 78	Ez így érdektelen, azonban
(FEDB 78	LD A, B)	0FEDBH-n mást jelent
FEDC 0EAF	LD C, AF	Ez is érdektelen

A belépési pont azonban éppen a 0FEDDH cím, így:

FEDD AF	XOR A	A belépési pontban tehát mást
FEDE 4F	LD C, A	"érez" a CPU, mint az idézőjel
		után

Figyeljük meg, hogy az idézőjelek B és C regiszterben való nyilvántartásával hogyan oldja meg a program -- túrhető szellemességgel -- az idézőjelbe tett bármilyen karakterek pontos és helyes kezelését.

FEDF 7E	LD A, (HL)	
FEE0 23	INC HL	
FEE1 3C	INC A	Ha a beolvasott karakter 0FFH
FEE2 C8	RET Z	volt (sor vége):
		visszatér
FEE3 3D	DEC A	
FEE4 F2CBFE	JP P, FECD	Ha nem token: kiírja
FEE7 E5	PUSH HL	Tokenek feldolgozása előtt
		tárolja a következő címet.
FEE8 0C	INC C	
FEE9 0D	DEC C	A C-beli feltételek figyelése!
FEEA F5	PUSH AF	
FEEB FEFB	CP FB	
FEED 2804	JR Z, FEED	Figyeljük meg, hogy a sor végét
FEED FEFC	CP FC	jelző karakterek: a REM, a "!"
FEF1 3802	JR C, FEFE	tokenje, illetve a ":"
FEF3 0EFF	LD C, FF	karakter kezelése, valamint a
FEF5 FEFD	CP FD	80H-nál nagyobb kódú
FEF7 2003	JR NZ, FEFC	karakterek feldolgozása milyen
FEF9 F1	POP AF	jelzőbitkezelési és C-t állító
FEFA 0C	INC C	eljárással valósul meg --
FEFB F5	PUSH AF	tekintettel az idézőjelekre is
FEFC F1	POP AF	
FEFD 2012	JR NZ, FFI1	

Ezután bonyolódik a tokenek visszafejtése, ugyanabból a már megismert 0DE6DH kezdőcímmű táblázatból.

FEFF 2F	CPL	A token sorszama
FF00 216DDE	LD HL,DE6D	Tudjuk, hogy egy-egy
FF03 CB7E	BIT 7,(HL)	tokenhez tartozó szó utolsó
FF05 23	INC HL	karakterében b7 értéke 1.
FF06 28FB	JR Z,FF03	Ennek alapján a kívánt
FF08 3D	DEC A	sorszámú szó megkereshető
FF09 20F8	JR NZ,FF03	A kiírás ezután már
FF0B 7E	LD A,(HL)	karakterenként történik
FF0C 23	INC HL	Figyelni kell itt is a b7
FF0D CB7F	BIT 7,A	bitet, mert 1 értéke a szó
FF0F CBBF	RES 7,A	végét jelzi
FF11 CD9AFE	CALL FE9A	Utána folytatja a sor
FF14 28F5	JR Z,FF0B	kiírását
FF16 E1	POP HL	
FF17 18C6	JR FEDF	

Ismét két olyan rutin következik, amelynek működése az eddigiek alapján teljesen nyilvánvaló, így alaposabb elemzésétől eltekinthetünk.

A HL-ben tárolt szám decimális kiírása

Belépéskor a B regiszter és a CY jelzőbit maszk-funkciókat lát el. CY-nal a kezdő nullák helyére íródó betűközöket nyomhatjuk el, B-vel pedig a szám végén álló szóköz karaktert.

FF19	LD	B,FF			
(Ha itt lépünk be, a szám végére szóköz kerül)					
FF1B	SBC	A,A	FF2E	ADD	HL,DE
FF1C	AND	Z0	FF2F	DEC	A
FF1E	LD	C,A	FF30	JR	Z,FF39
FF1F	PUSH	HL	FF32	LD	C,30
FF20	LD	HL,FF47	FF34	ADD	A,C
FF23	LD	E,(HL)	FF35	CALL	FE9A
FF24	INC	HL	FF38	LD	A,C
FF25	LD	D,(HL)	FF39	XOR	C
FF26	INC	HL	FF3A	CALL	NZ,FE9A
FF27	EX	(SP),HL	FF3D	EX	(SP),HL
FF28	XOR	A	FF3E	DEC	E
FF29	SBC	HL,DE	FF3F	JR	NZ,FF23
FF2B	INC	A	FF41	POP	HL
FF2C	JR	NC,FF29	FF42	LD	A,C
			FF43	AND	B
			FF44	JR	NZ,FEC7
			FF46	RET	

A hivatkozott 0FF47H címtől a decimális 1, 10, 100 és 1000 van lerakva hexadecimális formában:

FF46	ES 03	64 00	0A 00	01 00
	(1000)	(100)	(10)	(1)

A másik kis szubrutint a BASIC sorok billentyűzetről való bekérése során, ill. az INPUT utasításnál használja a rendszer. A beolvasott adatok a 1831H kezdőcímmű INPUT pufferbe kerülnek.

A rutin az editor közreműködésével végzi a dolgát, a karakteres inputot megvalósító 0A1H-s funkcióhívással (az editor-osztály alaphozzárendelése estén). Mint tudjuk ez a funkcióhívás előírt kilépési feltételekkel lehetővé teszi egy egész sor szerkesztését. Majd ismételt hívásokkal karakterenként beolvasható a sor.

A rutin a sort hossz-byte és sorvégjellel is ellátja.

Egy szerkesztett sor bekérése és tárolása az INPUT pufferben (1831H)

FF4F	LD	HL,1831			
FF52	PUSH	HL	FF7A	LD	A,C
FF53	LD	B,00	FF7B	CP	0D
FF55	PUSH	BC	FF7D	JR	Z,FF92
FF56	LD	DE,80FF	FF7F	CP	20
FF59	CALL	FEA9	FF81	JR	C,FF55
FF5C	POP	DE	FF83	SUB	80
FF5D	LD	B,D	FF85	JR	C,FF8C
FF5E	JR	Z,FF72	FF87	CP	20
FF60	POP	HL	FF89	JR	NC,FF8C
FF61	LD	B,00	FF8B	LD	C,A
FF63	LD	(HL),B	FF8C	LD	A,B
FF64	INC	HL	FF8D	CP	FB
FF65	LD	(HL),FF	FF8F	JR	NC,FF55
FF67	DEC	HL	FF91	INC	B
FF68	CP	F5	FF92	INC	HL
FF6A	JR	Z,FF9B	FF93	LD	(HL),C
FF6C	CP	EC	FF94	JR	NZ,FF55
FF6E	JR	Z,FF9B	FF96	LD	(HL),FF
FF70	OR	A	FF98	XOR	A
FF71	RST	10	FF99	POP	HL
FF72	LD	A,(0B16)	FF9A	LD	(HL),B
FF75	OR	A	FF9B	OR	A
FF76	LD	A,F5	FF9C	RET	
FF78	JR	NZ,FF60			

CTRL+ESC kezelése

A BASIC sűrűn foglalkozik ezzel a kérdéssel. Minden új parancs végrehajtása előtt -- többek között -- elvégzi ezt a vizsgálatot is.

Így gyakorlatilag az a helyzet, hogy amikor az operációs rendszer megszakításkezelő alprogramja nincs letiltva, akkor bármelyik pillanatban megszakíthatjuk a BASIC program, vagy parancs végrehajtását.

Beépítése felhasználói programjainkba is indokolt lehet.

FF9D 3A160B	LD	A, (0B16)	STOP-FLAG
FFA0 B7	OR	A	Volt CTRL+ESC?
FFA1 C8	RET	Z	Ha nem: mehet tovább!
FFA2 D9	EXX		Egyébként meg kell állni

Ha biztosak vagyunk a CTRL+ESC lenyomásában, akkor célszerű belépési pont a következő cím.

FFA3 D9	EXX		
FFA4 AF	XOR	A	
FFA5 32160B	LD	(0B16), A	A STOP-FLAG törlése
FFA8 CD35FC	CALL	FC35	Lezárja a file-okat,
FFAB CD18FC	CALL	FC18	majd az INK és PAPER
FFAE CD79FE	CALL	FE79	illesztése után soremelés
FFB1 06 0D 0A 53 54 4F 50			és kiírja: "STOP"

FFB8 DDCB0056	BIT	Z, (IX+00)	Futó program?
FFBC 2822	JR	Z, FFE0	Ha nem: ugrás előre!
FFBE 221017	LD	(1710), HL	A következő utasításbyte
FFC1 2A0C17	LD	HL, (170C)	címét tárolja, s a következő
FFC4 220E17	LD	(170E), HL	sort címeként is az aktuális
			sort jegyzi fel

FFC7 CD79FE	CALL	FE79	Majd kiírja:
FFCA 09 20 61 74 20 6C 69 6E 65 20			" at line "

FFD4 2A0C17	LD	HL, (170C)	
FFD7 23	INC	HL	Ezután az aktuális BASIC sor
FFD8 5E	LD	E, (HL)	címét beolvassa a DE
FFD9 23	INC	HL	regiszterpárba,
FFDA 56	LD	D, (HL)	
FFDB EB	EX	DE, HL	áttölti HL-be,
FFDC B7	OR	A	
FFDD CD19FF	CALL	FF19	és kiírja a sorszámot
FFE0 AF	XOR	A	
FFE1 CD8EFE	CALL	FE8E	Végül törlés a sor végéig és
FFE4 C3DADA	JP	DADA	soremelés után ugrás a BASIC
			alaphelyzetbe

A BASIC videomemória elérése

Ha a VID-t akarjuk BASIC-ből elérni, ehhez a ROM címeket kell beírni. A 0C000...0FFFFH cím kijelöléséből tudja meg a BASIC a felhasználó ezirányú szándékát, s megoldja a VID belapozását a 2. lapra, valamint a megadott cím szükséges transzformációját is.

FFE7 7C	LD	A,H	
FFE8 FEC0	CP	C0	Ha a felhasználó nem a ROM-ot
FFEA D8	RET	C	címezte: rendben!
FFEB CBB4	RES	6,H	ROM címzés esetén b6=0
			átírással elérjük, hogy a
			C000-FFFF címek 8000H-BFFFH-
			ra transzformálódjanak
FFED 3E50	LD	A,50	Beállítjuk a szükséges
FFEF C9	RET		memórialapozási byte-ot
			és mehet vissza!

3.6.10 Híd az EXT ROM-ba

Nem is annyira a BASIC tanulmányozása, mint inkább a korábbi fejezetekben talákoztunk sokszor azzal a hivatkozással, hogy egyik-másik rutin a munkáját, vagy annak egy részét az EXT ROM-ban folytatja.

Ennek technikája -- már mint az átlépésnek -- rendkívül egyszerű. A HL regiszterpárban elhelyezzük azt a címet, ahová az EXT ROM-ban ugrani akarunk, s ezzel a 0FFF0H címre ugrunk. Itt a következő történik:

FFF0 F5	PUSH	AF	Az A-ba az U0-U1-U2-EXT
FFF1 3EFO	LD	A,F0	konfigurációnak megfelelő
FFF3 320300	LD	(0003),A	értéket töltjük
FFF6 D302	OUT	(02),A	Itt lezajlik a memóriacsere a
			3. lapon

A programszámláló most a következő, 0FFF8H címre mutat. De mind az EXT ROM-ban, mind pedig itt, a fő ROM-ban a programhíd ugyanazokat az utasításokat tartalmazza. Így a CPU meg sem érzi azt, hogy ettől kezdve egy egészen más tartalmú memóriaszegmensből olvas.

Mindkét ROM-ban ez áll:

FFF8 F1	POP	AF	A változatlan U0-beli veremből visszatölti az elmentett A regisztert és ugrás a HL mutatta címre
FFF9 E9	JP	HL	

Közben azonban átjutottunk a másik ROM-ba!

Még egyszer megjegyezzük, hogy az EXT ROM ugyanezen címein pontosan ugyanezek az utasítások állnak.

És közben, kedves Olvasó, szinte észrevétlenül hátunk mögött is hagytuk a ROM hatalmas BASIC területét!

Az EXT ROM-ban ugyan találkozunk még egy rövid listaközlés erejéig vele -- emlékezzünk vissza a PRINT utasítás USING alparancsának végrehajtása a ROM kiegészítésében van -- azonban sok dolgunk már ott sem lesz vele. Olyan részletkérdésként kezeljük csupán, mint tettük ezt a BASIC néhány segédrutinjával: széttagolt, szűkített listákat közöljük csupán, hiszen alig valószínű, hogy az Olvasónak éppen ezekre lesz valaha is közvetlenül szüksége, a maguk nemében annyira speciálisak, annyira egyedi funkciók szolgálatában állnak.

Úgy hogy: a BASIC-et lezártak tekinthetjük!

Éppen ezért érdemes lesz megállnunk egy szusszanásnyi időre -- az előző fejezetek anyagát kicsit nem árt magunkban rendezni.

Előtte azonban nem feledkezhetünk meg arról, hogy a fő ROM-szegmensből még hátra van néhány byte:

FFFA 79	LD	A,C	Ezeknek azonban láthatóan már
FFFB FE0D	CP	0D	semmi különösebb értelmes
FFFD 2813	JR	Z,0012	funkció nem tulajdonítható
FFFF FE00	CP	00	

(Az utolsó 00 byte már nem is a ROM, hanem az U0 első memóriarekesze, de a listákat feldolgozó program ezt éppen úgy nem vette észre, mint ahogy a CPU sem venné, ha mégis ezt a programrészt kellene végrehajtania. Az U0 első byte-jainak végrehajtása teljesen értelmetlen, míg a 0012H címről a ROM hibakezelőjébe ugrunk.)

3.6.11 Mögöttünk a BASIC

Mit lenne érdemes összefoglalásul a BASIC-ről mondani? Ha másként nem, talán többszöri átolvasás, de legalább átgondolás után az Olvasó is bizonyára látja, érzi a lényegét. Azt, amire a BASIC készült. Egy meglehetősen monumentális gépi kódú program, amely a monumentalitása mellett igyekszik -- amennyire egyáltalán lehet -- univerzális is lenni! És most hangsúlyozottan nem csak a TV-Computer BASIC-jéről van szó.

Mivel az alapvető célkitűzés is többszörösen összetett, az építmény maga is az.

Az egyik alapvető dolog az emberrel folytatott magasszintű kapcsolat. Ez úgy realizálódik, hogy a megoldandó feladatokat nem a CPU utasításkészletével, hanem emberközelibb módon, kulcsszavakkal fogalmazhatjuk meg - igaz nagyon szigorú formai és logikai kötöttségeket tudomásul véve és meglehetősen szűk szókinccsel gazdálkodva.

A másik alapvető dolog e szókészlet -- utasításkészlet -- elemeivel megalkotott mondatok lefordítása a CPU saját nyelvére. Ez már belső probléma, de láthattuk, hogy igazából nem túl megerőltető logikai, bár eléggé munkaigényes feladat.

A magas szintű párbeszéd nem lényegtelen tartozéka egy hatékony hibakezelő rendszer, amely a konkrét eredmények produkálásán túl képes bizonyos üzenetekkel is tájékoztatni az embert a működés körülményeiről, az esetleges hibákról. Tudjuk azonban, hogy sokszor a rendszer által hibásnak minősített állapot egyáltalán nem jelent tényleges hibát! Máskor pedig egy jó működésről informáló "ok" üzenet valójában katasztrofálisan mást jelent, mint amire számítottunk.

Semmi sem tökéletes!

A TV-Computer BASIC rendszere valamilyen szinten megoldja az alapvető célkitűzéseket. A gépben rejlő fizikai képességek túlnyomó részét kihasználja, aránylag könnyen és gyorsan és hatékonyan kezelhető, bizonyos szolgáltatásai imponálóan kellemesek. Természetesen nem mentes a hibáktól, több dolgot lehetett volna másként is megoldani.

Amit viszont látni kell: a TV-Computer BASIC-je egy program. Hibáival és erényeivel együtt. Láttuk, hogy még egy-egy viszonylag egyszerűbb részfeladat is mennyire megdolgoztatja a mikroprocesszort: a nagyon sok mindenre ki-

terjedő ellenőrzés, a korrekt és logikailag biztonságos működés természetesen minden magasabbszintű rendszer működését lelassítja - ezt is.

Kihasználva a BASIC környezet által biztosított kényelmes szolgáltatásokat, használjuk addig ezt a programot, ameddig a komplexitásából és a logikai rendszer megszabta korlátozásokból fakadó hátrányok nem múlják fölül az előnyöket. S ha igen, akkor hagyjuk el, vagy használjuk fel csak bizonyos részeit!

E könyv egyik célkitűzése éppen az, hogy ez utóbbira biztassa az Olvasót, hiszen felhasználói szempontból a BASIC használatához nem szükséges a BASIC-et realizáló program ismerete: ahhoz a BASIC alkalmazásait ismertető szakkönyvekre lehet szükség. Saját munkáinkban azonban egyáltalán nem kötelező, sokszor pedig nem is célszerű ezt a rendszert használni. Annál inkább egyes részeit!

Ez a könyv a TV-Computer gépi kódú alkalmazásait kívánja támogatni, elsősorban az ROM tartalom oldaláról. Ennek részeként jelentős helyet és időt kellett szentelni a BASIC működésének.

Ez a programnyelv a számítógépek történelmi jelentőségű fejlődésének egyik állomása. Egy jól-rosszul kialakult átmeneti állapot, az egyes géptípusokban még jobban vagy még rosszabbul megvalósítva! Abban a számítógép kategóriában, amit a TV-Computer képvisel, jelenleg mindenestre általánosan elterjedtnek mondható.

Ezzel együtt a BASIC a számítógépek és az ember közötti kommunikáció fejlődésének egy olyan állapota, amelyről tudomásul kell venni, hogy korunkban kétségtelen realitás.

Elég azonban éppen a személyi számítógépeknek ebben a kategóriájában igen elterjedt és nagy népszerűségnek örvendő játékprogramokra gondolni ahhoz, hogy lássuk: e gépek használatát jelentősen hatékonyabbá, ha úgy tetszik üzleti szempontból is sikeresebbé lehet tenni a BASIC kiiktatásával. Még akkor is, ha ezzel a logikailag alacsonyabb szintű vezérlés irányába tolódunk el.

A felhasználói programok interaktivitása, emberközelsége más módon is biztosítható, vagy a BASIC más, magas szintű nyelvre cserélhető.

Mindenképpen indokolt tehát a számvetés.

Nekünk azonban még jelentős további feladataink vannak. Vár ránk az EXT ROM további 4 kbyte-os programja.

Tartalmát nagyvonalakban már az eddigiek során megismertük. Itt fut az inicializáló rutin egy része, itt kaptak helyet a soros vonalak vagy pl. a magnetofont kezelő rutinok. Egy csomó olyan rész kérdést kell még megismernünk, ami a gép működésének megértéséhez elengedhetetlen, ugyanakkor saját programjainkban is jó szolgálatokat tehet.

3.7 Ami a rendszer ROM fő részéből kimaradt

Minden, ami az EXT ROM-ban található, valamilyen módon kapcsolódik a korábban már megtárgyalt részekhez.

Ez azzal jár, hogy itt nem töltünk már külön időt a szóban forgó részek működésének átfogó kifejtésével. Egyszerűen csak utalunk arra, hogy az EXT ROM éppen vizsgált területe logikailag a fő ROM melyik részéhez kapcsolódik.

Haladjunk sorban!

3.7.1 Adalék a rendszer inicializálásához

Az EXT ROM első néhány száz byte-ja a rendszerinicializálás része.

Amikor eldöntöttük, hogy meleg vagy hideg resetről van-e szó, s az utóbbi esetben elvégeztük a memóriaszegmensek tesztelését, akkor az EXT ROM eleje kapja meg a vezérlést, ahol az inicializálás a következőképpen folytatódik (a listákban az EXT ROM-ot a 3. lapra helyeztük, így a megfelelő címek az F000... FFFFH tartományba esnek):

F000 3ED0	LD	A, D0	U0-U1-VID-EXT lapozási
F002 D302	OUT	(02), A	konfigurációt állít be
F004 3A210E	LD	A, (0B21)	
F007 E7	OR	A	Ha meleg reset van, akkor
F008 2004	JR	NZ, F00E	ugrás előre,
F00A 08	EX	AF, AF	hideg resetnél azonban be
F00B 321B0E	LD	(0B1B), A	kell írni az U3-STAT
			rendszerváltozóba az U3
			jó/rossz jelzést
F00E 216BFE	LD	HL, FB6B	Ezután 32x10 byte-on az FB6B
F011 114007	LD	DE, 0740	területről az U0-ba másolja
F014 014001	LD	BC, 0140	az ékezetes betűk és a
F017 EDE0	LDIR		félgrafikus karakterek
			mátrixait

F019 3E40	LD A,40	A többi definiálható karakter
F01B E5	PUSH HL	helyére 64 * 10 byte 00-t ír,
F01C 0E0A	LD C,0A	ezáltal törölve a 0A0...0FDH
F01E EDB0	LDIR	ASCII kódokhoz tartozó
F020 E1	POP HL	területet
F021 3D	DEC A	
F022 20F7	JR NZ,F01B	
F024 210EFB	LD HL,FBOE	A 0FB0EH területről az U0-ba
F027 11000B	LD DE,0B00	átmásolva inicializálja az
F02A 0E10	LD C,10	input-output hozzárendelési
F02C EDB0	LDIR	táblát,
F02E 113000	LD DE,0030	
F031 0E10	LD C,10	majd a funkcióhívások és az IT
F033 EDB0	LDIR	alprogram bevezető és
F035 11230B	LD DE,0B23	befejező részét
F038 0E26	LD C,26	
F03A EDB0	LDIR	
F03C 3EFC	LD A,FC	A kurzor IT eszközei: video és
F03E 32100B	LD (0B10),A	billentyűzet
F041 3E80	LD A,80	A 06-os port kópiájában 2-es
F043 32130B	LD (0B13),A	üzemmód, 0 hangerő és
F046 07	RLCA	nyugalmi STROBE szint
F047 321F0B	LD (0B1F),A	beállítása és egyelőre csak
F04A 21100F	LD HL,0F10	kurzor IT engedélyezése
F04B 22170B	LD (0B17),HL	Végül a verem megengedett
		alsó határa

E beállítások után a csatolókartyákkal foglalkozunk.

F050 DB5A	IN A,(5A)	Beolvassuk a csatolók
F052 010304	LD BC,0403	azonosítóit (2-2 bit) és 4-es
F055 214000	LD HL,0040	ciklusban kezdjük a
F058 57	LD D,A	vizsgálatot
F059 3E04	LD A,04	
F05B 90	SUB B	E kivonás és az ezt követő
F05C 0F	RRCA	forgatások eredményeként A
F05D 0F	RRCA	értéke sorra 00-40-80-C0H
F05E D303	OUT (03),A	lesz, így a 03-as porttal a
		megfelelő csatolón levő
		memóriát választjuk ki
F060 3EFC	LD A,FC	A kurzor IT eszközei:
F060 7A	LD A,D	
F061 A1	AND C	C=03, tehát csak b1-b0 kell.
F062 FE03	CP 03	Az elforgatások miatt ez is
F064 E5	PUSH HL	mindig a kiválasztott
F065 D9	EXX	kartyákhoz tartozik

F066 2811	JR	Z,F079	Ha üres, vagy nem azonosított kártya: ugrás!
F068 215CFB	LD	HL,FB5C	A soros vonal nevének címe. Ha tényleg erről van szó: mehet előre
F06B B7	OR	A	
F06C 281B	JR	Z,F089	
F06E 2158FB	LD	HL,FB58	A játékmodul-azonosító címe. Ha A=01, tehát ez van bedugva: mehet
F071 3D	DEC	A	
F072 2815	JR	Z,F089	
F074 2162FB	LD	HL,FB62	Ezután már csak floppy csatoló lehet
F077 1810	JR	F089	

HL tehát az azonosított kártyához tartozó név címére mutat.

Megvizsgáljuk a nem azonosított kártyákat:

F079 2100C0	LD	HL,C000	Amennyiben ezek működőképes eszközök, akkor a memóriájuk elején a MOPS szót kell tartalmazniuk
F07C 1154FB	LD	DE,FB54	
F07F 0604	LD	B,04	
F081 1A	LD	A,(DE)	Az azt követő byte-okon pedig a kártya nevének kell állnia
F082 BE	CP	(HL)	
F083 200D	JR	NZ,F092	Itt ellenőrizzük a MOPS szó jelenlétét Ha nincs: a csatoló üres
F085 13	INC	DE	
F086 23	INC	HL	
F087 10F8	DJNZ	F081	

Az üres csatoló kivételével az inicializálás most már közösen azzal folytatódik, hogy az egyes csatolók számára fenntartott, U0 RAM-beli 0040H...0070H...00A0H...00D0H pufferek elejére betöltjük az azonosító nevét.

F089 4E	LD	C,(HL)	HL az EXT ROM-ban a névre mutat, BC-ben a névhossz, DE a puffercím, lehet a nevet bemásolni, majd ugrás előre
F08A 0C	INC	C	
F08B 0600	LD	B,00	
F08D D1	POP	DE	
F08E EDB0	LDIR		
F090 1803	JR	F095	

Azonosítatlan és a MOPS szót sem tartalmazó kártyák:

F092 E1	POP	HL	A puffer elejére csak egy 0FFH kerül
F093 36FF	LD	(HL),FF	

Ezzel egy-egy kártya kész.

Előkészítjük a ciklusban a következő csatolóhely feldolgozását:

F095 D9	EXX	
F096 7A	LD A,D	Az 5AH-s portról beolvasott
F097 0F	RRCA	azonosítók bitjeit 2-vel
F098 0F	RRCA	tovább forgatjuk, s ráállunk a
F099 113000	LD DE,0030	következő pufferre
F09C 19	ADD HL,DE	
F09D 10B9	DJNZ F058	

Ezután külön foglalkozunk a soros vonallal.

709F 0604	LD B,04	Ismét 4-es ciklus
70A1 57	LD D,A	Vesszük ismét a bekötött 2
70A2 A1	AND C	bites azonosítót, mindig csak
70A3 200B	JR NZ,70E0	b1-b0 kell
70A5 3E04	LD A,04	00 lenne a soros vonal, ha nem
70A7 90	SUB B	az: mehet tovább
70A8 32070B	LD (0B07),A	Ha soros vonalat talált, a
70AB 320F0B	LD (0B0F),A	megfelelő csatoló számát
		bejegyzzi az I/O hozzárendelési
		tábla utolsó rekeszeibe és
70AE 1807	JR 70B7	ugrás előre

Ha nem soros vonalat talál, továbblép:

70B0 7A	LD A,D	Az alsó bitekbe a következő
70B1 0F	RRCA	kártya azonosítót mozgatja és
70B2 0F	RRCA	ugrás vissza
70B3 10EC	DJNZ 70A1	Ha egyik helyen sem soros vonal
70B5 3EFF	LD A,FF	van: A=FF
70B7 321C0B	LD (0B1C),A	Az első talált soros vonal
		számát vagy 0FFH-t ír az
		alaphozzárendelés rendszervál-
		tozójába

Adminisztrálni kell még az I/O pufferek 7. relatív című rekeszeiben az azonos típusú kártyák egység számait.

70BA 214700	LD HL,0047	A cím a 0. csatoló pufferében
70BD 113000	LD DE,0030	
70C0 0604	LD E,04	
70C2 AF	XOR A	Először minden egység szám: 0
70C3 77	LD (HL),A	
70C4 19	ADD HL,DE	
70C5 10FC	DJNZ 70C3	

70C7 1E40	LD E,40	DE=0040 a 0. csatolóhoz,
70C9 217000	LD HL,0070	HL az 1. csatolóhoz tartozó
70CC D5	PUSH DE	pufferre mutat
70CD DDE1	POP IX	IX pedig arra, amellyel a többi
70CF E5	PUSH HL	összehasonlítjuk, míg IY
70D0 FDE1	POP IY	végigfut a többi pufferen
70D2 46	LD B,(HL)	
70D3 04	INC B	Először ellenőrzi, hogy a két
70D4 1A	LD A,(DE)	bejegyzett hossz egyezik-e, ha
70D5 BE	CP (HL)	már ez sem: ugrás tovább!
70D6 2007	JR NZ,70DF	
F0D8 23	INC HL	Ha a hossz megegyezik,
F0D9 13	INC DE	ellenőrizzük a név karaktereit.
F0DA 10F8	DJNZ F0D4	Ha azonosak: növeljük az
F0DC FB3407	INC (IX+07)	egységszámot
F0DF FDE5	PUSH IY	
FOE1 E1	POP HL	Ezután IY-nal ráállunk a
FOE2 113000	LD DE,0030	következő puffer elejére
FOE5 19	ADD HL,DE	
FOE6 CB44	BIT 0,H	Ha még van összehasonlíthatlan
FOE8 280C	JR Z,FOF6	puffer: ugrás
FOEA DDE5	PUSH IX	Ha pedig az IX mutatta pufferrel
FOEC E1	POP HL	már mindegyiket összehasonlí-
FOED 19	ADD HL,DE	tottuk, akkor ezzel is
FOEE CB44	BIT 0,H	továbblépünk
FOF0 2009	JR NZ,FOFB	Ha már az utolsó is megvan,
FOF2 E5	PUSH HL	akkor ugrás előre,
FOF3 DDE1	POP IX	egyébként folytatjuk a
FOF5 19	ADD HL,DE	vizsgálatot
FOF6 DDE5	PUSH IX	
FOF8 D1	POP DE	
FOF9 18D4	JR F0CF	

Végül hátra van még a bővítőkétyákkal csatlakoztatott eszközök inicializálása:

FOFB 3EFO	LD A,F0	U0-U1-U2-EXT memórialapozást
FOFD D302	OUT (02),A	állít be
FOFF CDE8FF	CALL FFEB	Inicializálja a soros vonalat,
		ez a rutint nemsokára megnézzük
F102 DD214000	LD IX,0040	A többi eszköz rutinját a saját
F106 0604	LD B,04	memóriájának kell tartalmaznia,
F108 DB5A	IN A,(5A)	ezt ciklusban végeztetjük el
F10A 4F	LD C,A	A bekötött azonosítók

F10B 3E04	LD	A,04	
F10D 90	SUB	B	A szokásos eljárással
F10E 0F	RRCA		kiválasztjuk a 3. lap elejére
F10F 0F	RRCA		az éppen kezelt kártya
F110 D303	OUT	(03),A	memóriáját
F112 DD7E00	LD	A,(IX+00)	Ha a puffer alapján
F115 3C	INC	A	üres: érdektelen,
F116 2815	JR	Z,F12D	ugrás tovább!
F118 79	LD	A,C	Ha az azonosító alapján:
F119 E603	AND	03	soros vonal, akkor pedig már
F11B 2810	JR	Z,F12D	kész, mehet tovább!

A többi eszköznek a saját memóriája 0C00BH címén kell tartalmaznia az inicializáló rutinja címét (szigorúan!).

F11D 2A0BC0	LD	HL,(C00B)	HL=az inicializáló rutin címe
F120 C5	PUSH	BC	
F121 DDE5	PUSH	IX	
F123 CDF9FF	CALL	FFF9	Ez egy JP HL, vagyis elvégezzük
F126 DDE1	POP	IX	az eszköz inicializálását
F128 C1	POP	BC	
F129 3EFO	LD	A,F0	A biztonság kedvéért
F12B D302	OUT	(02),A	visszaállítjuk a korábbi
F12D 113000	LD	DE,0030	memórialapozást és vesszük a
F130 DD19	ADD	IX,DE	következő puffert, ill. b1-b0-
F132 79	LD	A,C	ba mozgatjuk a következő
F133 0F	RRCA		kártya azonosító bitjeit
F134 0F	RRCA		Folytatjuk a vizsgálatot
F135 10D3	DJNZ	F10A	
F137 21D3C2	LD	HL,C2D3	A már megismert "hídon"
F13A C3FOFF	JP	FFF0	keresztül a fő ROM C2D3H címén
			folytatódik az inicializálás

Az EXT ROM következő, rövid kis rutinja szintén az inicializálás alatt kapja meg a vezérlést. Feladata a reset körülményeinek tisztázása és további sorsának eldöntése az U0 épségének ellenőrzésével és a két reset FLAG rendszerváltozó alapján:

F13D 08	EX	AF,AF	A fő ROM-ból hozott A=C0
			értéket tárolja
F13E 3ED0	LD	A,D0	U0-U1-VID-EXT konfigurációt
F140 D302	OUT	(02),A	állít be
F142 112EFB	LD	DE,FB2E	A 0FB2EH címen kezdődő minta
F145 21230B	LD	HL,0B23	alapján ellenőrzi a funkció-
F148 011E00	LD	BC,001E	hívásokkal foglalkozó program-
F14B 1A	LD	A,(DE)	rész épségét az U0 RAM-ban
F14C 13	INC	DE	

F14D EDA1	CPI		Ha nem egyezik: ugrás előre!
F14F 200B	JR	NZ,F15C	
F151 EA4BF1	JP	PE,F14B	
F154 3A210B	LD	A,(0B21)	Ha az U0 vizsgált része jó és
F157 B7	OR	A	korábban még nem volt meleg
F158 3EFF	LD	A,FF	reset állapot, akkor most
F15A 2801	JR	Z,F15D	ezzel a jelzéssel megy tovább:
			A=FF: meleg reset lehet
F15C AF	XOR	A	Ha azonban már eleve meleg
			reset állapot van (pl a RESET
			gombot 2-szer is megnyomták),
			akkor hideg reset kell!
F15D 32210B	LD	(0B21),A	A megfelelő jelzést beállítja
F160 08	EX	AF,AF	a WARM-FLAG-ben, és az A'-ben
F161 D302	OUT	(02),A	is visszaadja. Ezzel egyúttal
F163 C33A02	JP	023A	A=C0 lesz, így visszaál-
			líthatja az eredeti lapozást:
			SYS-U1-VID-EXT és ugrás a fő
			ROM-ba

3.7.2 Adalék a funkcióhívásokhoz

Az ezután következő rész a funkcióhívások lebonyolításához kapcsolódik. Amennyiben az derült ki, hogy valamely funkcióosztályhoz közvetlenül egy csatolókártya van hozzárendelve, akkor a funkció kiszolgálását az itt következő programrész vezérli.

Az első belépési pontra akkor lép a rendszer, ha közvetlen hozzárendelés van, azaz a hozzárendelési táblázatban az adott osztályhoz tartozó memóriarekeszbe b7=1 bejegyzéssel a csatoló számát írták. Belépéskor az A regiszterben már csak ez a szám van, a B-ben a kijelölt funkcióosztály száma, DE pedig az in- vagy out-táblázat elejére mutat.

F166 68	LD	L,B	
F167 2600	LD	H,00	HL-be a kijelölt
F169 19	ADD	HL,DE	funkcióosztály táblázatbeli
F16A 1805	JR	F171	címét teszi és ugrás

Itt a másik belépési pont. Ha az osztályt kiszolgáló eszközként 06-os eszközszámmal csatolókátyát jelöltek meg, akkor a táblázatok utolsó (kernelhez tartozó) rekeszeibe kellett tölteni a csatlakozó számát. Ezen a belépési címen ezt dolgozzuk fel:

F16C 210700	LD	HL,0007	Az utolsó táblaelemből
F16F 19	ADD	HL,DE	kiolvassuk a csatlakozó számát
F170 7E	LD	A,(HL)	

Ettől kezdve a végrehajtás közös.

F171 FE04	CP	04	Ha 3-nál nagyobb számot adtak
F173 3064	JR	NC,F1D9	meg: hiba
F175 D9	EXX		
F176 214000	LD	HL,0040	
F179 113000	LD	DE,0030	
F17C 47	LD	B,A	
F17D 0F	RRCA		A csatoló számát b7-b6-ba
F17E 0F	RRCA		visszük és belapozzuk a
F17F 32110B	LD	(0B11),A	kiválasztott kártyán levő
F182 D303	OUT	(03),A	memóriát
F184 78	LD	A,B	A csatoló száma alapján
F185 B7	OR	A	ráállunk a hozzá tartozó U0 RAM
F186 2803	JR	Z,F18B	puffer elejére
F188 19	ADD	HL,DE	(0040...70...A0...D0H)
F189 10FD	DJNZ	F188	
F18B 7E	LD	A,(HL)	Ha az első byte-on 0FFH áll
F18C 3C	INC	A	(üres kártyahely), akkor hiba!
F18D 2849	JR	Z,F1D8	
F18F 4E	LD	C,(HL)	
F190 23	INC	HL	
F191 115DFB	LD	DE,FB5D	A soros vonal azonosító címére
F194 1A	LD	A,(DE)	állva ellenőrzi, hogy a
F195 13	INC	DE	kiválasztott eszköz soros
F196 EDA1	CPI		vonal-e?
F198 2009	JR	NZ,F1A3	Ha nem: ugrás előre
F19A EA94F1	JP	PE,F194	
F19D D9	EXX		
F19E 2191F2	LD	HL,F291	Ha igen: rááll a soros vonal
F1A1 1804	JR	F1A7	ugrótáblájára

A többi eszköznek funkcióhívások kiszolgálásához szükséges ugrótáblát kötelezően a saját memóriájuk 0C00DH címén kell tartalmazniuk.

F1A3 D9	EXX		
F1A4 210DC0	LD	HL,000D	
F1A7 7E	LD	A,(HL)	Az első byte a hívható funkciók
F1A8 3D	DEC	A	száma
F1A9 B9	CP	C	Ha a kért rutinsorszám túl
F1AA 3834	JR	C,F1E0	nagy: hiba!

```

F1A0 23      INC  HL
F1A1 EB      EX   DE,HL
F1A2 69      LD   L,C
F1A3 2600    LD   H,00
F1B1 29      ADD  HL,HL
F1B2 19      ADD  HL,DE
F1B3 5E      LD   E,(HL)
F1B4 23      INC  HL
F1B5 56      LD   D,(HL)
F1B6 EB      EX   DE,HL
F1B7 3A110B  LD   A,(0B11)
F1BA 07      RLCA
F1BB 07      RLCA
F1BC E603    AND  03
F1BE DD214800 LD  IX,0048
F1C2 113000  LD  DE,0030
F1C5 2805    JR   Z,F1CC
F1C7 47      LD   B,A
F1C8 DD19    ADD  IX,DE
F1CA 10FC    DJNZ F1C8
F1CC 08      EX  AF,AF
F1CD D1      POP DE
F1CE C1      POP BC
F1CF CDF9FF  CALL FFF9
F1D2 21FCC3  LD  HL,C3FC
F1D5 C3FOFF  JP  FFF0

```

Ha a hívott rutin létezik, HL-ben előállítjuk az ugrótábla adott sorszámú címét, majd ebből a végrehajtó rutin címét

Ezután a 03-as port tárolt kópiája alapján a b7-b6 bitet a b1-b0-ba mozgatva ráállunk az adott számú csatolóhoz tartozó U0-beli puffer adat-területére

Visszatöltjük a funkcióhívás paramétereit hordozó regisztereket és végrehajtjuk a feladatot, majd ugrás vissza a fő ROM-ba

Hibakezelés:

üres kártyahely esetén először visszaállítjuk a felcserélt regisztereket:

```
71D8 D9      EXX
```

Ha túl nagy csatolószaómot adtak meg, elég itt belépni:

```

F1D9 3A1C0B  LD   A,(0B1C)
F1DC 77      LD   (HL),A
F1DD 3EFE    LD   A,FE
F1DF 113EFF  LD   DE,FF3E

```

Visszaáll a bővítőkártya alaphozzárendelés I/O hozzárendelési hiba jelzése

Ez itt érdektelen utasítás, csak arra való, hogy az A=FE értéket a CPU ne változtassa meg.

Ha ugyanis az eszközre megengedettnél nagyobb funkciószáómot adtak meg, akkor az ugrás a 71E0H címre, tehát az előbbi utasítás belsejébe történik, ahol a CPU számára ez áll:

```
F1E0 3EFF    LD   A,FF      Nem létező hívási kód jelzése
```

```

F1E2 D1      POP  DE
F1E3 C1      POP  BC
F1E4 18EC    JR   F1D2

```

Ezután a regisztereket, visszatöltve, a hibakóddal ugrás a fő ROM-ba

Hibára vezető funkcióhívások esetén a rendszer általában visszaállítja az alaphozzárendeléseket. Ez a rutin is az EXT ROM-ban kapott helyet és működése a következő:

```

F1E6 07      RLCA
F1E7 21000B  LD   HL,0B00
F1EA 110EFB  LD   DE,FB0E
F1ED DD21070B LD   IX,0B07
F1F1 380A    JR   F,71FD
F1F3 21080B  LD   HL,0B08
F1F6 1116FB  LD   DE,FB16
F1F9 DD210F0B LD   IX,0B0F
F1FD 07      RLCA
F1FE 07      RLCA
F1FF 07      RLCA
F200 E607    AND  07
F202 FE07    CP   07
F204 2812    JR   Z,F218
F206 4F      LD   C,A
F207 0600    LD   B,00
F209 09      ADD  HL,BC
F20A EB      EX  DE,HL
F20B 09      ADD  HL,BC
F20C EB      EX  DE,HL
F20D 7E      LD   A,(HL)
F20E FE06    CP   06
7210 CC1EF2  CALL Z,F21E

F213 07      RLCA
F214 3802    JR   C,F218
F216 1A      LD   A,(DE)
F217 77      LD   (HL),A

F218 210EC4  LD   HL,C40E
F21B C3F0FF  JP   FFF0

```

Most A-ban az eredeti funkciókód áll, tehát a bitvizsgálattal az I/O irányt tisztázzuk

Ezt figyelembe véve végül HL=a tábla eleje az U0-ban, DE= az eredeti hozzárendeléseket tartalmazó táblázat az EXT ROM-ban, IX az utolsó rekeszre mutat, A-ban pedig a funkcióosztály

Ha kernel hívás volt, ezzel semmit sem kell csinálni

Egyébként mindkét táblázatban a kijelölt funkcióosztály címére áll

Ha bővítőkartya kijelölés volt, visszaállítja az alaphozzárendelést, közvetlen hozzárendelés esetén azonban nem

A többi esetben az osztály rekeszébe az inicializált értéket tölti

Utána ugrás vissza a fő ROM-ba

A csatolókarták alaphozzárendelésének visszaállítása:

```

F21E 08      EX  AF,AF
F21F 3A1C0B  LD   A,(0B1C)
F222 DD7700  LD   (IX+00),A
F225 08      EX  AF,AF
F226 C9      RET

```

Az alaphozzárendelés értékét tölti az I/O táblázat utolsó rekeszébe

3.7.3 Adalék az IT alprogramhoz

A következő rutin a megszakításkezelő alprogram része. Feladata a csatolókarták IT kéréseinek kiszolgálása.

Belépéskor a C regiszter b3...b0 bitje a karták IT kéréseit tartalmazza, b=0 érték jelzi az aktivitást.

F227 0604	LD B,04	Ciklusban vizsgáljuk
F229 CB19	RR C	Ha a vizsgált kártya nem aktív:
F22B 3813	JP C,F240	ugrás előre
F22D C5	PUSH BC	
F22E 3F04	LD A,04	Aktív kérés esetén a szokásos
F230 90	SUB B	módon behozzuk a vizsgált
F231 0F	PRCA	kártyán levő memóriát
F232 0F	RRCA	
F233 D303	OUT (03),A	Ebben az IT rutin címét
F235 2A0E00	LD HL,(000E)	kötelezően a 0C00EH címen kell
F238 CDF9FF	CALL FFF9	elhelyezni
F23B 3EF0	LD A,F0	Meghívjuk az eszköz IT
F23D D302	OUT (02),A	rutinját, helyreállítjuk a
F23F C1	POP BC	memórialapozást és jöhet a
7240 10E7	DJNZ F229	következő
F242 3A110B	LD A,(0B11)	Az összes kártya kiszolgálása
F245 D303	OUT (03),A	után az eredeti értéket írja a
F247 E1	POP HL	portra, majd vissza a fő ROM-
F248 C3F0FF	JP FFF0	ba!

3.8 Még két eszközmeghajtó

3.8.1 Bevezetés

Az EXT ROM első fő részén túl vagyunk.

Még két eszköz teljes szoftvere, a soros vonalat és a magnetofont kezelő rutinok maradtak hátra.

A soros vonal vezérlése az USART-tal kapcsolatban a 2. fejezetben megismert lehetőségek alapján nem túlságosan komplikált feladat.

Nem úgy a magnetofon vezérlése, amelyről mindeddig szinte csak annyit mondhattunk, hogy vele kapcsolatban -- objektív okokból -- mindent itt, a számítógépben kell megoldani.

A programozók az EXT ROM-ban is elhelyezték azt a két szubrutint, amely a blokkos adatforgalmat vezérli az I/O műveletek során. A rutinok ellenőrzik a legmagasabb használható memóriacímet, és ismételten hívják a karakteres I/O rutint. Nem ismételjük meg az ezzel kapcsolatban leírtakat (3.5.3.1 szakasz), s a működést sem részletezzük újra.

Annyit azért felidézünk, hogy belépéskor a BC és DE regiszterpárok a funkcióhívásuknál megszokott paramétereket tartalmazzák (hossz és memóriacím), míg a HL regiszterpárban a karakteres I/O műveletet megvalósító rutin kezdőcímét kell elhelyezni.

A legmagasabb memóriacím átlépésétől nekünk magunknak kell megóvni a rendszert! Ellenkező esetben nagyon kínos következményekkel számolhatunk.

A teljesség kedvéért természetesen itt is közöljük a két szubrutin listáját, azonban minden megjegyzés nélkül, a már korábban is használt, tömörített formában.

Blokkos output HIMEM átlépés
ellenőrzéssel

```
F24B  EX  DE,HL
F24C  PUSH HL
F24D  PUSH DE
F24E  PUSH BC
F24F  PUSH HL
F250  PUSH DE
F251  EX  DE,HL
F252  LD  HL,(0B19)
F255  OR  A
F256  SBC HL,DE
F258  LD  A,FA
F25A  POP  DE
F25B  POP  HL
F25C  RET  C
F25D  LD  C,(HL)
F25E  EX  DE,HL
F25F  CALL FFF9
F262  POP  BC
F263  POP  DE
F264  POP  HL
F265  OR  A
F266  RET  NZ
F267  CPI
F269  RET  FO
F26A  JR  F24C
```

Blokk input HIMEM átlépés
ellenőrzéssel

```
F26C  PUSH HL
F26D  PUSH DE
F26E  PUSH BC
F26F  PUSH HL
F270  PUSH DE
F271  LD  HL,(0B19)
F274  OR  A
F275  SBC HL,DE
F277  LD  A,FA
F279  POP  DE
F27A  POP  HL
F27B  RET  C
F27C  CALL FFF9
F27F  EX  AF,AF'
F280  LD  A,C
F281  POP  BC
F282  POP  DE
F283  POP  HL
F284  EX  AF,AF'
F285  OR  A
F286  RET  NZ
F287  EX  AF,AF'
F288  LD  (DE),A
F289  XOR  A
F28A  EX  DE,HL
F28B  CPI
F28D  EX  DE,HL
F28E  RET  FO
F28F  JR  F26C
```

3.8.2 A soros vonal

Erről a könyv korábbi fejezeteiben már sok szó esett, itt gyorsan a működés lényegi elemzésébe kezdhetünk.

Ez az egyetlen olyan külső eszköz, amelynek működtető programja a számítógép ROM-jában kapott helyet. A struktúra teljesen megegyezik az összes többi eszköznél megismerttel. A rendszer egy ugrótábla révén éri el a meghatározott feladatokat tényleges megvalósító rutinokat.

Az ugrótábla után a soros vonal inicializáló szubrutinja kezdődik.

A 0. funkcióhoz tartozó program itt is egyetlen RET utasításból áll.

3.8.2.1 A soros vonal ugrótáblája

ROM	Tartalom	Jelentés
F291	05	A hívható funkciók száma
F292	F305	0. funkcióhoz tartozó rutin címe
F294	F339	1. funkció (SER-CHIN/OUT)
F296	F3B7	2. funkció (SER-BKIN/OUT)
F298	F2A4	3. funkció (SER-SET)
F29A	F357	4. funkció (NOP)

3.8.2.2 Rutinok

A rendszer az inicializálás során A=04 értékkel hívja meg a következő rutint:

F29C	32690B	LD	(0B69),A	BAUD, a rendszerváltozó 04 értéke 1200 bit/s átviteli sebességet jelent
F29F	3EEE	LD	A,EE	Az üzemmód-paraméterek inicializálása
F2A1	326A0B	LD	(0B6A),A	

A program további része már megegyezik a funkcióhívással elérhető SER-SET rutinnal.

Soros vonal 03 SER-SET (serial line set)
63H, E3H A soros vonali USART alaphelyzetbe hozása

F2A4	C5	PUSH	BC	
F2A5	E5	PUSH	HL	
F2A6	3A130B	LD	A, (0B13)	A 06-os port kópiája
F2A9	E6C3	AND	C3	b5...b2=0: hangerőbitek, a soros vonal működése közben hang nem hallható
F2AB	32130B	LD	(0B13),A	
F2AE	D306	OUT	(06),A	
F2B0	3A690B	LD	A, (0B69)	BAUD, a soros vonal sebességére beírt értékre legfeljebb 08-at enged meg
F2B3	FE09	CP	09	
F2B5	3802	JR	C, F2B9	
F2B7	3E08	LD	A, 08	
F2B9	6F	LD	L, A	
F2BA	2600	LD	H, 00	
F2BC	29	ADD	HL, HL	A kód kétszerese, s a hozzá tartozó hangfrekvenciákat tartalmazó táblázatból ennek alapján DE-be tölti a sebességhez tartozó adatot
F2BD	11C6F3	LD	DE, F3C6	
F2C0	19	ADD	HL, DE	
F2C1	5E	LD	E, (HL)	
F2C2	23	INC	HL	
F2C3	56	LD	D, (HL)	

F2C4 7A	LD	A,D	A PITCH felső byte-ja,
F2C5 E60F	AND	0F	csak b3...b0 lehet
F2C7 F610	OR	10	A hangjel engedélyezése
F2C9 57	LD	D,A	
F2CA 3A120B	LD	A,(0B12)	A 05-ös port kópiájában b7 és
F2CD E6C0	AND	C0	b6 nem változhat
F2CF B2	OR	D	Hozzámaszkoljuk az előbbieket
F2D0 32120B	LD	(0B12),A	és érvényesítjük a beállított
F2D3 D805	OUT	(05),A	értékeket
F2D5 7B	LD	A,E	PITCH alsó byte, ezzel a
F2D6 D804	OUT	(04),A	hanggenerátor beállítása kész
F2D8 3A6A0B	LD	A,(0B6A)	FORMAT, a beírt értékből csak
F2DB E6B4	AND	B4	b7, b5, b4 és b2 lesz
F2DD F64A	OR	4A	figyelembevétel, b6, b3 és b1
F2DF 67	LD	H,A	lesz, b0=0
F2E0 011104	LD	BC,0411	4-es ciklus
F2E3 110340	LD	DE,4003	
F2E6 DB5A	IN	A,(5A)	A kártyaazonosítók,
F2E8 6F	LD	L,A	L-ben tároljuk
F2E9 A3	AND	E	Csak b1 és b0 kell
F2EA 200A	JR	NZ,F2F6	Ha nem soros vonal: ugrás előre
F2EC ED79	OUT	(C),A	C sorra a 11, 21, 31 és 41H
F2EE ED79	OUT	(C),A	portcímekeket veszi fel
F2F0 ED79	OUT	(C),A	
F2F2 ED51	OUT	(C),D	D=40: belső reset
F2F4 ED61	OUT	(C),H	üzemmód-kiválasztás
			(alapérték: 0EEH)
F2F6 3E10	LD	A,10	
F2F8 81	ADD	A,C	A következő csatolóhoz tartozó
F2F9 4F	LD	C,A	port cím,
F2FA 7D	LD	A,L	
F2FB 0F	RRCA		ill. annak azonosító bitjei
F2FC 0F	RRCA		b1-re és b0-ra
F2FD 10E9	DJNZ	F2E8	

Az üzemmód-kiválasztó parancs után még egy adás-vétel engedélyezés parancsot is ki lehetett volna küldeni (C portra 05).

F2FF AF	XOR	A	"Az órajel frekvenciája jó" --
F300 32710B	LD	(0B71),A	ezt jelzi
F303 E1	POP	HL	
F304 C1	POP	BC	
F305 C9	RET		

A 0F305H címen található RET utasítás egyben a 0. funkcióhoz tartozó szubrutin egyetlen utasítása is.

Három segédrutin következik.

S1 Megvárja az esetleg folyamatban levő hang végét
F306H és rááll a megfelelő port címre

in: C=egy kiküldendő karakter kódja
out: carry=1, sikeres előkészítés, ekkor:
- C=portcím;
- H=karakterkód;
- A=00
carry=0, sikertelen előkészítés
- A=F5 : CTRL+ESC volt

F306 CD31F3	CALL F331	CTRL+ESC vizsgálata
F309 C0	RET NZ	STOP esetén visszatér
F30A 3A140B	LD A,(0B14)	Hang van folyamatban?
F30D B7	OR A	Ha igen: ugrás vissza,
F30E 20F6	JR NZ,F306	várakozik!
F310 61	LD H,C	A kódot áttölti
F311 3A110B	LD A,(0B11)	A 03-as port kópiája alapján a
F314 0F	RRCA	kiválasztott kártyához tartozó
F315 0F	RRCA	port címét állítja elő:
F316 E630	AND 30	
F318 C611	ADD A,11	A=11, 21, 31 vagy 41H a csatoló
F31A 4F	LD C,A	száma szerint
F31B AF	XOR A	Végül A=00,
F31C 3F	CCF	carry=0 és
F31D C9	RET	kész
F31E 00	NOP	

S2 Az USART adásra kész állapotát várja
F31EH

in: C=a kiküldendő karakter kódja
out: carry=1, ha az USART munkára kész. Ekkor:
- H=karakterkód
- A=81, a megfelelő állapot két bitje
- C=portcím (állapot/parancskezelés)
carry=0, a megfelelő állapot nem volt kivártható
- A=F5 CTRL+ESC-t nyomtak

F31F CD0AF3	CALL F30A	
F322 C0	RET NZ	Ha STOP volt, visszatér
F323 ED78	IN A,(C)	Az USART állapota
F325 E681	AND 81	Csak b7 és b0 kell
F327 FE81	CP 81	Az USART kész?
F329 3F	CCF	Ha igen: CY=1 és visszatér,
F32A D8	RET C	egyébként
F32B CD31F3	CALL F331	CTRL+ESC vizsgálata,
F32E 28F3	JR Z,F323	s ha nem volt STOP, tovább vár,
F330 C9	RET	egyébként RET

S3 CTRL+ESC kezelése
F331H

F331	3A160B	LD	A, (0B16)	STOP-FLAG
F334	B7	OR	A	Volt CTRL+ESC?
F335	C8	RET	Z	Ha nem, A=00-val RET,
F336	3EF5	LD	A, F5	egyébként a hibakóddal tér
F338	C9	RET		vissza

Ezekután az RS232 soros vonal funkcióhívásaihoz tartozó rutinok következnek.

Soros vonal 01 SER-CHOUT/IN
61H, E1H Karakter kiküldés (61H)
Karakter beolvasás (E1H)
in: C=karakterkód
out: A=00 sikeres
hibakód

F339	FA59F3	JP	M, F359	Input műveletnél ugrás előre
------	--------	----	---------	------------------------------

Itt a karakter kiküldés rutinja folytatódik.

F33C	CD1FF3	CALL	F31F	Az USART kész?
F33F	D0	RET	NC	Ha nem: visszatérés
F340	3A710B	LD	A, (0B71)	Az órafrekvencia jó?
F343	B7	OR	A	Ha nem megfelelő, meghívja az
F344	C4A4F2	CALL	NZ, F2A4	inicializáló rutint
F347	3E05	LD	A, 05	Adás-vétel engedélyezés
F349	ED79	OUT	(C), A	parancsot küld ki, majd STOP
F34B	CD31F3	CALL	F331	vizsgálattal megvárja az USART
F34E	C0	RET	NZ	adásra kész állapotát, amit
F34F	ED78	IN	A, (C)	b=1 jelez
F351	0F	RRCA		
F352	30F7	JR	NC, F34E	Amíg nem kész: vár!
F354	0D	DEC	C	Rááll az adatport címre és
F355	ED61	OUT	(C), H	kiküldi a karaktert, majd
F357	AF	XOR	A	"sikeres" jelzéssel visszatér
F358	C9	RET		

A karakterbeolvasás rutinja

F359	CDOAF3	CALL	F30A	Kivárja az esetleges hangképzés végét és elkészíti a portcímet
------	--------	------	------	--

F35C	D0	RET	NC	CTRL+ESC esetén RET
F35D	3A710B	LD	A, (0B71)	Az órajel megfelelő?
F360	B7	OR	A	Ha nem, meghívja az
F361	C4A4F2	CALL	NZ, F2A4	inicializáló rutint
F364	ED78	IN	A, (C)	Beolvassa az USART állapotát:
F366	0F	RRCA		b1=1, ha van vett és
F367	0F	RRCA		kiolvasható karakter
F368	3006	JR	NC, F370	Ha nincs: ugrás előre
F36A	0D	DEC	C	
F36B	ED78	IN	A, (C)	A vett karaktert beolvassa és
F36D	4F	LD	C, A	ilyenkor
F36E	AF	XOR	A	kész is
F36F	C9	RET		

Ha nincs az USART pufferében korábban vett és beolvasásra váró karakter, akkor a beolvasásról külön intézkedni kell.

F370	F3	DI		A megszakítások tiltása után a
F371	3A110B	LD	A, (0B11)	03-as port-kópia alapján csak
F374	E6F0	AND	F0	a CTRL+ESC vizsgálatára
F376	F607	OR	07	készülünk fel, az USART-nak
F378	D303	OUT	(03), A	pedig kiadjuk az adás-vétel és
F37A	3E25	LD	A, 25	az ellenállomás adását
F37C	ED79	OUT	(C), A	engedélyező parancsot
F37E	DB58	IN	A, (58)	Ezután figyeljük a
F380	E618	AND	18	billentyűzeten a CTRL+ESC
F382	2009	JR	NZ, F38D	lenyomását, s ha ezt
F384	21620B	LD	HL, 0B62	tapasztaljuk, a PICTURE-ben
F387	CBDE	SET	3, (HL)	való adminisztrálás után a
F389	3EF5	LD	A, F5	hibakóddal visszatérés.
F38B	FB	EI		
F38C	C9	RET		
F38D	ED78	IN	A, (C)	Másrészt figyeljük, hogy az
F38F	0F	RRCA		USART a b1=1 állapotbittel
F390	0F	RRCA		mikor jelez vett karaktert.
F391	30EB	JR	NC, F37E	Várakozunk
F393	3E05	LD	A, 05	Karakter vétele után ismét
F395	ED79	OUT	(C), A	kiadjuk az adás-vétel
F397	ED78	IN	A, (C)	engedélyezése parancsot,
F399	0D	DEC	C	beolvassuk az állapotbyte-ot
F39A	ED60	IN	H, (C)	A-ba és a vett karaktert is, H-
F39C	FB	EI		ba

A beolvasott állapotbyte alapján a karaktervétel körülményeinek vizsgálatát kell még elvégezni.

F39D	E638	AND	38	Csak a b5, b4 és b3 kell
F39F	2814	JR	Z, F3B5	Ha nincs hiba, A=00-val ugrás
				a végére

F3A1 47	LD B,A	A hibabitekett nézzük, de
F3A2 0C	INC C	előtte kiadunk egy hibatörlet (b4) és adásengedélyezés (b0)
F3A3 3E11	LD A,11	parancsot
F3A5 ED79	OUT (C),A	Utána a hibabitek alapján a
F3A7 3EF4	LD A,F4	paritáshiba,
F3A9 CB58	BIT 3,B	
F3AB 2008	JR NZ,F3B5	
F3AD 3EF2	LD A,F2	túlfutás hiba, vagy
F3AF CB60	BIT 4,B	
F3B1 2002	JR NZ,F3B5	
F3B3 3EF3	LD A,F3	kerethiba kódjával
F3B5 4C	LD C,H	a beolvasott karaktert
F3B6 C9	RET	visszaadjuk a hívó programnak

Soros vonal 02 SER-BKOUT/IN
62H, E2H Karakterblokk kiküldés (62H)
Karakterblokk beolvasás (E2H)

in: BC=a blokk hossza
DE=memóriacím

out: A=00 sikeres
hibakód

A blokk I/O művelet lebonyolítása a legmagasabb használható memóriacím ellenőrzésével, az ismert vezérlő rutin segítségével, a karakteres műveletek rutinjainak ismételt hívásával történik.

73B7 FAC0F3 JP M,F3C0 Input esetén ugrás előre

Itt a blokk-output vezérlésével folytatja:

F3BA 213CF3	LD HL,F33C	A CH-OUT rutin címével ugrás a
F3BD C34BF2	JP F24B	vezérlőrutinra

Vezérlés átadás a blokk-input rutinnak:

F3C0 2159F3	LD HL,F359	A CH-IN rutin címével ugrás a
F3C3 C36CF2	JP F26C	vezérlőrutinra

Megemlítjük még, hogy a 4-es funkció hívása esetén a vezérlés a 0F357H címre adódik, mely mindössze egy XOR A és RET utasítás végrehajtását jelenti, tehát nem csinál semmit. Szerepeltetésére elvileg azért van szükség, hogy a külső eszközökkel lebonyolított kétirányú adatforgalom során jelentős feladatot ellátó CLOSE funkció hívása a soros vonal esetén se okozzon hibát. A 4-es funkció a magnetofon és a diszkes adattárolás esetén a korrekt működéshez nem nélkülözhető. Kínos lenne, ha a soros vonal hozzárendelése esetén állandó hibát kapnánk.

A soros vonalat működtető ROM területhez tartozik még a BAUD-ban tárolt érték és a neki megfelelő PITCH érték egymáshoz rendelését tartalmazó táblázat:

ROM	F3C6	F3C8	F3CA	F3CC	F3CE	F3D0	F3D2	F3D4	F3D6
(BAUD)	(00)	(01)	(02)	(03)	(04)	(05)	(06)	(07)	(08)
PITCH	0C88	0D75	0E6A	0F5D	0F4F	0FD7	0FEC	0FF6	0FFB

Az USART inicializálása során a BAUD rendszerváltozóban beállított érték alapján e táblázatból választja ki a rendszer a hanggenerátor beállításához tartozó PITCH értéket.

3.8.3 A magnetofon kezelése

3.8.3.1 Bevezetés

Ezzel, kedves Olvasó, el is érkeztünk -- az utolsó eszköz -- a magnetofon működését vezérlő programokhoz.

Előrebocsátom, hogy eléggé nagyszabású feladat előtt állunk, alapos és koncentrált figyelmet igénylő munkára kell felkészülnünk.

A magnetofon alapvetően nem számítástechnikai eszköz. Pillanatnyilag ami fölött rendelkezni tudunk az három port cím:

- az 50H port írásával a magnetofon felvétel esetén használt vonalát tudjuk az ellenkező jelszintű állapotba hozni;
- az 59H port b5 bitjén tudjuk érzékelni a magnóról érkező jel szintjét;
- a 05 port b7-b6 bitjei a két magnócsatlakozó vezérlését látják el.

Ezekből kell egy elfogadhatóan megbízható bitsoros adatforgalmat szervezni. A munkához az U0 RAM-ban tekintélyes méretű munkaterületet használunk. Emellett néhány olyan rendszerváltozó is a felhasználó rendelkezésére áll, amelyekkel az átvitel körülményei kívülről is befolyásolhatók.

A magnetofon kezeléséhez tartozó funkcióhívások ugrótáblája a fő ROM 0D9BBH címén található. Az ottani címeken azonban egy-egy ugróutasítás a vezérlést közvetlenül átadja az EXT ROM-nak. Az ott kijelölt végrehajtási címek a következő elágazáshoz vezetnek:

F3D8 CD93F5	CALL F593	CAS 01
F3DB 1817	JR F3F4	
F3DD CD30F6	CALL F630	CAS 02
F3E0 1812	JR F3F4	
F3E2 CDFAF3	CALL F3FA	CAS 03
F3E5 180D	JR F3F4	
F3E7 CD58F5	CALL F558	CAS 04
F3EA 1808	JR F3F4	
F3EC CD05F6	CALL F605	CAS 05
F3EF 1803	JR F3F4	
F3F1 CD07F7	CALL F707	CAS-INIC

A 0F3F4H közös, relatív ugrási címen pedig a fő ROM-ba való visszatéréshez szükséges két utasítás található:

73F4 21E7D9	LD HL, D9E7	Itt a fő ROM-ban már csak egy
73F7 C3F0FF	JP FFF0	RET utasítás található

Látható tehát, hogy az egyes funkciók lebonyolítását meghatározott szubrutinokra bizzuk, s a közös pontból való elágazás után a vezérlés ugyanehhez a közös ponthoz tér vissza. Itt fogadjuk és innen is adjuk vissza a fő ROM-nak a vezérlést. Jól kézben tartható és jól áttekinthető megoldás.

Az elágazási címekből kitűnik, hogy most már egyetlen lépést is téve a legnagyobb munka közepébe csöppenünk.

3.8.3.2 A funkciók vezérlése

CAS 03 CAS-CRTE/OPEN
53H, D3H File megnyitása írásra/olvasásra

in: DE=a file nevének kezdőcíme a memóriában
out: DE=az érvényesített file név címe
A=00 rendben
 hibakódok

F3FA F2CFF4 JP P,F4CF Ha írásra nyitás: ugrás előre

A munka itt az olvasásra való megnyitással folytatódik (0D3H funkciókód, CAS-OPEN).

F3FD CD62F7	CALL F762	Ellenőrzi, hogy van-e már
F400 3EEB	LD A,EB	olvasásra nyitott file
F402 C0	RET NZ	Ha igen A=EB hibakóddal visszatér
F403 21F30B	LD HL,0BF3	Ezután nullázza a
F406 012101	LD BC,0121	0BF3H...0D13H munkaterületet,
F409 CD4FF5	CALL F54F	majd a DE-ben megadott cím
F40C 21F40B	LD HL,0BF4	felhasználásával a 0BF4H
F40F CD26F5	CALL F526	címtől előállítja a szabványos nevet,
F412 CD3CF7	CALL F73C	új sort kezd és
F415 214EF7	LD HL,F74E	kiírja:
F418 CD30F7	CALL F730	"Searching"
F41B CD3CF7	CALL F73C	Ismét új sor
F41E 3EFF	LD A,FF	Adminisztrál:
F420 32130D	LD (0D13),A	"Nyitási folyamat" jelzése,
F423 21050C	LD HL,0C05	olvasási puffercím,
F426 22100B	LD (0D10),HL	a még beolvasható byte-ok
F429 210001	LD HL,0100	száma,
F42C 22090D	LD (0D09),HL	
F42F AF	XOR A	
F430 320B0D	LD (0D0B),A	az aktuális hibakód,
F433 320C0D	LD (0D0C),A	nem írásvédett file,
F436 320D0D	LD (0D0D),A	a beolvasható szektor száma
F439 CD7BF7	CALL F77B	Ezután beolvas egy blokkot
F43C 3813	JR C,F451	Ha sikeres volt: ugrás!

Ha a beolvasás sikertelen, a munka így zajlik tovább:

F43E 3A0B0D	LD	A, (0D0B)	Nézzük a hibakódot
F441 FEF5	CP	F5	CTRL+ESC?
F443 20D9	JR	NZ, F41E	Ha nem: újra próbálkozik
F445 210000	LD	HL, 0000	STOP esetén törli az
F448 226F0B	LD	(0B6F), HL	ellenőrzőbyte-okat (CRC),
F44B 21F30B	LD	HL, 0BF3	"nincs nyitott file" jelzést
F44E 3600	LD	(HL), 00	állít be és a hibakóddal
F450 C9	RET		viSSzatér

Ha a beolvasás sikeres volt, akkor a további munka a következő:

F451 2158F7	LD	HL, F758	A "Found:" szövegre áll
F454 D9	EXX		
F455 21050C	LD	HL, 0C05	A beolvasott név címe
F458 E5	PUSH	HL	
F459 11F40B	LD	DE, 0BF4	A megadott név címe
F45C 1A	LD	A, (DE)	A név hossza. Amennyiben nevet
F45D B7	OR	A	nem adtak meg (0), akkor bármit
F45E 280A	JR	Z, F46A	elfogad, ugrás előre
F460 47	LD	B, A	
F461 04	INC	B	
F462 1A	LD	A, (DE)	Egyébként ellenőrzi a megadott
F463 96	SUB	(HL)	és beolvasott nevek azonos-
F464 2008	JR	NZ, F46E	ságát, s ha nem egyezik: ugrás!
F466 13	INC	DE	
F467 23	INC	HL	
F468 10F8	DJNZ	F462	Ha a kért file-t találta meg,
F46A 2144F7	LD	HL, F744	a "Reading:" szóra áll
F46D D9	EXX		
F46E D9	EXX		
F46F 08	EX	AF, AF	
F470 CD30F7	CALL	F730	Kiírja a megfelelő szót
F473 E1	POP	HL	(Found/Reading),
F474 E5	PUSH	HL	
F475 7E	LD	A, (HL)	
F476 B7	OR	A	majd a beolvasott nevet,
F477 C430F7	CALL	NZ, F730	
F47A CD3CF7	CALL	F73C	új sort kezd,
F47D 08	EX	AF, AF	
F47E E1	POP	HL	s ha a két név nem egyezett,
F47F 209D	JR	NZ, F41E	tovább keres

Ezután az írásvédelemmel kapcsolatos felügyeletet látjuk el.

F481 3A140D	LD	A, (0D14)	Van írásra nyitott file?
F484 B7	OR	A	Ha nincs, akkor baj nem lehet:
F485 280E	JR	Z, F495	meHet előre!

F487	210C0D	LD	HL,0D0C	Írásra is nyitott file esetén
F48A	7E	LD	A,(HL)	megnézzük, hogy a beolvasott
F48B	B7	OR	A	file írásvédett-e. Ha nem:
F48C	2807	JR	Z,F495	mehet tovább, írásvédett file
F48E	AF	XOR	A	esetén azonban ez az állapot
F48F	77	LD	(HL),A	tilos! Ekkor "védelem
F490	3EE6	LD	A,E6	megsértése" hibát jelez
F492	C345F4	JP	F445	

F495	3AF30B	LD	A,(0BF3)	A file típusa:
F498	D611	SUB	11	=00: nem pufferelt
F49A	326B0B	LD	(0B6B),A	Betölti a rendszerváltozót,
F49D	3E00	LD	A,00	majd a "nyitási állapot"
F49F	32130D	LD	(0D13),A	jelzést törli

F4A2	21050C	LD	HL,0C05	A megnyitott file nevét
F4A5	11F40B	LD	DE,0BF4	viSSzatölti a megadott file-
F4A8	7E	LD	A,(HL)	név helyére -- itt a
F4A9	FE11	CP	11	későbbiekben is megmarad, míg
F4AB	3802	JR	C,F4AF	a 0C05H puffer-ben átíródhat.
F4AD	3E10	LD	A,10	A nevet maximum 10H
F4AF	3C	INC	A	karakterrel jegyzi meg
F4B0	4F	LD	C,A	
F4B1	0600	LD	B,00	
F4B3	EDB0	LDIR		

F4B5	21050C	LD	HL,0C05	A puffer kezdőcíméből és a
F4B8	5E	LD	E,(HL)	név hosszából megkeresi a név
F4B9	1C	INC	E	utáni első beolvasott byte
F4BA	1600	LD	B,00	címét, s ezt, mint a következő
F4BC	19	ADD	HL,DE	felhasználható byte címét,
F4BD	22070D	LD	(0B07),HL	valamint a még felhasználatlan
F4C0	2A050D	LD	HL,(0D05)	byte-ok számát feljegyzi
F4C3	AF	XOR	A	
F4C4	ED52	SBC	HL,DE	
F4C6	22050D	LD	(0D05),HL	

F4C9	11F40B	LD	DE,0BF4	DE-ben visszaadja a név címét
F4CC	C3B5F5	JP	F5B5	és visszatér

File megnyitása írásra (53H, CAS-CRTE)

F4CF	CD6BF7	CALL	F76B	Van már nyitott file?
F4D2	3EEB	LD	A,EB	Ha igen, a hibakóddal
F4D4	C0	RET	NZ	visszatér
F4D5	3A0C0D	LD	A,(0D0C)	Írásvédett file?
F4D8	B7	OR	A	Ha igen: az "írásvédelem
F4D9	3EE6	LD	A,E6	megsértése" hibával tér vissza
F4DB	C0	RET	NZ	

F4DC 21140D	LD	HL,0D14	Ezután inicializálja (törli) a
F4DF 012001	LD	BC,0120	0D14...0E33H munkaterületet. A
F4E2 CD4FF5	CALL	F54F	megadott nevet szabványos
F4E5 21150D	LD	HL,0D15	formában a 0D15H...0D24H
F4E8 CD26F5	CALL	F526	területre másolja
F4EB 3A6B0B	LD	A,(0B6B)	A rendszerváltozóban előírt
F4EE B7	OR	A	file-típus jelzést a saját
F4EF 3E01	LD	A,01	változójába írja:
F4F1 2002	JR	NZ,F4F5	01: pufferelt;
F4F3 3E11	LD	A,11	11: nem pufferelt file
F4F5 32140D	LD	(0D14),A	
F4F8 322FOE	LD	(0E2F),A	
F4FB 216D0B	LD	HL,0B6D	A file védelmére vonatkozó
F4FE 7E	LD	A,(HL)	jelzést a rendszerváltozóban
F4FF 3600	LD	(HL),00	törli, de itt feljegyzi
F501 32300E	LD	(0E30),A	
F504 3EFF	LD	A,FF	
F506 32320E	LD	(0E32),A	"Nyitási állapot" jelzés
F509 21150D	LD	HL,0D15	A már feldolgozott file-nevet
F50C 11260D	LD	DE,0D26	a kiviteli puffer elejére
F50F ED532C0E	LD	(0E2C),DE	másolja (e címet feljegyzi), s
F513 4E	LD	C,(HL)	a névhossz lesz egyelőre a
F514 0C	INC	C	kiírandó byte-szám is
F515 0600	LD	B,00	
F517 ED43260E	LD	(0E26),BC	
F51B EDB0	LDIR		A név utáni első byte címét is
F51D ED53280E	LD	(0E28),DE	tárolja (a következő kiírandó
			byte)
F521 11150D	LD	DE,0D15	A név kezdőcíme DE-be
F524 AF	XOR	A	
F525 C9	RET		és kész

Ezután két segédrutin következik.

Szabványos file-név előállítás

Belépéskor DE tartalmazza a megadott file-név kezdő memóriacímét, HL pedig a célpuffer címét, ahová a konvertált név kerülni fog.

F526 EB	EX	DE,HL	A két címet felcseréli
F527 7E	LD	A,(HL)	Ellenőrzi a névhosszt:
F528 FE11	CP	11	ha a megadott név 10-nél
F52A 3802	JR	C,F52E	hosszabb, belőle csak az első
F52C 3E10	LD	A,10	10H karaktert veszi figyelembe
F52E 12	LD	(DE),A	

F52F B7	OR	A	Ha a hossz 0: visszatér
F530 C8	RET	Z	
F531 47	LD	B,A	
F532 23	INC	HL	
F533 13	INC	DE	
F534 7E	LD	A, (HL)	Ezután a névhossznak megfelelő
F535 FE61	CP	61	ciklust szervez és
F537 3812	JR	C,F54B	karakterenként ellenőrzi a
F539 FE7B	CP	7B	nevet:
F53B 3004	JR	NC,F541	a kisbetűket nagybetűkre
F53D E6DF	AND	DF	változtatja
F53F 180A	JR	F54B	
F541 FE90	CP	90	
F543 3806	JR	C,F54B	
F545 FE99	CP	99	
F547 3002	JR	NC,F54B	
F549 D610	SUB	10	Az esetleges átalakítás után a
F54B 12	LD	(DE),A	karaktert átmásolja
F54C 10E4	DJNZ	F532	
F54E C9	RET		

Megadott memóriaterület nullázása

HL a kijelölt terület kezdőcímét, BC a byte-ok számát tartalmazza.

F54F 3600	LD	(HL),00	
F551 23	INC	HL	Ebben legfeljebb az a
F552 0B	DEC	BC	kiemelendő, hogy a DE
F553 78	LD	A,B	regiszterpárt nem használja
F554 B1	OR	C	
F555 20F8	JR	NZ,F54F	
F557 C9	RET		

CAS 04 CAS-CLOSE
54H, D4H File-ok lezárása

in: -
out: -

7558 FA85F5 JP M,F585 Ha olvasásra nyitott file-t
kell lezárni, ugrás előre

Itt az írásra nyitott file-ok lezárásával folytatja.

F55B CD6BF7	CALL F76B	Van írásra nyitott file?
F55E 281C	JR Z,F57C	Ha nincs: ugrás előre
F560 21260D	LD HL,0D26	Feljegyzzi az output puffer
F563 222C0E	LD (0E2C),HL	elejét
F566 3EFF	LD A,FF	
F568 322B0E	LD (0E2B),A	
F56B 3A2A0E	LD A,(0E2A)	Ha már be van állítva a file
F56E B7	OR A	vége jelzés, ugrás
F56F 200B	JR NZ,F57C	előre,
F571 CD72F9	CALL F972	egyébként a tömb kiírása és ha
F574 3806	JR C,F57C	sikeres volt, ugrás előre
F576 08	EX AF,AF	Hiba esetén a lezárási
F577 CD7CF5	CALL F57C	adminisztráció közben megőrzi
F57A 08	EX AF,AF	a hibakódot
F57B C9	RET	
F57C AF	XOR A	Adminisztráció:
F57D 32140D	LD (0D14),A	nincs írás,
F580 322A0E	LD (0E2A),A	file vége volt,
F583 1807	JR F58C	a többet később

Olvasásra nyitott file-ok lezárása

F585 AF	XOR A	Saját adminisztráció:
F586 32F30B	LD (0BF3),A	nincs nyitott file,
F589 320C0D	LD (0B0C),A	nincs írásvédelem

S végül minden CLOSE műveletnél közös:

F58C 210000	LD HL,0000	
F58F 226F0B	LD (0B6F),HL	A CRC induló értéke: 0
F592 C9	RET	

CAS 01	CAS-CHOUT/IN
51H, D1H	Karakter kiküldése (51H)
	Karakter beolvasása (D1H)

F593 F253F6	JP P,F653	Outputnál ugrás előre
-------------	-----------	-----------------------

Karakter beolvasása

A karakter kódját a C regiszterben, a hibakódot pedig az A-ban kapjuk.

F596 CD62F7	CALL F762	Ha nincs olvasásra nyitott
F599 C8	RET Z	file: hiba!
F59A CD72F7	CALL F772	Ha már file vége volt: hiba!
F59D C0	RET NZ	

F59E ED4B050D	LD	BC, (0D05)	A szalagról betöltött, de még nem felhasznált byte-ok száma
F5A2 78	LD	A, B	
F5A3 B1	OR	C	
F5A4 281B	JR	Z, F5C1	Ha a pufferben több már nincs: ugrás előre
F5A6 2A070D	LD	HL, (0D07)	A következő byte címe a pufferben, innen tölti C-t, a címet tovább lépteti, majd tárolja,
F5A9 4E	LD	C, (HL)	egyben a még felhasználható byte-ok számát is csökkenti
F5AA 23	INC	HL	
F5AB 22070D	LD	(0D07), HL	
F5AE 2A050D	LD	HL, (0D05)	
F5B1 2B	DEC	HL	
F5B2 22050D	LD	(0D05), HL	
F5B5 7C	LD	A, H	
F5B6 B5	OR	L	
F5B7 2006	JR	NZ, F5BF	Ha még nem 00: mehet, de ha nincs több byte, a szektor vége jelet áttölti a file-vége rendszerváltozóba
F5B9 3A0E0D	LD	A, (0D0E)	
F5BC 326E0B	LD	(0B6E), A	
F5BF AF	XOR	A	
F5C0 C9	RET		

A szektorok végét 00 byte jelzi, tehát az áttöltés nem okoz file-vége beállítást, kivéve az utolsó szektort, ahol a szektor végjel egyben a file végjele is: 0FFH.

Ha az input pufferben már nincs több felhasználatlan karakter, akkor egy új szektort kell beolvasni. Ez a következőképpen történik.

Egy szektor betöltése a szalagról

F5C1 3AF30B	LD	A, (0BF3)	Nem pufferelt file-ok esetén ismételt beolvasásra nincs lehetőség: a hibakóddal visszatér!
F5C4 3D	DEC	A	
F5C5 3EE7	LD	A, E7	
F5C7 C0	RET	NZ	
F5C8 210001	LD	HL, 0100	Pufferelt file-oknál beállítja újra a beolvasható byte-ok számát, a puffer kezdőcímét, s a következő beolvasható byte címét
F5CB 22090D	LD	(0D09), HL	
F5CE 21050C	LD	HL, 0C05	
F5D1 22100B	LD	(0B10), HL	
F5D4 22070D	LD	(0D07), HL	
F5D7 3A0E0D	LD	A, (0D0E)	Ellenőrzi az előző szektor végjelét: ha még nem file-végjel volt, akkor betölt a szalagról egy új tömböt
F5DA B7	OR	A	
F5DB 3EEC	LD	A, EC	
F5DD CC7BF7	CALL	Z, F77B	
F5E0 300C	JR	NC, F5EE	Ha nem sikerült: ugrás előre

F5E2 3AF30B	LD	A,(0BF3)	Ha pufferelt file volt, akkor
F5E5 3D	DEC	A	A=00, ugrás vissza:
F5E6 28BE	JR	Z,F5A6	folytatható a
			karakterbeolvasás
F5E8 AF	XOR	A	Nem pufferelt file esetén CY=1
F5E9 37	SCF		jelzi a tömb sikeres
F5EA C9	RET		betöltését
F5EB 320B0D	LD	(0D0B),A	Elteszi a hibakódot
F5EE 210B0D	LD	HL,0D0B	
F5F1 7E	LD	A,(HL)	A hibakódot A-ba tölti,
F5F2 36EC	LD	(HL),EC	beállítja a "file vége"
F5F4 210E0D	LD	HL,0D0E	hibakódot és
F5F7 36FF	LD	(HL),FF	jelzést,
F5F9 210000	LD	HL,0000	a még beolvasható byte-ok
F5FC 22050D	LD	(0D05),HL	számát törli,
F5FF 216E0B	LD	HL,0B6E	s a rendszerváltozóba is
F602 36FF	LD	(HL),FF	betölti a "file vége" jelzést,
F604 C9	RET		majd a hibakóddal visszatér

Ezt a rutint több ponton belépve is használja a rendszer. Az egyes vizsgálatok más funkciók esetén is elvégezhetők, pl. az OPEN rutin a 0F5B5H résztől belépve fejezi be itt a működést, vagy a szektorvégjel vizsgálatával kezdődő 0F5D7H részt a VERIFY is használja. A funkciók ilyen összefonódása természetesen nehezíti a működés követését.

CAS 05 CAS-VERIFY
D5H File- és memóriatartalom összehasonlítás

in: DE=memóriacím
out: BC=hossz
A=00 egyezik
hibakódok, ekkor:
DE=az eltérést mutató memóriacím
BC=a maradék hossz

F605 CD62F7	CALL	F762	Ellenőrzi, hogy van-e
F608 C8	RET	Z	olvasásra nyitott file
			Ha nincs, A=E9 hibakóddal
			visszatér
F609 CD72F7	CALL	F772	Ha file vége volt: A=EC
F60C C0	RET	NZ	hibakóddal tér vissza
F60D 3EFF	LD	A,FF	
F60F 32F10B	LD	(0BF1),A	VERIFY jelzése
F612 3AF30B	LD	A,(0BF3)	Ha nem pufferelt file, akkor
F615 3D	DEC	A	ugrás előre, itt csak a
F616 2027	JR	NZ,F63F	pufferelt file-t ellenőrzi

F618 C5	PUSH BC	
F619 D5	PUSH DE	
F61A CD9EF5	CALL F59E	A CH-IN rutinnal beolvas egy
F61D 69	LD L,C	byte-ot
F61E D1	POP DE	
F61F C1	POP BC	
F620 B7	OR A	Olvasási hiba esetén vissz-
F621 C0	RET NZ	szatér,
F622 1A	LD A,(DE)	egyébként összehasonlítja a
F623 BD	CP L	memória tartalmával és eltérés
F624 3EE8	LD A,E8	esetén a hiba kódjával tér
F626 C2EBF5	JP NZ,F5EB	vissza
F629 13	INC DE	Ha a két adat azonos, továbblép
F62A 0B	DEC BC	a memóriában és folytatja a
F62B 78	LD A,B	komparálást, amíg BC=0000 lesz
F62C B1	OR C	
F62D 20E9	JR NZ,F618	
F62F C9	RET	

A nem pufferelt file-ok komparálása a blokkbetöltési funkció végrehajtása keretében zajlik: a 0BF1H változó tartalma szerint történik tényleges betöltés vagy csak összehasonlítás.

A blokkműveletek vezérlőrutinja következik. A két műveleti irány itt is határozottan ketté válik, hiszen mindegyiknél teljesen másképpen kell a munkát megszervezni.

CAS 02	CAS-BKOUT/IN
52H, D2H	Karakterblokk kiírása szalagra (52H), Blokk beolvasása szalagról (0D2H)

F630 F2ADF6	JP P,F6AD	Output műveletnél ugrás előre
-------------	-----------	-------------------------------

Itt a blokkbeolvasás vezérlőrutinjával folytatjuk.

F633 CD62F7	CALL F762	Ha nincs olvasásra nyitott
F636 C8	RET Z	file vagy ha file vége volt, a
F637 CD72F7	CALL F772	hiba kódjával visszatér
F63A C0	RET NZ	
F63B 3AF30B	LD A,(0BF3)	Ellenőrzi a file típusát
F63E 3D	DEC A	Pufferelt file-ok esetén a
F63F 219EF5	LD HL,F59E	karakteres beolvasás ismételt
F642 CA6CF2	JP Z,F26C	hívásával oldja meg a fela-
		datot

F645	ED53100D	LD	(0D10),DE	Nem pufferelt file-oknál pedig
F649	ED43090D	LD	(0D09),BC	közvetlenül a tömbök beol-
F64D	CDD7F5	CALL	F5D7	vasását szervező rutint hívja
F650	C3F4F5	JP	F5F4	meg

Karakterek kiírása szalagra (51H)

A CAS-CHOUT funkcióhívás esetén a vezérlést a közvetkező rutin kapja meg. Belépéskor a kiírandó karakter a C regiszterben található.

F653	CD6BF7	CALL	F76B	Ha nincs írásra megnyitott
F656	C8	RET	Z	file vagy már file vége volt,
F657	3A2A0E	LD	A,(0E2A)	akkor a megfelelő hibakóddal
F65A	B7	OR	A	visszatér
F65B	3EEC	LD	A,EC	
F65D	C0	RET	NZ	

F65E	79	LD	A,C	A kiírandó karaktert egyelőre
F65F	322E0E	LD	(0E2E),A	itt tárolja
F662	212F0E	LD	HL,0E2F	Pufferelt file-oknál a puffer
F665	7E	LD	A,(HL)	állapotától függően történik
F666	B7	OR	A	tényleges írás, nem pufferelt
F667	2804	JR	Z,F66D	file esetén pedig csak blokk-
F669	35	DEC	(HL)	írási parancs hatására
F66A	CA89F6	JP	Z,F689	

F66D	ED5B260E	LD	DE,(0E26)	
F671	210001	LD	HL,0100	Ha már 256 db kiküldendő byte
F674	B7	OR	A	összegyűlt (pufferelt), akkor
F675	ED52	SBC	HL,DE	felírja a tömböt, egyébként
F677	CA89F6	JP	Z,F689	csak adminisztrál
F67A	13	INC	DE	
F67B	ED53260E	LD	(0E26),DE	A kiírandó byte-ok száma
F67F	2A280E	LD	HL,(0E28)	A következő kiírandó byte
F682	71	LD	(HL),C	címére leteszi a karaktert és
F683	23	INC	HL	a címmel továbblép
F684	22280E	LD	(0E28),HL	
F687	AF	XOR	A	Ilyenkor egyelőre kész
F688	C9	RET		

Következik a tömb tényleges felírásának vezérlése.

F689	21260D	LD	HL,0D26	Betölti a kiírandó terület
F68C	222C0E	LD	(0E2C),HL	kezdőcímét és szalagra írja a
F68F	CD72F9	CALL	F972	tömböt
F692	3805	JR	C,F699	Ha sikeres volt tovább,
F694	212A0E	LD	HL,0E2A	egyébként a "file vége" jelzés
F697	35	DEC	(HL)	beállításával tér vissza
F698	C9	RET		

F699 210100	LD	HL,0001	A puffer kiürítése után az új
F69C 22260E	LD	(0E26),HL	karakter lesz az első: beírja
F69F 3A2E0E	LD	A,(0E2E)	a sorszámot és az elmentett
F6A2 21260D	LD	HL,0D26	karakter a puffer elejére
F6A5 77	LD	(HL),A	kerül, a következőt byte címét
F6A6 23	INC	HL	feljegyzí
F6A7 22280E	LD	(0E28),HL	
F6AA AF	XOR	A	
F6AB 37	SCF		
F6AC C9	RET		és kész

A blokk írási funkció végrehajtása (52H)

Belépéskor DE és BC a megszokott módon a memória kezdőcímét és a hosszt tartalmazza

F6AD CD6BF7	CALL	F76B	Ha nincs írásra nyitott file,
F6B0 C8	RET	Z	visszatér
F6B1 3A140D	LD	A,(0D14)	Pufferelt file-ok esetén a
F6B4 3D	DEC	A	karakterek kiírását a HI-MEM
F6B5 2157F6	LD	HL,F657	átlépés ellenőrzésével, a
F6B8 CA4BF2	JP	Z,F24B	karakterírás ismételt meghí-
			vásával végzi
F6BB 3A2A0E	LD	A,(0E2A)	
F6BE B7	OR	A	Ha file vége volt, akkor a
F6BF 3EEC	LD	A,EC	megfelelő hibakóddal visszatér
F6C1 C0	RET	NZ	

Ezután kerül sor a nem pufferelt file-ok tényleges felírására.

F6C2 C5	PUSH	BC	
F6C3 D5	PUSH	DE	
F6C4 212F0E	LD	HL,0E2F	
F6C7 7E	LD	A,(HL)	Először -- ha még nem történt
F6C8 3600	LD	(HL),00	meg -- a megfelelő jelzés
F6CA B7	OR	A	beállítása után
F6CB 37	SCF		szalagra írja a fejléct
F6CC C489F6	CALL	NZ,F689	
F6CF D1	POP	DE	
F6D0 C1	POP	BC	
F6D1 30C1	JR	NC,F694	és, ha ez sikeres volt, akkor
F6D3 ED43260E	LD	(0E26),BC	a byte-ok számát és a kiírandó
F6D7 ED532C0E	LD	(0E2C),DE	terület kezdőcímét beállítva
F6DB C366F5	JP	F566	felírja az egész tömböt

3.8.3.3 A magnetofonkezelés alacsony szintű rutinjai

Ezzel valamennyi hívható funkciót, s a hozzájuk szorosan kapcsolódó néhány segédrutint elintéztük. Az Olvasó azonban nyilván észrevette, hogy valójában csak a rendszerváltozók, a belső munkaterületek és pufferek gondos könyvelését végeztük el. Az Olvasót is feltehetően komolyan foglalkoztató tényleges kiírási és beolvasási műveletekről csak annyit mondtunk még mindig, hogy meghívjuk a hozzájuk tartozó szubrutinokat.

Azt hiszem nem túlzás azt állítani, hogy az érdekesebb munka csak ezután következik.

Rövidesen nagyon alacsony szintű rutinokat ismerünk meg. Tisztázzuk, hogy a rendelkezésre álló minimális hardvereszközökkel hogyan lehet a magnetofon felé irányuló összetettebb feladatokat megvalósítani.

Kezdjük egy egyszerűbbel!

Magnetofonmotorok távvezérlése

in: A=00 olvasó magnetofon kezelése
FF írásra kijelölt magnetofon kezelése
D=00 motor bekapcsolása
D=FF motor kikapcsolása

F6DE 2E40	LD	L,40	Aszerint, hogy az olvasásra
F6E0 B7	OR	A	vagy az írásra kijelölt magnót
F6E1 2801	JR	Z,F6E4	kell-e kezelni: L=40H vagy 80H
F6E3 29	ADD	HL,HL	A felhasználó kiosztási szán-
F6E4 3A6C0B	LD	A,(0B6C)	dékát ez a rendszerváltozó
			tartalmazza
F6E7 A5	AND	L	Olvasásnál a b6, írásnál a b7
			bitet vizsgáljuk

A rendszerváltozóba beírt értékkel összevetve ez azt jelenti, hogy ezen a ponton olvasás esetén A=00: ha a bal oldali, A=40H: ha a jobb oldali készüléket kell kapcsolni. Írásnál pedig 00 a jobb, 80H a bal oldali magnetofont jelenti.

F6EB 2804	JR	Z,F6EE	Ha olvasásnál a bal, írásnál a
			jobb oldali kell: ugrás előre
F6EA 3ECC	LD	A,C0	Egyébként a két bitet fel-
F6EC AD	XOR	L	cseréli

F6ED 6F	LD	L,A	
F6EE F3	DI		
F6EF 3A120B	LD	A, (0B12)	Ezután a 05-ös port másolatát felhasználva kijelöli a kapcsolandó magnetofonmotor bitjét, s aszerint, hogy be- vagy kikapcsolásról van szó, elvégzi a kapcsolást
F6F2 B5	OR	L	
F6F3 14	INC	D	
F6F4 15	DEC	D	
F6F5 2801	JR	Z,F6F8	
F6F7 AD	XOR	L	
F6F8 D305	OUT	(05),A	
F6FA 32120B	LD	(0B12),A	
F6FD FB	EI		
F6FE C9	RET		

Mindkét magnetofonmotor kikapcsolása esetén a következő belépési pont használható:

F6FF F3	DI		
F700 3A120B	LD	A, (0B12)	A 05-ös port kópiájában a többi bitet változatlanul hagyva a motorvezérlő biteket nullázzuk
F703 E63F	AND	3F	
F705 18F1	JR	F6F8	

És mielőtt a legkomolyabb munkába kezdenénk, előtte még két egyszerűbb funkció rutinja.

CAS-INIC

F707 AF	XOR	A	
F708 21F00B	LD	HL,0BF0	Nullázza a magnetofonkezelő rutinok munkaterületét (0BF0...0E33H), majd alapértékeket állít be:
F70B 11F10B	LD	DE,0BF1	nem írásvédett file,
F70E 014402	LD	EC,0244	nincs file vége és
F711 77	LD	(HL),A	az ellenőrző összeg (CRC) alapértéke: 0
F712 EDB0	LDIR		
F714 326D0B	LD	(0B6D),A	
F717 326E0B	LD	(0B6E),A	
F71A 210000	LD	HL,0000	
F71D 226F0B	LD	(0B6F),HL	
F720 3E80	LD	A,80	Olvasásra és írásra is a bal oldali magnetofont jelöli ki a motorvezérléssel, de mindkét motort kikacsolja
F722 326C0B	LD	(0B6C),A	
F725 3A120B	LD	A, (0B12)	
F728 E63F	AND	3F	
F72A 32120B	LD	(0B12),A	
F72D D305	OUT	(05),A	
F72F C9	RET		

Ezután egy egyszerű szövegkiíró rutin következik. Belépéskor HL a szöveg kezdőcímére mutat, amelyet itt is hossz+szöveg alakban kell a memóriában elhelyezni.

F730 46	LD B, (HL)	
F731 29	INC HL	
F732 C5	PUSH BC	Láthatóan egyszerű ciklus
F733 E5	PUSH HL	szervezéséről van szó, amely-
F734 4E	LD C, (HL)	ben a szöveg karaktereit az
F735 F721	RST 30: 21	editor karakteres output
F737 E1	POP HL	funkcióhívásával küldünk ki
F738 C1	POP BC	
F739 10F6	DJNZ F731	
F73B C9	RET	
F73C 2141F7	LD HL, F741	Ez egy soremelést okoz
F73F 18EF	JR F730	

Az itt aktuális szövegek pedig:

ROM	Hossz	Karakterkódok	Szöveg
F741	02	0D 0A	új sort kezd
F744	09	52 65 61 64 69 6E 67 3A 20	"Reading: "
F74E	09	53 65 61 72 63 68 69 6E 67	"Searching"
F758	09	46 6F 75 6E 64 3A 20 20 20	"Found: "

Majd két ellenőrző/hibakezelő rutin jön.

F762 AF	XOR A	"Nem verify" jelzést állít be
F763 32F10B	LD (OBF1), A	
F766 3AF30B	LD A, (OBF3)	Van olvasásra megnyitott file?
F769 1803	JR F76E	Ugrás előre
F76B 3A140B	LD A, (OD14)	Van már írásra megnyitott file?
F76E B7	OR A	Z=1: nincs
F76F 3EE9	LD A, E9	Előkészíti a "nincs megnyitva file" hibát és a Z jelzőbittel
F771 C9	RET	visszatér

A másik rutin az I/O művelet hibátlanágát ellenőrzi. Ha az átvitel nem volt rendben, file vége hibakódot állít be.

Hibaellenőrzés

F772 210B0D	LD	HL,0D0B	
F775 7E	LD	A,(HL)	Volt hiba?
F776 B7	OR	A	
F777 C8	RET	Z	Ha nem: visszatér
F778 36EC	LD	(HL),EC	Egyébként lerakja a hibakódot
F77A C9	RET		

Hozzálátunk a nehezéhez. Sajnos itt is lesz még néhány olyan szubrutinhivatkozás, amelyről csak közlöm a működésének lényegét, de már rövidesen mindenre sor kerül.

Egy tömb beolvasása a szalagról

F77B ED73330E	LD	(0E33),SP	
F77F AF	XOR	A	
F780 57	LD	D,A	A kijelölt motor bekapcsolása
F781 CDDEF6	CALL	F6DE	olvasásra
F784 CD7CFA	CALL	FA7C	A hang IT bekapcsolása

Ez egy igen fontos szubrutin, majd annak idején részletesen tanulmányozzuk.

F787 AF	XOR	A	
F788 32F00B	LD	(0BFO),A	Fekete border jelzése

Nagyszabású, összetett mintavételezési eljárás kezdődik a magnetofon szalagsebességének és a jelek információtartalmának ellenőrzésére.

F78B D9	EXX		
F78C 210000	LD	HL,0000	Kezdő paraméterek a mérésekhez
F78F 0600	LD	B,00	
F791 D9	EXX		
F792 3EDC	LD	A,DC	PITCH érték, alsó byte,
F794 D304	OUT	(04),A	periódusidő beállítása
F796 0E00	LD	C,00	
F798 CD3CF9	CALL	F93C	Olvas egy négyszögjelet a magnetofonról, hossza E-ben

Ez is egy fölöttébb figyelemreméltó szubrutin lesz.

F79B 210000	LD	HL,0000	Ezután egy rövidebb ciklust
F79E 0620	LD	B,20	indítunk
F7A0 54	LD	D,H	

```

F7A1 CD38F9    CALL F938
F7A4 19        ADD  HL,DE
F7A5 10FA     DJNZ F7A1
F7A7 29        ADD  HL,HL
F7A8 29        ADD  HL,HL
F7A9 29        ADD  HL,HL
F7AA 7C        LD   A,H
F7AB CB15     RL   L
F7AD 88        ADC  A,B
F7AE 57        LD   D,A

```

Most már a hanggenerátor alaphelyzetével pontosan szinkronizálva megmérjük a magnetofonról jövő 20H db négyszögperiódus összes idejét, melynek 8-szorosából csak a felső byte-ot véve és felfelé kerekítve -- az eredményt végül 256-tal elosztva -- 1 periódus átlagidejét kapjuk

A most a D regiszterbe került érték tehát már a magnetofon szalagsebességére jellemző, de némiképp azt is mutatja, hogy a szalagon egyáltalán négyszögjelek vannak-e.

A második nagyon jelentős ellenőrző fázis kezdődik.

```

F7AF 210004    LD   HL,0400
F7B2 CD38F9    CALL F938
F7B5 D9        EXX
F7B6 80        ADD  A,B
F7B7 47        LD   B,A
F7B8 3001     JR   NC,F7BB
F7BA 23        INC  HL
F7BB D9        EXX
F7BC 7B        LD   A,E
F7BD 92        SUB  D
F7BE 3002     JR   NC,F7C2
F7C0 ED44     NEG
F7C2 FE03     CP   03
F7C4 30C1     JR   NC,F787
F7C6 2B        DEC  HL
F7C7 7C        LD   A,H
F7C8 B5        OR   L
F7C9 20E7     JR   NZ,F7B2

```

1024 (dec) periódust nézünk

Az egymásutáni periódusok hosszát HL'-ben és B'-ben összegezve számláljuk

Ha bármelyik periódus hossza 3-nál többel tér el az előbb mért és számolt átlagtól, akkor az egész mintavételt újra kezdjük

Mindezt tehát 1024 perióduson át (!) és csak akkor megy tovább, ha folyamatosan ennyi jó négyszögjelet talált

```

F7CB D9        EXX
F7CC 29        ADD  HL,HL
F7CD 29        ADD  HL,HL
F7CE 29        ADD  HL,HL
F7CF 29        ADD  HL,HL
F7D0 29        ADD  HL,HL
F7D1 29        ADD  HL,HL
F7D2 7C        LD   A,H
F7D3 CB15     RL   L
F7D5 CE00     ADC  A,00
F7D7 D9        EXX
F7D8 F5        PUSH AF

```

Az így nyert összegből ismét egy átlagértéket képezünk. Dec. 64-gyel szorozva (64x1024=65536) és kerekítés után ismét csak a felső byte-ot véve az összes mért impulzus hosszának átlagát kapjuk

Ezt a veremben tároljuk

F7D9 3E88	LD	A,88	
F7DB 32F00B	LD	(0BF0),A	Vörös border

Újabb-ellenőrzés következik, de most már a 0 szintű és az 1 szintű jelfázisokat külön-külön is figyeljük, sőt az órajelet is átállítjuk.

F7DE 3EE8	LD	A,E8	Új PITCH értékkel állítjuk be
F7E0 D304	OUT	(04),A	a hanggenerátor periódusidejét

F7E2 CD3CF9	CALL	F93C	Megvárjuk egy teljes periódus
F7E5 210000	LD	HL,0000	befutását, majd

F7E8 E5	PUSH	HL	az időt csak 1-1 jelváltás között mérve, ügyes veremkezelési technikával külön-külön összeszámoljuk 128 (dec.) db 0 és 1 fázis teljes hosszát, ill. ebből a már ismert eljárással előállítjuk az 1 fázis hosszának átlagát D-ben,
F7E9 44	LD	B,H	
F7EA 54	LD	D,H	
F7EB E3	EX	(SP),HL	
F7EC CD2BF9	CALL	F92B	
F7EF 19	ADD	HL,DE	
F7F0 19	ADD	HL,DE	
F7F1 10F8	DJNZ	F7EB	
F7F3 7C	LD	A,H	
F7F4 CB15	RL	L	

F7F6 88	ADC	A,B	a 0 fázis hosszának átlagát H-ban,
F7F7 57	LD	D,A	
F7F8 E1	POP	HL	
F7F9 7C	LD	A,H	
F7FA CB15	RL	L	
F7FC 88	ADC	A,B	
F7FD 67	LD	H,A	
F7FE 6A	LD	L,D	

s végül mindkettőt:

HL-ben

F7FF D1	POP	DE
F800 CD2BF9	CALL	F92B
F803 7D	LD	A,L
F804 6C	LD	L,H
F805 67	LD	H,A
F806 CD2BF9	CALL	F92B
F809 94	SUB	H
F80A 3007	JR	NC,F813

Ellenőrizzük, hogy egy-egy újabb fázis beolvasása esetén a mért érték és az előbbi átlag között az eltérés megengedett-e

F80C C602	ADD	A,02
F80E 38F3	JR	C,F803
F810 C387F7	JP	F787
F813 FE04	CP	04
F815 38EC	JR	C,F803

Egyúttal megvárjuk a tömb elején a szalagra vitt bevezető hang végét (szinkronperiódus)

F817 3EDC	LD	A,DC
F819 D304	OUT	(04),A

Ismét az előző PITCH érték

F81B CD2BF9	CALL	F92B
F81E FB210C0C	LD	IX,0000
F822 B32A10CB	LD	IX,(0B10)

A szinkronperiódus (a puffer eleje)

F826 CD19F9	CALL F919	Beolvassa az első üres byte-ot,
F829 D9	EXX	beállítja a CRC kezdőértékét,
F82A 2A6F0B	LD HL, (0B6F)	s
F82D ED5B090D	LD DE, (0D09)	a betöltendő byte-ok számát
F831 D9	EXX	

Ezután már a beolvasott információk ellenőrzése következik.

F832 CD19F9	CALL F919	Az ellenőrzőbyte beolvasása.
F835 FE6A	CP 6A	Ha ez nem egyezik:
F837 C287F7	JP NZ, F787	új mintavételezést kezd.
F83A CD19F9	CALL F919	Újabb byte-ot olvas: ez már a
F83D 21130D	LD HL, 0D13	tömböt specifikálja. Ha a tömb
F840 BE	CP (HL)	típusa is megegyezik a
F841 2809	JR Z, F84C	várttal, mehet tovább
F843 7E	LD A, (HL)	Ha fejtömb kellett volna, de
F844 FEFF	CP FF	nem azt talált, akkor tovább
F846 CA87F7	JP Z, F787	vár, különben olvasási hiba!
F849 C3FDF8	JP F8FD	
F84C 320F0D	LD (0D0F), A	A beolvasott tömbtípust, majd
F84F CD19F9	CALL F919	az olvasott file típusát
F852 32F30B	LD (0BF3), A	tárolja, végül az
F855 CD19F9	CALL F919	írásvédelemre vonatkozó
F858 320C0D	LD (0D0C), A	jelzést tölti be
F85B CD19F9	CALL F919	Beolvassa a szektorok számát
F85E 47	LD B, A	és ciklusban szervezett
F85F C5	PUSH BC	beolvasás kezdődik
F860 1806	JR F868	
F862 C5	PUSH BC	
F863 D9	EXX	
F864 2A6F0B	LD HL, (0B6F)	A CRC kezdőértéke
F867 D9	EXX	
F868 CD19F9	CALL F919	Az új szektor sorszáma a
F86B 210D0D	LD HL, 0D0D	szalagon és a beolvasandó
F86E BE	CP (HL)	sorszám. Ha nem egyezik:
F86F C2FDF8	JP NZ, F8FD	olvasási hiba!
F872 CD19F9	CALL F919	A byte-ok száma a szektorban
F875 47	LD B, A	
F876 CD19F9	CALL F919	Egy adatbyte
F879 08	EX AF, AF	
F87A D9	EXX	
F87B 7A	LD A, D	Ha az előírt byte-számot
F87C B3	OR E	elértük, akkor "nem megfelelő
F87D 1B	DEC DE	számú byte olvasási kísérlete"
F87E D9	EXX	hiba,
F87F CA01F7	JP Z, F701	ugrás előre

F882	3AF30B	LD	A, (0BF3)	
F885	3D	DEC	A	
F886	280F	JR	Z, F897	Pufferelt file-oknál ugrás
F888	3AF10B	LD	A, (0BF1)	előre, nem pufferelt file-ok
F88B	B7	OR	A	esetében ha verify funkció
F88C	2809	JR	Z, F897	van, itt folytatja
F88E	08	EX	AF, AF	
F88F	DDBE00	CP	(IX+00)	Összehasonlítás
F892	2807	JR	Z, F89B	Ha jó: mehet tovább,
F894	C309F9	JF	F909	egyébként hiba!
F897	08	EX	AF, AF	
F898	DD7700	LD	(IX+00), A	Leteszti a beolvasott byte-ot
F89B	DD23	INC	IX	
F89D	FD23	INC	IY	Ezt ismételve, beolvassa az
F89F	10D5	DJNZ	F876	egész szektort
F8A1	CD19F9	CALL	F919	Végül betölti és tárolja
F8A4	320E0D	LD	(0D0E), A	a szektor végjelét
F8A7	D9	EXX		
F8A8	E5	PUSH	HL	A CRC számolt értéke.
F8A9	D9	EXX		
F8AA	CD19F9	CALL	F919	A CRC alsó byte-ja a szalagról
F8AD	BF	CP	A	
F8AE	08	EX	AF, AF	
F8AF	E1	POP	HL	
F8B0	44	LD	B, H	
F8B1	CD19F9	CALL	F919	A CRC felső byte
F8B4	4D	LD	C, L	Végül BC=a számolt CRC,
F8B5	67	LD	H, A	
F8B6	08	EX	AF, AF	
F8B7	6F	LD	L, A	HL=a szalagra írt CRC
F8B8	ED42	SBC	HL, BC	Egyezik?
F8BA	C1	POP	BC	
F8BB	2040	JR	NZ, F8FD	Ha nem: olvasási hiba!
F8BD	FD22050D	LD	(0D05), IY	A beolvasott byte-szám
F8C1	DD22100D	LD	(0D10), IX	A következő adatbyte címe, s a
F8C5	210D0D	LD	HL, 0D0D	következő beolvasandó szektor
F8C8	34	INC	(HL)	sorszáma
F8C9	D9	EXX		
F8CA	ED53090D	LD	(0D09), DE	A még beolvasandó byte-szám
F8CE	D9	EXX		
F8CF	1091	DJNZ	F862	Folytatja a szektorok
				beolvasását
F8D1	3AF30B	LD	A, (0BF3)	Pufferelt file-ok esetén ezzel
F8D4	3D	DEC	A	kész,
F8D5	280C	JR	Z, F8E3	ugrás előre,
F8D7	3A130D	LD	A, (0D13)	file-megnyitás esetén
F8DA	FEFF	CP	FF	szintén
F8DC	2805	JR	Z, F8E3	

F8DE D9	EXX		
F8DF 7A	LD	A,D	Ha még ezután is maradt
F8E0 B3	OR	E	beolvasatlan byte: hiba!
F8E1 201E	JR	NZ,F901	
78E3 C3B0FA	JP	FABD	Végül visszakapcsolja a kurzor
			IT-t és kész

Pontosabban, mint majd látni fogjuk, ez a szubrutin a rendszernek azt az állapotát állítja vissza, amelyben a magnetofonról való beolvasást megelőzően volt.

Hibakezelés:

F8E6 3EF5	LD	A,F5	
F8E8 322A0E	LD	(0E2A),A	Adminisztrálja a CTRL+ESC
F8EB 1805	JR	F8F2	lenyomását
F8ED 3EF5	LD	A,F5	
F8EF 320B0D	LD	(0D0B),A	
F8F2 21620B	LD	HL,0B62	Feljegyzzi az OLDPIC mátrixban
F8F5 CBDE	SET	3,(HL)	is
F8F7 0600	LD	B,00	
F8F9 10FE	DJNZ	F8F9	Rövid várakozás után ugrás
F8FB 1811	JR	F90E	előre
F8FD 3EEA	LD	A,EA	
F8FF 180A	JR	F90B	Olvasási hiba jelzése
F901 3AF10B	LD	A,(0BF1)	
F904 B7	OR	A	Nem megfelelő számú byte
F905 3EE7	LD	A,E7	olvasási kísérlete
F907 2802	JR	Z,F90B	
F909 3EE8	LD	A,E8	Összehasonlítási hiba.
F90B 320B0D	LD	(0D0B),A	
F90E F5	PUSH	AF	
F90F CDBDFA	CALL	FABD	Visszaállítja a rendszer
F912 F1	POP	AF	korábbi IT állapotát és a
F913 B7	OR	A	hibakóddal visszatér
F914 ED7B330E	LD	SP,(0E33)	
F918 C9	RET		

Egy byte beolvasása

Belépéskor a D regiszter egy négyszögjelperiódus átlagos hosszát tartalmazza.

F919 2680	LD H,80	H-ban gyűjtjük a biteket
F91B CD38F9	CALL F938	Mérünk egy periódust
F91E BA	CP D	
F91F F5	PUSH AF	A carry tárolása
F920 9F	SBC A,A	
F921 CDF7FA	CALL FAF7	CRC számítás
F924 F1	POP AF	
F925 CB1C	RR H	A bitet beléptetjük H-ba és
F927 30F2	JR NC,F91B	ugrás vissza
F929 7C	LD A,H	Ha a 8 bit megvan: kész
F92A C9	RET	

A következő nagyon fontos rutin -- ha lehet -- még alacsonyabb szintű: közvetlenül a magnetofonról érkező jelváltásig terjedő időt számlálja.

Olvas fél bitet a magnetofonról (egy jelváltás ideje)

F92B DB59	IN A,(59)	A pillanatnyi jelszint a b5
F92D E620	AND 20	biten olvasható le
F92F EE20	XOR 20	Negálja,
F931 4F	LD C,A	erre az állapotra fog várni. A
F932 1E00	LD E,00	számlálót nullázza és
F934 FB	EI	
F935 76	HALT	vár a megszakításkérő jelig,
F936 180F	JR F947	aztán start!

Hogy ezen a címen az időmérés hogyan történik, azt rövidesen látni fogjuk.

Egy bit olvasása a magnetofonról (egy négyszögperiódus hossza)

F938 1E00	LD E,00	Nullázza a számlálót és
F93A FB	EI	várakozik
F93B 76	HALT	
F93C 1C	INC E	Számlál. Időnként ellenőrzi a
F93D CC69F9	CALL Z,F969	CTRL+ESC lenyomását, s állan-
F940 DB59	IN A,(59)	dóan figyel a magnetofon
F942 A9	XOR C	jelét.
F943 E620	AND 20	Addig vár, amíg a C b5 bitjével
F945 28F5	JR Z,F93C	megegyezik

F947 1C	INC	E	
F948 CC69F9	CALL	Z,F969	Majd ugyanezt teszi addig,
F94B DB59	IN	A,(59)	amíg a magnetofonról jövő jel
F94D A9	XOR	C	a C b5 bitjével ellentétes lesz
F94E E620	AND	20	
F950 20F5	JR	NZ,F947	
F952 DB5B	IN	A,(5B)	Ebben a pillanatban a
F954 D307	OUT	(07),A	frekvenciaosztót alaphelyzetbe
F956 3AF00B	LD	A,(0BF0)	állítja, az IT kérést
F959 B7	OR	A	nyugtázza, s ha a border színe
F95A 280D	JR	Z,F969	fekete volt: ugrás előre,
F95C FE88	CP	88	
F95E 3EA0	LD	A,A0	egyébként pedig a vörös színt
F960 2802	JR	Z,F964	kékre változtatja és fordítva
F962 3E88	LD	A,88	
F964 32F00B	LD	(0BF0),A	
F967 D300	OUT	(00),A	
F969 DB58	IN	A,(58)	Ellenőrzi a CTRL+ESC lenyo-
F96B E618	AND	18	mását. Ha STOP volt, akkor
F96D CAEDF8	JP	Z,F8ED	befejezi a beolvasást, egyéb-
F970 7B	LD	A,E	ként pedig a számláló aktuális
F971 C9	RET		értékével visszatér

Talán ez volt ebben a részben a leglényegesebb.

Most a másik oldalt nézzük: hogyan küldi a számítógép a magnetofon felé a felveendő információkat.

Ez annyiból könnyebb feladat, hogy a számítógépben pontosan tudjuk, hogy mikor, s mit akarunk. Ráadásul nem kell azzal foglalkozni, hogy a magnetofon közben hogyan viselkedik -- sajnos nem is tudnánk ezzel foglalkozni! Míg az előbb, ha kellett százszor, ezerszer megkísérelhettük a magnetofonról jövő jelek újbóli és újbóli mintavételezését -- most nem tehetünk mást: kiküldjük a jeleket úgy, ahogy azt kell és kész! Hogy a magnetofon képes-e helyesen feldolgozni, vagy egyáltalán be van-e kapcsolva -- nem tudjuk ellenőrizni!

Tömb felírása a szalagra (kiküldése a magnetofonhoz)

F972 ED73330E	LD	(0E33),SP	
F976 AF	XOR	A	
F977 57	LD	D,A	Először be kell kapcsolni az
F978 3D	DEC	A	írásra kijelölt magnetofon
F979 CDDEF6	CALL	F6DE	motorját


```

F97C 010000 LD BC,0000
F97F E3 EX (SP),H
F980 00 DEC BC
F981 78 LD A,B
F982 B1 OR C
F983 39FA JR NZ,122F

F985 CD7CFA CALL FA7C
F988 3E98 LD A,88
F98A 32F00B LD (0BF0),A
F98D 011400 LD BC,0014
F990 3A320E LD A,(0E32)
F993 B7 OR A
F994 2803 JR Z,F999
F996 012800 LD BC,0028
F999 CD31FA CALL FA31

F99C CD62FA CALL FA62
F99F CD62FA CALL FA62
F9A2 CD3DFA CALL FA3D
F9A5 2A6FOB LD HL,(0B6F)
F9A8 D9 EXX
F9A9 0E6A LD C,6A
F9AB CD3DFA CALL FA3D

F9AE FD2A2C0E LD IY,(0E2C)
F9B2 3A320E LD A,(0E32)
F9B5 4F LD C,A
F9B6 CD3DFA CALL FA3D
F9B9 3A140D LD A,(0D14)
F9BC 4F LD C,A
F9BD CD3DFA CALL FA3D
F9C0 3A300E LD A,(0E30)
F9C3 4F LD C,A
F9C4 CD3DFA CALL FA3D

F9C7 2A260E LD HL,(0E26)
F9CA 4C LD C,H
F9CB 7D LD A,L
F9CC B7 OR A
F9CD 2801 JR Z,F9D0
F9CF 0C INC C
F9D0 CD3DFA CALL FA3D
F9D3 1805 JR F9DA

```

Rövid ideig várakozik, hogy a magnetofon sebessége stabilizálódjon a bekapcsolás után

Bekapcsolja a hang IT-t

A bordert vörösre állítja, majd a tömb típusa szerint (FF/00:fej/adat) BC-ben 40*256 vagy pedig 20*256 (dec) periódus küldését készíti elő és hajtja végre (bevezető hang)

Utána kiküldi a szinkron-periódust és egy üres byte-ot

Betölti a CRC alapértékét HL-be, majd kiír a szalagra egy 6AH ellenőrző byte-ot

A memóriaterület kezdőcímét tölti be, majd szalagra írja a tömb típusát,

azt követően a file típusát (pufferelt vagy nem pufferelt)

Majd a file írásvédelmét jegyzi a szalagra

A kiküldendő byte-ok számának felső byte-ja a teljes szektorok száma. Ha azonban az alsó byte nem 0, akkor a maradék miatt ezt eggyel növelni kell. Kiírja a szektor számot

Ezután az adatbyte-ok következnek, ciklusban, szektoronként.

```

F9D5 D9 EXX
F9D6 2A6FOB LD HL,(0B6F)
F9D9 D9 EXX

```

A CRC számítás kezdőértéke

```

F9DA 3A310E    LD    A,(0E31)
F9DD 4F        LD    C,A
F9DE CD3DFA    CALL  FA3D
F9E1 7C        LD    A,H
F9E2 B7        OR    A
F9E3 55        LD    D,L
F9E4 2802     JR    Z,F9E8
F9E6 AF        XOR   A
F9E7 57        LD    D,A
F9E8 4A        LD    C,D
F9E9 CD3DFA    CALL  FA3D
F9EC FD4E00    LD    C,(IY+00)
F9EF FD23     INC   IY
F9F1 2B        DEC   HL
F9F2 CD3DFA    CALL  FA3D
F9F5 15        DEC   D
F9F6 20F4     JR    NZ,F9EC

```

A kiküldendő szektor sorszáma

Ennek kiírása után megnézzük, hogy van-e még teljes szektor

Ha nincs, akkor szektor byte-jainak száma a maradék lesz, egyébként pedig 256 (dec), amit 00H érték jelöl

Ezt is szalagra írjuk

Ezután egymás után kiküldjük a szektorhoz tartozó adatbyte-okat

```

F9F8 7C        LD    A,H
F9F9 B5        OR    L
F9FA 3A2B0E   LD    A,(0E2B)
F9FD 2801     JR    Z,FA00
F9FF AF        XOR   A
FA00 4F        LD    C,A
FA01 CD3DFA    CALL  FA3D

```

Az adatbyte-okat a szektor végjel követi, amely nem az utolsó szektorok esetében 00, az egész file végén: 0FFH

```

FA04 EB        EX    DE,HL
FA05 D9        EXX
FA06 4D        LD    C,L
FA07 CD3DFA    CALL  FA3D
FA0A 4C        LD    C,H
FA0B CD3DFA    CALL  FA3D
FA0E D9        EXX

```

Utána még a szalagra írjuk a szektor byte-jaiból számított ellenőrző összeget (CRC)

```

FA0F EB        EX    DE,HL
FA10 22260E   LD    (0E26),HL
FA13 FD222C0E LD    (0E2C),IY
FA17 3A310E   LD    A,(0E31)
FA1A 3C        INC   A
FA1B 32310E   LD    (0E31),A
FA1E 7C        LD    A,H
FA1F B5        OR    L
FA20 20B3     JR    NZ,F9D5

```

Tároljuk a még hátralevő, kiírandó byte-ok számát, a következő memóriacímet, eggyel növeljük a kiírandó szektor sorszámát,

s ha még van kiküldendő adatbyte, ugrás vissza

```

FA22 010105   LD    BC,0501
FA25 CD31FA    CALL  FA31
FA28 FB        EI
FA29 76        HALT
FA2A AF        XOR   A
FA2B 32320E   LD    (0E32),A
FA2E C3BDFA    JP    FABD

```

Végül még 5 periódus lezáró hangot küldünk ki,

beállítjuk az adattömb jelzést és visszakapcsoljuk a rendszer eredeti IT állapotát

Itt a fő tömbkiíró program munkáját támogató, még alacsonyabb szintű rutinok is egyszerűbbek, mint a beolvasás esetében voltak.

Négy rövid kis szubrutint kell megbeszélni.

Adott számi négyszögperiódus kiküldése

Belépéskor B és C tartalmazza az impulzusok számát: $N=B * C$ alakban (C-szer kell B számú impulzust kiküldeni)

FA31 CD5DFA	CALL FA5D	Egy jelváltás a szalagon
FA34 CD5DFA	CALL FA5D	(tehát a kettő együtt tesz ki
FA37 10F8	DJNZ FA31	egy periódust)
FA39 0D	DEC C	Ezt ismétli B-szer, s az
FA3A 20F5	JR NZ,FA31	egész C-szer
FA3C C9	RET	

Egy byte kiírása a szalagra

Belépéskor a karaktert a C regiszterben kell elhelyezni.

FA3D 0608	LD B,08	
FA3F CB09	RRC C	Az aktuális bit szerint
FA41 CD4FFA	CALL FA4F	beállítja az időzítést és egy
FA44 CB79	BIT 7,C	jelváltást ír a szalagra. CRC
FA46 CDF7FA	CALL FAF7	számítás után a bithez tartozó
FA49 CD4FFA	CALL FA4F	másik jelváltást írja ki és ezt
FA4C 10F1	DJNZ FA3F	ismétli 8-szor
FA4E C9	RET	

Az időzítés beállítása a bit értéke szerint és a jelváltás vezérlése a szalagon

Egy fél bitnek megfelelő jelet ír a szalagra, a C regiszter 7. bitje szerint.

FA4F CB79	BIT 7,C	
FA51 2805	JR Z,FA58	Ha a bit értéke 0, akkor ugrás
		előre
FA53 3AF4FA	LD A,(FAF4)	b7=1: PITCH=00DEH
FA56 180D	JR FA65	
FA58 3AF5FA	LD A,(FAF5)	b7=0: PITCH=00CEH
FA5B 1808	JR 7A65	

Az időzítés beállítása

FA5D 3AF3FA	LD	A, (FAF3)	Bevezető hang: 0D6H
FA60 1803	JR	FA65	
FA62 3AF6FA	LD	A, (FAF6)	Szinkronperiódus: 0BCH
FA65 08	EX	AF, AF	
FA66 DB58	IN	A, (58)	CTRL+ESC figyelése, s ha STOP-
FA68 E618	AND	18	ot nyomtak, megszakítja a
FA6A CAE6F8	JP	Z, F8E6	munkát
FA6D FB	EI		Megvárja a megszakításjelet, s
FA6E 76	HALT		abban a pillanatban küld egy
FA6F D350	OUT	(50), A	jelváltást a magnóra, beál-
FA71 08	EX	AF, AF	lítja a hanggenerátor aktuális
FA72 D304	OUT	(04), A	periódusidejét, alaphelyzetbe
FA74 DB5B	IN	A, (5B)	hozza, az IT-t nyugtázza, a
FA76 D307	OUT	(07), A	bordert megváltoztatja és
FA78 CD56F9	CALL	F956	
FA7B C9	RET		kész

Még három alapvető jelentőségű szubrutin maradt hátra. Különösen az első kettő tekinthető igen tanulságosnak az operációs rendszer IT programjának lekapcsolása, ill. visszakapcsolásának technikája miatt. A program készítői arra is felkészültek, hogy esetleg éppen másik IT programot szolgál ki közben a gép. A lekapcsolás és az eredeti állapot visszaállítása ebben az esetben is korrekt.

A harmadik szubrutin a TV-Computerben alkalmazott CRC számítást végzi el. Ennek számtalan, jobbnál jobb megoldása van, ám végül is egyik sem jelent tökéletes védelmet a hibás átvitelek ellen. Az itt használt elv a jobbak közül való.

Nézzük sorban, először tehát az IT kezelő rutinokat.

A hang IT bekapcsolása

A magnetofonnal megvalósult fizikai adatátvitel közben a kiszámított és változó, pontos időzítéseket, a bitek értékének megfelelő hosszúságú négyszögimpulzusok előállítását, a CPU megszakításkezelő rendszerével vezéreljük. Ezalatt a rendszer teljes megszakításkezelő programját le kell tiltani.

FA7C F3	DI		
FA7D AF	XOR	A	
FA7E D358	OUT	(58),A	Tiltja valamennyi bővítő-
FA80 D359	OUT	(59),A	kártyáról érkező megszakítást
FA82 D35A	OUT	(5A),A	
FA84 D35B	OUT	(5B),A	
FA86 D304	OUT	(04),A	PITCH alsó byte-ja: 00
FA88 3D	DEC	A	A=0FFH-val jelzi, hogy a soros
FA89 32710B	LD	(0B71),A	vonall órajel-frekvenciája
FA8C 3A110B	LD	A,(0B11)	rossz
FA8F E6F0	AND	F0	A billentyűzeten csak a CTRL és
FA91 F607	OR	07	ESC sorát figyeli
FA93 D303	OUT	(03),A	
FA95 3A120B	LD	A,(0B12)	A 05-ös port kópiájában b4=0-
FA98 E6EF	AND	EF	val tiltja a hangot
FA9A 32120B	LD	(0B12),A	
FA9D E6C0	AND	C0	b7, b6-ot meghagyva (motor-
FA9F F62F	OR	2F	vezérlés), a hang IT engedé-
FAA1 D305	OUT	(05),A	lyezése és PITCH felső 4 bit
			értéke: 1
FAA3 AF	XOR	A	
FAA4 32140B	LD	(0B14),A	"Nincs hang" jelzése
FAA7 3E0A	LD	A,0A	A képmegjelenítést vezérlő CRT
FAA9 D370	OUT	(70),A	0AH regiszterében b5=1
FAAB 3E23	LD	A,23	beállításával tiltja a kurzort
FAAD D371	OUT	(71),A	
FAAF DB5E	IN	A,(5B)	A frekvenciaosztót alap-
			helyzetbe állítja, a korábbi
FAB1 D307	OUT	(07),A	IT kérést nyugtázza, az
FAB3 213800	LD	HL,0038	eredeti IT rutin első byte-ját
FAB6 7E	LD	A,(HL)	tárolja és a helyére egyetlen
FAB7 36C9	LD	(HL),C9	C9H kódot ír (RET), majd mehet!
FAB9 32F20B	LD	(0BF2),A	
FABC C9	RET		

Ez természetesen kizárólag a magnetofonnal való adatforgalom idejére praktikus. Amint a munka befejeződött, azonnal vissza kell állítani az eredeti IT állapotot. Ez a következőképpen történik.

A rendszer IT visszaállítása

FABD F3	DI		Megállítja a magnetofon
FABE CDFFF6	CALL	F6FF	motorokat, a 03-as portra az
FAC1 3A110B	LD	A,(0B11)	eredeti biteket küldi, a 05-ös
FAC4 D303	OUT	(03),A	portra szintén
FAC6 3A120B	LD	A,(0B12)	
FAC9 D305	OUT	(05),A	
FACB 3A1F0B	LD	A,(0B1F)	A korábban tiltott IT
FACE 47	LD	B,A	forrásokat engedélyezi

FA0F CB40	BIT	0,B	
FA01 2808	JR	Z,FA0B	
FA03 3E0A	LD	A,0A	
FA05 B370	OUT	(70),A	
FA07 3E03	LD	A,03	A kurzor
FA09 0371	OUT	(71),A	
FA0B 78	LD	A,B	
FA0C 07	RLCA		
FA0D 07	RLCA		
FA0E D358	OUT	(58),A	a 3. bővítőkártya csatlakozó
FA00 07	RLCA		
FAE1 D35A	OUT	(5A),A	a 2. csatoló,
FAE3 07	RLCA		
FAE4 D359	OUT	(59),A	az 1. csatoló és végül a
FAE6 07	RLCA		
FAE7 D358	OUT	(58),A	0. csatoló IT engedélyezése -- ha korábban volt
FAE9 3AF20B	LD	A,(0BF2)	Végül visszaállítja az IT
FAEC 323800	LD	(0038),A	rutin első byte-ját, jelzések
FAEF AF	XOR	A	a hívó rutinoknak és kész
FAF0 37	SCF		
FAF1 FB	EI		
FAF2 C9	RET		

A következő négy byte-on a különböző időzítésekhez tartozó PITCH értékek alsó byte-jai vannak tárolva:

ROM-cím	FAF3	FAF4	FAF5	FAF6
Érték	D6	DE	CE	BC
Funkció	bevezető hang	1 bit	0 bit	szinkron- periódus

Az ígért harmadik nagyon fontos rutin a két byte-os ellenőrző összeg számítása. Ezt minden írási-olvasási művelet során használja a rendszer: írásnál szektoronként a szalagra kerül, visszaolvasáskor pedig a lementett és számított értékek összehasonlítása nagy biztonsággal lehetővé teszi a hibák észlelését. Nézzük ezt a rutint!

CRC számítás

Belépéskor a Z jelzőbit mutatja, hogy az éppen kezelt bit milyen értékű, HL'-ben pedig az addig számolt ellenőrzőösszeg van.

FAF7 D9	EXX	
FAF8 3E80	LD A,80	
FAFA 2001	JR NZ,FAFD	
FAFC AF	XOR A	A kezelt bit és a H b7 bitje
FAFD AC	XOR H	szerint alakul az A b7 bitje.
FAFE 17	RLA	Ha ez 0, ugrás előre
FAFF 3009	JR NC,FBOA	
FB01 7C	LD A,H	Ha b7=1 volt, akkor a H b3
FB02 EE08	XOR 08	bitjét
FB04 67	LD H,A	
FB05 7D	LD A,L	
FB06 EE10	XOR 10	és az L b4 bitjét billenti át
FB08 6F	LD L,A	
FB09 37	SCF	
FBOA ED6A	ADC HL,HL	
FBOC D9	EXX	
FB0D C9	RET	

Igen kis valószínűsége van annak, hogy eltérő byte-ok esetén az így számolt CRC ugyanazt az értéket adja, a hibás adatátvitelt a rendszer nagy biztonsággal észreveszi.

Emlékeztetem még a kedves Olvasót arra, hogy a rendszer a MUDDLE (0B6FH) rendszerváltozóba beírt kezdőértékből kiindulva képezi a szektorok ellenőrzőösszegeit. Ez, mint jelszó, az avatatlanok beolvasási kísérletétől megvédheti file-jainkat.

Ennyi volt a magnetofon kezelését végző szoftver.

3.9 Az utolsó akkordok

Közös, nagy munkánk végére értünk! Ezután semmi új dologgal nem terhelem már Önt, kedves Olvasó.

Néhány táblázat következik. Valamennyi szerepéről már beszéltünk korábban. Van az EXT ROM hátralevő részében ezenkívül néhány olyan terület, amely a rendszerinicializálás során másolódik át az U0 RAM-ba, vagy bizonyos szövegkonstansok, amelyekkel a rendszer pl. a csatolókarttyákat vagy éppen az USING formátumbyte-okat azonosítja.

Végül, itt található a definiálható karakterek mátrixai és a PRINT utasítás USING alparancsához tartozó programok egy része. Ezek szűkített listáit a megfelelő helyen természetesen közreadom, de a részletekre már ott sem térek ki. Ha szükséges, az ezekkel kapcsolatos elemző munkát -- a könyv korábbi fejezetei alapján -- már bárki elvégezheti.

Fussuk át gyorsan ezeket a dolgokat.

3.9.1 Byte-ok az U0 RAM feltöltéséhez

A ROM következő címein az input-output hozzárendelési táblázat található, az inicializáláshoz használt értékekkel (U0:0B00...0B0FH).

Osztály	Input			Output		
	ROM	Érték	Eszköz	ROM	Érték	Eszköz
0. VIDEO	FB0E	FF	video	FB16	00	video
1. BILL.	FB0F	01	bill.	FB17	FF	-
2. EDITOR	FB10	02	editor	FB18	02	editor
3. HANG	FB11	FF	-	FB19	FF	-
4. PRINTER	FB12	FF	-	FB1A	04	printer
5. MAGN.	FB13	05	magnetofon	FB1B	05	magnetofon
6. CSTL.	FB14	06	bővítő	FB1C	06	bővítő
7. KERNEL	FB15	FF	-	FB1D	FF	-

Az RST 30 utasítás és az IT kezelő belépési pont, az U0 0030...003FH területére másolódik. A programokat ott megadtuk, itt csak a byte-okat soroljuk fel.

```
FB1E  C3 23 0B 00 00 00 00 00 00 F5 3E 70 D3 02 C3 12 C4
```

A következő 26H byte az U0 0B23...0B48H címekre kerül (funkcióhívások eleje, vége és IT vége):

```
FB2E  E3 7E 23 E3 08 F5 3A 03 00 F5 3E 70 32 03 00 D3
FB3E  02 C3 63 C3 08 F1 32 03 00 D3 02 F1 08 C9 32 03
FB4E  00 D3 02 F1 FB C9
```

Néhány karakterlánc következik.

Az első négyet a csatolóártyák azonosításához használja a rendszer:

```
FB54      4D 4F 50 53      MOPS
FB58  03   56 47 42      VGB
FB5C  05   52 53 32 33 32  RS232
FB62  04   44 49 53 4B      DISK

FB67      43 49 53 4C      CISL
```

A következő nem jelentéktelen területet a definiálható karakterek mátrixa foglalja el (ékezetes betűk és félgrafikus szimbólumok):

```
FB6B  08 08 3E 6B 63 7F 63 63 00 00 08 08 7E 68 60 7C
FB7B  60 7E 00 00 08 08 3C 18 18 18 18 3C 00 00 08 08
FB8B  3E 6B 63 63 63 3E 00 00 14 00 3E 63 63 63 63 3E
FB9B  00 00 14 14 3E 63 63 63 63 3E 00 00 08 08 6B 63
FBAB  63 63 63 3E 00 00 14 00 63 63 63 63 63 3E 00 00
FBBB  14 14 77 63 63 63 63 3E 00 00 00 00 00 00 1F 1F
FBCB  18 18 18 18 18 18 18 18 1F 1F 00 00 00 00 18 18
FBDB  18 18 18 18 18 18 18 18 18 18 18 18 1F 1F 18 18
FBEB  18 18 00 00 00 00 FF FF 18 18 18 18 18 18 18 18
FBFB  FF FF 18 18 18 18 FF E7 C3 99 99 81 99 99 FF FF
FC0B  08 08 08 3C 06 3E 66 3E 00 00 08 08 08 3C 66 7E
FC1B  60 3C 00 00 08 08 00 38 18 18 18 3C 00 00 08 08
FC2B  08 3C 66 66 66 3C 00 00 00 24 00 3C 66 66 66 3C
FC3B  00 00 24 24 00 3C 66 66 66 3C 00 00 08 08 08 66
FC4B  66 66 66 3E 00 00 00 24 00 66 66 66 66 3E 00 00
FC5B  24 24 00 66 66 66 66 3E 00 00 00 00 00 00 F8 F8
FC6B  18 18 18 18 18 18 18 18 F8 F8 00 00 00 00 00 00
FC7B  00 00 FF FF 00 00 00 00 18 18 18 18 F8 F8 18 18
FC8B  18 18 18 18 18 18 FF FF 00 00 00 00 FF C3 99 9F
FC9B  9F 9F 99 C3 FF FF FF C3 99 9F C3 F9 99 C3 FF FF
FCAB  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

3.9.2 Ami a PRINT-ből kimaradt

Az EXT ROM következő byte-jain az USING alparancshoz tartozó rutinok találhatóak.

```
FCE5    EXX
FCE6    PUSH BC
FCE7    PUSH DE
FCE8    PUSH HL
FCE9    LD    HL, (172E)
FCFB    PUSH HL
FCFC    EX    (SP), IY
FCFD    XOR   A
FCE0    LD    B, A
FCE1    LD    C, A
FCE2    EXX
FCE3    LD    H, A
FCE4    LD    L, A
FCE5    LD    D, A
FCE6    LD    E, A
FCE7    LD    B, A
FCE8    LD    C, A
```

USING "+" feldolgozása

```
FCE9    LD    A, 40
FCEB    JP    C, 803E
```

USING "-" feldolgozása

```
FCEC    LD    A, 80
FCEE    JR    NZ, FD47
FCF0    PUSH AF
FCF1    OR    C
FCF2    LD    C, A
FCF3    POP  AF
FCF4    XOR  03
FCF6    AND  C
FCF7    LD    C, A
FCF8    JR    FCC9
```

USING "," feldolgozása

```
FCC9    LD    A, (IY+00)
FCCA    INC  IY
FCCB    PUSH HL
FCCD    PUSH BC
FCDD    LD    HL, FF3F
FCDE    LD    BC, 000B
FCE0    CPIR
FCE1    JR    NZ, FD45
FCE2    SLA  C
FCE3    LD    HL, FF4A
FCE4    ADD  HL, BC
FCE5    LD    A, (HL)
FCE6    INC  HL
FCE7    LD    H, (HL)
FCE8    LD    L, A
FCE9    POP  BC
FCEA    EX    (SP), HL
FCEB    LD    A, L
FCEC    OR   A
FCED    RET
```

USING "\$" feldolgozása

```
FCFA    JR    NZ, FD47
FCFC    SET  1, C
FCFE    JR    FCC9
```

USING "<" és ">" feldolgozása

```
FD00    JR    NZ, FD47
FD02    INC  L
FD03    JR    FCC9
```

USING "*" és "%" feldolgozása

```
FD05    INC  L
FD06    INC  H
FD07    EXX
FD08    LD    A, B
FD09    OR   C
FD0A    EXX
FD0B    JR    NZ, FCC9
```

```

FD0D  DEC  H
FD0E  LD   A,(IY+FF)
FD11  CP   2A
FD13  LD   E,A
FD14  JR   Z,FCC9
FD16  LD   E,30
FD18  JR   FCC9

```

Hibakezelés:

```

FD1A  LD   A,04
FD1C  LD   HL,FD5C
FD1F  JP   FFF0

```

USING "#" feldolgozása

```

FD22  INC  L
FD23  EXX
FD24  LD   A,B
FD25  OR   C
FD26  EXX
FD27  JR   Z,FCC9
FD29  INC  H
FD2A  JR   FCC9

```

USING "^" feldolgozása

```

FD2C  EXX
FD2D  LD   A,B
FD2E  OR   C
FD2F  JR   NZ,FD35
FD31  PUSH IY
FD33  POP  BC
FD34  DEC  BC
FD35  EXX
FD36  INC  D
FD37  JR   FCC9

```

USING "." feldolgozása

```

FD39  EXX
FD3A  LD   A,B
FD3B  OR   C
FD3C  JR   NZ,FD1A
FD3E  PUSH IY
FD40  POP  BC
FD41  DEC  BC
FD42  EXX
FD43  JR   FCC9

```

USING általános feldolgozás

```

FD45  POP  BC
FD46  POP  HL
FD47  EXX
FD48  LD   A,B
FD49  OR   C
FD4A  JR   NZ,FD50
FD4C  PUSH IY
FD4E  POP  BC
FD4F  DEC  BC
FD50  POP  IY
FD52  EXX
FD53  LD   (171C),HL
FD56  LD   A,D
FD57  CP   04
FD59  JR   NC,FD68
FD5B  OR   A
FD5C  JP   NZ,FD1A
FD5F  LD   A,(IY+08)
FD62  AND  7F
FD64  SUB  3F
FD66  ADD  A,H
FD67  LD   L,A
FD68  PUSH BC
FD69  LD   A,(IY+06)
FD6C  OR   A
FD6D  JR   Z,FD80
FD6F  LD   A,(IY+08)
FD72  AND  80
FD74  JR   Z,FD80
FD76  XOR  (IY+08)
FD79  LD   (IY+08),A
FD7C  POP  BC
FD7D  SET  0,C
FD7F  LD   A,C1
FD81  LD   A,(IY+08)
FD84  SUB  3F
FD86  LD   B,A
FD87  EXX
FD88  LD   E,C
FD89  LD   D,B
FD8A  EXX
FD8B  XOR  A
FD8C  LD   H,A
FD8D  LD   L,A
FD8E  OR   D
FD8F  JR   NZ,FD96
FD91  OR   B
FD92  JP   M,FD96
FD95  LD   L,B

```

FD96 EXX
 FD97 DEC DE
 FD98 OR A
 FD99 SBC HL, DE
 FD9B ADD HL, DE
 FD9C LD A, (DE)
 FD9D EXX
 FD9E JR Z, FDA2
 FDA0 JR NC, FDBB
 FDA2 CP 2C
 FDA4 JR Z, FDA7
 FDA6 SCF
 FDA7 INC D
 FDA8 DEC D
 FDA9 JR NZ, FDB6
 FDAB INC L
 FDAC DEC L
 FDAD JP M, FDB8
 FDB0 JR Z, FDB8
 FDB2 JR NC, FD96
 FDB4 DEC L
 FDB5 OR A
 FDB6 JR NC, FD96
 FDB8 INC H
 FDB9 JR FD96
 FDBB BIT 1, C
 FDBD JR Z, FDC0
 FDBF DEC H
 FDC0 BIT 0, C
 FDC2 JR NZ, FDC9
 FDC4 LD A, C
 FDC5 AND FC
 FDC7 JR Z, FDCA
 FDC9 DEC H
 FDCA BIT 7, H
 FDCC JP NZ, FD1A
 FDCF LD A, L
 FDD0 DEC A
 FDD1 JP P, FD1A
 FDD4 INC D
 FDD5 DEC D
 FDD6 JR NZ, FDEC
 FDD8 XOR A
 FDD9 BIT 7, B
 FDDB JR NZ, FDDE
 FDDD LD A, B
 FDDE LD L, A
 FDDF OR A
 FDE0 JR NZ, FDF0
 FDE2 INC H
 FDE3 DEC H
 FDE4 JR Z, FDF0

FDE6 SET Z, C
 FDE8 LD L, 01
 FDEA JR FDF0
 FDEC LD L, H
 FDED LD A, B
 FDEE SUB H
 FDEF LD B, A
 FDF0 INC D
 FDF1 DEC D
 FDF2 JR NZ, FDFA
 FDF4 INC E
 FDF5 DEC E
 FDF6 JR NZ, FDFA
 FDF8 LD E, 20
 FDFA CALL NZ, FF62
 FDFD INC H
 FDFE DEC H
 FDFE JR Z, FE14
 FE01 INC D
 FE02 DEC D
 FE03 JR NZ, FE14
 FE05 BIT Z, C
 FE07 JR Z, FE0E
 FE09 LD A, H
 FE0A DEC A
 FE0B CALL Z, FF60
 FE0E LD A, E
 FE0F CALL FF8D
 FE12 JR FDFE
 FE14 LD A, E
 FE15 CP 20
 FE17 CALL Z, FF62
 FE1A EXX
 FE1B LD E, C
 FE1C LD D, B
 FE1D EXX
 FE1E INC B
 FE1F DEC B
 FE20 JR Z, FE25
 FE22 JP P, FE29
 FE25 LD A, L
 FE26 DEC A
 FE27 JR Z, FE3F
 FE29 INC L
 FE2A DEC L
 FE2B JR Z, FE3F
 FE2D EXX
 FE2E DEC DE
 FE2F LD A, (DE)
 FE30 CP 2C
 FE32 JR Z, FE2E
 FE34 EXX

FE35	JR	FE2A	FE87	CP	10
FE37	CALL	FF96	FE89	LD	A,00
FE3A	LD	B,04	FE8B	CALL	C,FF87
FE3C	LD	C,B	FE8E	ADD	A,30
FE3D	JR	FE72	FE90	INC	B
FE3F	PUSH	DE	FE91	CALL	FF8D
FE40	LD	C,B	FE94	JR	FE76
FE41	PUSH	BC	FE96	LD	A,(BC)
FE42	LD	E,B	FE97	CP	2E
FE43	DEC	E	FE99	JR	NZ,FEE0
FE44	LD	A,(IY+08)	FE9B	CALL	FF8D
FE47	SUB	3B	FE9E	INC	BC
FE49	LD	C,A	FE9F	LD	A,(BC)
FE4A	SUB	04	FEA0	CP	2C
FE4C	LD	HL,(171C)	FEA2	JR	Z,FE9B
FE4F	DEC	H	FEA4	EXX	
FE50	ADD	A,H	FEA5	LD	A,D
FE51	JP	M,FE37	FEA6	OR	A
FE54	CP	H	FEA7	JR	NZ,FEB5
FE55	JR	NC,FE5A	FEA9	LD	A,(171D)
FE57	LD	A,H	FEAC	CPL	
FE58	LD	C,04	FEAD	CP	E
FE5A	ADD	A,04	FEAE	JR	NC,FEB5
FE5C	LD	B,A	FEB0	XOR	A
FE5D	LD	L,B	FEB1	INC	E
FE5E	DEC	B	FEB2	JP	M,FEDB
FE5F	LD	A,B	FEB5	EXX	
FE60	CP	10	FEB6	LD	A,(BC)
FE62	LD	A,00	FEB7	CP	2A
FE64	CALL	C,FF87	FEB9	JR	Z,FEC6
FE67	OR	A	FEBB	CP	25
FE68	JR	NZ,FE72	FEBD	JR	Z,FEC6
FE6A	LD	A,B	FEBF	CP	23
FE6B	DEC	A	FEC1	JR	NZ,FEE0
FE6C	CP	C	FEC3	EXX	
FE6D	JR	NC,FE5E	FEC4	JR	FED2
FE6F	LD	L,00	FEC6	EXX	
FE71	JP	NC,0C48	FEC7	LD	A,B
FE74	LD	B,04	FEC8	SUB	C
FE76	EXX		FEC9	JR	C,FED2
FE77	EX	DE,HL	FECB	OR	L
FE78	OR	A	FECC	JR	Z,FEDA
FE79	SBC	HL,BC	FECE	LD	A,F0
FE7B	ADD	HL,BC	FED0	JR	FEDA
FE7C	EX	DE,HL	FED2	LD	A,B
FE7D	JR	Z,FE96	FED3	CP	10
FE7F	LD	A,(DE)	FED5	LD	A,00
FE80	INC	DE	FED7	CALL	C,FF87
FE81	EXX		FEDA	INC	B
FE82	CP	2C	FEDB	EXX	
FE84	JR	Z,FE91	FEDC	ADD	A,30
FE86	LD	A,B	FEDE	JR	FE9B

FEE0	LD	A, (BC)	FF11	EXX	
FEE1	CP	5E	FF12	JR	FF04
FEE3	EXX		FF14	LD	A, B
FEE4	POP	BC	FF15	LD	BC, 0AFF
FEE5	POP	DE	FF18	SUB	B
FEE6	LD	B, C	FF19	INC	C
FEE7	JR	NZ, FF30	FF1A	JR	NC, FF18
FEE9	LD	A, (IY+06)	FF1C	ADD	A, B
FEEC	OR	A	FF1D	LD	B, A
FEED	JR	NZ, FEFO	FF1E	LD	A, C
FEEF	LD	B, A	FF1F	ADD	A, 30
FEFO	LD	A, 45	FF21	CALL	FF8D
FEF2	CALL	FF8D	FF24	LD	A, B
FEF5	BIT	7, B	FF25	ADD	A, 30
FEF7	LD	A, 2B	FF27	CALL	FF8D
FEF9	JR	Z, FF01	FF2A	EXX	
FEFB	LD	A, B	FF2B	INC	BC
FEFC	NEG		FF2C	INC	BC
FEFE	LD	B, A	FF2D	INC	BC
FEFF	LD	A, 2D	FF2E	INC	BC
FF01	CALL	FF8D	FF2F	EXX	
FF04	DEC	D	FF30	EXX	
FF05	LD	A, D	FF31	LD	(172E), BC
FF06	CP	04	FF35	POP	HL
FF08	JR	C, FF14	FF36	POP	DE
FF0A	LD	A, 30	FF37	POP	BC
FF0C	CALL	FF8D	FF38	EXX	
FF0F	EXX		FF39	LD	HL, E67A
FF10	INC	BC	FF3C	JP	FFF0

USING formátumbyte-ok

FF3F	2B	2D	24	3C	3E	2A	25	23	2C	5E	2E
	+	-	\$	<	>	*	%	#	,	^	.

Az egyes formátumbyte-okhoz tartozó rutincímek:

FD39	FD2C	FCC9	FD22	FD05	FD05	FD00	FD00	FCFA	FCEC	FCE9
.	^	,	#	%	*	>	<	\$	-	+

További USING segédrutinok

FF60	LD	E, 30	FF6F	LD	A, C
FF62	BIT	1, C	FF70	AND	FS
FF64	LD	A, 24	FF72	ADD	A, A
FF66	CALL	NZ, FF8D	FF73	LD	A, 20
FF69	BIT	0, C	FF75	JR	C, FF8D
FF6B	LD	A, 2D	FF77	RET	P
FF6D	JR	NZ, FF8D	FF78	LD	A, 2B

FF7A	JR	FF8D	FFAC	INC	E
FF7C	DEC	A	FFAD	INC	D
FF7D	RET	M	FFAE	INC	HL
FF7E	PUSH	AF	FFAF	LD	(172E),HL
FF7F	LD	A,20	FFB2	LD	A,(HL)
FF81	CALL	FF8D	FFB3	CP	23
FF84	POP	AF	FFB5	JR	Z,FFAD
FF85	JR	FF7C	FFB7	CP	25
FF87	PUSH	HL	FFB9	JR	Z,FFAD
FF88	LD	HL,F7CE	FFBB	CP	2A
FF8B	JR	FF91	FFBD	JR	Z,FFAD
FF8D	PUSH	HL	FFBF	LD	A,(IY+01)
FF8E	LD	HL,FE83	FFC2	SUB	D
FF91	CALL	FFF0	FFC3	NEG	
FF94	POP	HL	FFC5	JP	M,FD1A
FF95	RET		FFC8	DEC	E
FF96	LD	HL,FA14	FFC9	JP	M,FFD2
FF99	JP	FFF0	FFCC	LD	C,A

USING szövegkifejezés

FF9C	LD	HL,(172E)	FFCD	JR	NZ,FFD1
FF9F	LD	A,(HL)	FFCF	SRL	C
FFA0	LD	DE,0000	FFD1	SUB	C
FFA3	CP	3C	FFD2	PUSH	AF
FFA5	JR	Z,FFAD	FFD3	LD	A,C
FFA7	INC	E	FFD4	CALL	P,FF7C
FFA8	CP	3E	FFD7	POP	AF
FFAA	JR	NZ,FFB2	FFD8	LD	HL,E64F
			FFDB	JP	FFF0

Ezután még egy karakterlánc:

FFDE 28 43 29 31 39 38 35 49 53 4C (C)1985ISL

A soros vonal inicializálásához tartozó kis rutin:

FFE8 3E04	LD	A,04	A soros vonal sebességét 1200
FFEA C39CF2	JP	F29C	baud-ra állítja
FFED 000000	NOF		

3.9.3 A másik hídfő: vissza a SYSTEM ROM-ba

FFF0 F5	PUSH	AF
FFF1 3E70	LD	A,70
FFF3 320300	LD	(0003),A
FFF6 D302	OUT	(02),A
FFF8 F1	POP	AF
FFF9 E9	JP	HL

Az utolsó byte-ok: 7B E6 1F 1B FE 04

4. BEFEJEZÉS

4.1 Végül

Az út végére értünk.

Részletesen is megismerve a TV-Computer ROM programjának túlnyomó részét -- vajon értjük-e most a számítógép működését?

Ez így kínos kérdés -- másképpen tesszük fel: mennyire látjuk át a gépben zajló folyamatokat most, a ROM adott szintű ismerete után?

A hardverről nem kell többet mondanunk. A számítógép speciális fizikai részegységei -- a Z 80-as CPU, a memóriák, a képmegjelenítést vezérlő vagy a soros vonali USART IC, a különböző portok, az órajel- és a vonalmaghajtó áramkörök -- teszik a dolgukat. Középpontban a CPU, amely az összes eszközzel dinamikus kapcsolatot tartva, a memóriában tárolt program szerint vezérli a gépben zajló eseményeket.

Láttuk, hogy a ROM legnagyobb részét azok a szubrutinok és táblázatok teszik ki, amelyek lényegében a TV-Computer operációs rendszerét valósítják meg. A számítógép által kezelt belső és külső eszközök vezérlése a funkcióhívások egységes rendszerével van összefogva. Több száz különböző működési és logikai szintű szubrutin áll rendelkezésre ahhoz, hogy a gép funkcionális eszközeivel differenciált feladatok legyenek megszervezhetők.

A ROM-nak ez a része igen terjedelmes. Az ide tartozó táblázatok, a funkcióhívások fő vezérlőrutinja és az egyes eszközök közvetlen kiszolgálását végző programrészek a ROM-nak mintegy 40 százalékát teszik ki!

Az operációs rendszer inicializáló és megszakításokat kezelő alprogramja további 5 százalék körüli van. A ROM többi részét a gép BASIC-je foglalja el.

A TV-Computer mikroprocesszora tehát láthatóan nem marad munka nélkül.

Ha a vezérlés a BASIC-re van bízva, akkor a munka vagy a BASIC parancsok és programsorok beviteléből és szerkesztéséből áll, vagy pedig a memóriába írt, ill. betöltött BASIC programok -- a kulcsszavak és ezek köré szerkesztett kifejezések -- értelmezését és végrehajtását jelenti.

Láttuk, hogy a TV-Computer hogyan értelmezi és kezeli -- a tokenek és szimbólumok segítségével -- az ember magas szintű nyelven -- kulcsszavakból alkotott mondatokkal -- megfogalmazott utasításait.

Határozott ismereteink vannak arról, hogy a RAM tetején dinamikusan használt BASIC veremben milyen munka folyik.

A BASIC helyett a vezérlést valamely más felhasználói programra, külső modulra vagy eszközre bízva az operációs rendszer összes szolgáltatása praktikusán éppúgy igénybevehető, mint azt a BASIC teszi. Sőt, ez utóbbi által betöltött ROM területek jelentős része is felhasználható. Gondoljunk a beépített függvényekre, az RST 18 hívásokkal megvalósítható lehetőségekre, vagy pl. azokra a konverziós rutinokra, amelyek a BASIC verem és a CPU regiszterek vagy bizonyos pufferek közötti adatátalakításokra és mozgatásokra szolgálnak. Ezek a rutinok is szinte minden felhasználói programból hívhatók.

A fizikai és logikai szintek egymásra épülő, egymást feltételező hierarchikus rendjét és összefüggéseit most már bizonyos részleteiben is látva kijelenthetjük: a számítógép egy nagy bonyolultságú elektronikai és logikai rendszer. Ám éppen mert az, a hozzá értelemmel, a megértés igényével forduló ember számára jól megközelíthető, a legkülönbözőbb feladatok megoldására felhasználható, fejleszthető. A rendszer megismerésével együtt járó tanulságok pedig maradandóan kiterjeszthetők életünk akár egészen más szféráira is.

Ha az Olvasó mindezeket most hasonlóan érzi, akkor ez a könyv már részben elérte célját.

A következő felsorolás a ROM-ban elérhető rutinok nagy részét felöleli. Nem adjuk meg az egyes belépési pontokhoz tartozó részletes paramétereket. Ez azt jelenti, hogy használatuk előtt mindenképpen ajánlatos a megfelelő ROM listarész fellapozása, esetenként alapos tanulmányozása. Egyfajta iránytű akart lenni, és nem mindenkor és mindenre használható receptgyűjtemény. Ha az Olvasó akár az egész könyvet is csak iránytűként használja -- de ebben a minőségében előreviszi a munkáját -- akkor könyvünk a másik célját is elérte. Ennek reményében tárom Ön elé a fejezet hátralévő sorait.

4.2 Rutinyűjtemény

Ha egy programrészt nem szubrutinként kell hívni, hanem ugrási utasítással, akkor ezt a cím előtt J megjelöléssel tüntetjük fel.

Cím	Funkció	Oldal
0008	Hibakezelés	15
0010	Hibakezelés	16
0018	A BASIC verem műveletei	16
001B	Egy A regiszterben megadott funkcióhívás . . .	17
0030	Funkcióhívások	17
0038	Megszakításkezelő fő belépési pont	18
0B23	A funkcióhívások végrehajtásának kezdőcíme . .	28
C229 J	A RESET gomb lenyomásával egyenértékű	62
C2C2 J	Meleg reset	66
C33E	Memóriateszt, az inicializálás használja . . .	72
C363	A funkcióhívások vezérlőrutinja	76
C412	A megszakításkezelő program	83
C4B7	KERNEL 01, HI-MEM-SET, a legnagyobb memóriacím csökkentésével előírt memóriaterület biztosítása	91
C4D0	KERNEL 02, SLOT-ASN, csatolókártya hozzárendelés a 6. funkcióosztályhoz	91
C4E2	KERNEL 03, IO-ASN, input-output hozzárendelés beállítása	92
C50E	KERNEL 04, SLOT-NUM, csatoló pozíció	93
C98F	A VID kezelő rutinok általános segédprogramja .	102
C9AA	Ugrótáblakezelő	103
C9C6	HL osztása a 2 B-edik hatványával	105
C9F2	4-színű üzemmód beállítása	106
C9F4	VIDEO 04, VID-MODE, üzemmód beállítása	106
CA38	VIDEO 0C, PAL-DEF, palettaszínek	108
CA49	VIDEO 05, CLS, képernyőtörlés	108
CA65	Fizikai pontkoordinátákból VID cím és pontbitek	109
CACD	Egy pont bitjeinek átírása a VID-ben	112
CAD7	VIDEO 07, B-REL, pontpozíció relatív megváltoztatása	113
CADA	VIDEO 06, B-ABS, abszolút pontpozíció	113
CAED	Egyenesrajzoló rutin	114
CBF3	VIDEO 08, B-ON, leteszi a tollat	122

Cím	Funkció	Oldal
CBFF	VIDEO 09, B-OFF, felemeli a tollat	122
CC05	PAFER színű vonalbyte előállítása	123
CC0D	INK színű vonalbyte	123
CC10	A-színű vonalbyte	123
CC3F	Az utoljára címzett pont színkódja	125
CC86	VIDEO 02, VID-BKOUT, szöveg kiírása a képernyőre	127
CC94	VIDEO 01, VID-CHOUT, karakter kiírása a képernyőre	127
CCD7	C-kódú karakter kiírása, ellenőrzés nélkül . .	129
CD2B	Kiírás a sor elejére, VIDEO rutin	131
CD31	Soremelés VIDEO rutinja	131
CD48	VIDEO 0A, PAINT, zárt alakzatok festése	132
CF2C	VIDEO 0B, CH-DEF, megadott kódú karakter definiálása	138
CF4B	VIDEO 03, CH-POS, pontpozicionálás normál karakterpozícióra	139
CFA3	EDITOR 00, a kurzor vezérlőrutinja	142
CFD4	EDITOR INIC, az editor alaphelyzetbe állítása .	143
D013	EDITOR 04, CU-FIX, az aktuális kurzorpozíció megjegyzése	145
D01D	EDITOR 03, CU-POS, kurzor pozicionálás	145
D041	EDITOR 02, BK-IN/OUT, szövegbeolvasás, kiküldés	147
D052	EDITOR 01, CH-IN/OUT, karakterbeolvasás, karakterkiküldés	148
D124	A-kódú nyomtatható, vagy vezérlő karakterek feldolgozása	153
D191	Kurzor balra	155
D195	Kurzor fel	156
D198	Kurzor jobbra	156
D19E	Kurzor le	156
D1A8	Törlés a bekezdés végéig	156
D1CD	Kurzor a következő tabulált pozícióra	157
D1F7	Sorbeszúrás	158
D1FD	Sortörlés	158
D205	Karakterbeszúrás	159
D278	Karaktertörlés	161
D2CB	Új sor beiktatása	164
D311	Teljes sormozgatás felfelé	165
D31E	Sormozgatás megadott sortól	166
D363	Kurzor a következő sor elejére	167
D370	Kurzor az aktuális bekezdés elejére	167
D384	Kurzorkoordinátákból ASCII puffer cím	168
D39C	A kurzor sorának sorjellemezője	169
D39F	Az A-adik sor sorjellemezője	169
D3AD	Karakterkiírás, 2-es mód, editor	170
D3BC	Karakterkiírás, 4-es mód, editor	170

Cím	Funkció	Oldal
D3D9	Karakterkiírás, 16-os mód, editor	171
D420	Karakterkiírás ASCII kód alapján, editor	174
D449	PAPER és INK színű vonalbyte-ok	175
D459	Kurzor pozícióból VID cím, editor	176
D477	A kurzor helyén álló karakter tárolása	177
D491	A tárolt VID terület visszaállítása	178
D4AD	A kurzor sorának törlése a képernyőn	178
D4D3	Megadott VID kezdőcímű karakterterület törlése a képernyőn	179
D4E0	Sormozgatás felfelé a képernyőn	179
D509	Sormozgatás lefelé a képernyőn	180
D52A	Karakter elmozgatás a képernyőn	181
D542	Megadott karaktersor mozgatása balra, megadott pozíciótól, a képernyőn	182
D578	Adott karakterpozíció törlése a képernyőn	183
D592	Megadott karaktersor mozgatása jobbra, adott pozíciótól, a képernyőn	184
D5EC	A billentyűzet inicializálása	187
D612	BILL 03, van-e lenyomott billentyű?	188
D618	BILL 01, a lenyomott billentyű kódja	188
D62D	BILL 00, a billentyűzet leolvasása	189
D6C7	A billentyűzetmátrix feldolgozása	195
D790	A betűváltó billentyűk állapota	201
D7A5	A billentyűzet fizikai leolvasása	202
D906	PRINTER 02, szöveg kinyomtatása	205
D90C	PRINTER 01, karakter kinyomtatása	206
D933	HANG 00, SOUND-INT, a hangképzés folyamatának felügyelő programja	208
D961	HANG 03, TONE-SET, hang beállítása	209
D9C8	CAS 03, CAS-OPEN/CRTE, file-megnyitás	211
D9CD	CAS 04, CAS-CLOSE, file lezárás	211
D9D2	CAS 01, CAS-CHIN/OUT, karakterbeolvasás, karakterkimentés magnetofonnal	211
D9D7	CAS 02, CAS-BKIN/OUT, karakterblokk kezelése magnetofonnal	211
D9DC	CAS 05, CAS-VERIFY	211
D9E2	A magnetofon inicializálása	211
D9EF J	A BASIC teljes inicializálása	213
DA19 J	A TV-Computer bekapcsolási nyitóképe	214
DA4F J	A nyitóképek villogó "TV COMPUTER" szövege	215
DAB7 J	Kiírja a BASIC szabad terület nagyságát	216
DBBB J	REM	225
DC0C	Kiír A-számú szóközt	227
DC19	Tömörített BASIC sor előállítás	228
DD45	Adott sorszámú sor keresése	234

Cím	Funkció	Oldal
DD65 J	CONTINUE	234
DD80 J	LLIST	235
DD85 J	LIST	236
DD93 J	DELETE	236
DE08 J	NEW	239
DE1B J	RUN	239
DE31 J	TRACE	240
DE4D	A HL címen tárolt sorszám kiírása	240
DFF2 J	DATA	244
DFFC J	CLS	244
E002 J	DEF	244
E053 J	DIM	246
E104 J	ELSE	249
E10E J	END	249
E117 J	EXT	249
E15C J	FOR	250
E1CB J	INPUT	252
E21B J	READ	253
E2EE J	IF	257
E332 J	ON	258
E382 J	GOSUB	259
E3B2 J	GOTO	260
E3BC J	LET	260
E452 J	LOMEM	263
E4B6 J	NEXT	264
E542 J	OUT	266
E553 J	POKE	266
E570 J	LPRINT	267
E573 J	PRINT	267
E6C7 J	RANDOMIZE	272
E6D8	A véletlenszámok törzsrutinja	272
E6F4 J	RESTORE	272
E70F J	RETURN	273
E733 J	GRAPHICS	273
E754 J	PLOT	274
E790 J	SET	276
E833 J	SOUND	279
E89B J	CLOSE	281
E8C9 J	OPEN	281
E910 J	GET	282
E951 J	LOAD	283
E982 J	SAVE	284
E9D3 J	VERIFY	285
E9F9	File megnyitása, BASIC segédrutin	285
EA35	File-ok fejlécének beolvasása, BASIC	286
EA5A	File lezárása, BASIC	286
EA68	Zárójeles számkifejezések kiértékelése	289
EA75	Zárójeles szövegkifejezések kiértékelése	289
EAE9	ABS	291
EAF1	ATN	292

Cím	Funkció	Oldal
EBA4	COS	295
EBB5	EXP	296
EC4C	FREE	298
EC89	INKEY\$	299
ECAC	INT	300
ED07	LEN	301
ED1E	LOG	301
ED8B	Osztási maradék kiszámítása	303
EDC6	PEEK	304
EDE3	PI értékét teszi a BASIC verembe	304
EDEE	RND	305
EE3B	SGN	306
EE5B	SIN	306
EEC7	SQR	308
EF17	Hatványozás	310
EFF8	TAN	313
F013	USR	313
F03D	VAL	314
F0A7	Számkifejezések értékelése	317
F294	Szövegkifejezések értékelése	326
F35F	Új szimbólumelem láncolása	330
F3D2	Szimbólumkeresés	332
F40B	Új szimbólumok létrehozása	333
F42E	Az adat címe a szimbólumtáblában	333
F48E	Két szám kivonása a BASIC veremben	335
F493	Két szám összeadása a BASIC veremben	336
F512	Két szám szorzása a BASIC veremben	337
F5FB	Két szám osztása a BASIC veremben	342
F693	Két szám összehasonlítása a BASIC veremben	344
F6D7	Két string összehasonlítása a BASIC veremben	346
F80E	A veremben levő szám konvertálása ASCII-ra	352
F8A1	Szöveg konvertálása a verembe	354
F914	ASCII kódú számjegyek konvertálása a verembe	356
FA1B	Egy szám törlése a veremben	360
FA21	Egy szöveg törlése a veremben	360
FA2B	A HL értékét a veremben tölti	361
FAC3	A verembeli számot HL-be konvertálja	363
FB16	A verembeli számot A-ba konvertálja	365
FB82	RST 18 vezérlőrutinja	368
FD5C	BASIC hibakezelő általános belépési pont	377
FE79	Szövegkiíró rutin	380
FEBA	A veremben levő szám kiírása	381
FF19	A HL-ben tárolt szám kiírása	383
FF4F	Egy szerkesztett BASIC sor bekérése, tárolása	384
FF9D	CTRL+ESC kezelése	385

Cím	Funkció	Oldal
*F2A4	A soros vonal inicializálása	404
*F33C	SER-CHOUT, karakterkiküldés a soros vonalra . .	407
*F359	SER-CHIN, karakterbeolvasás a soros vonalról .	407
*F3BA	SER-BKOUT, blokkkiküldés a soros vonalra	409
*F3C0	SER-BKIN, blokkbeolvasás a soros vonalról . . .	409
*F3FD	CAS 03, CAS-OPEN, file megnyitása olvasásra . .	412
*F4CF	CAS 03, CAS-CRTE, file megnyitása írásra	414
*F55B	CAS 04, CLOSE, írás, a nyitott file lezárása .	416
*F566	Egy blokk felírása a szalagra	417
*F585	CAS 04, CLOSE, olvasott file lezárása	417
*F596	CAS 01, CAS-CHIN, karakter beolvasása	417
*F5C1	Egy szektor beolvasása a szalagról	418
*F605	CAS 05, CAS-VERIFY	419
*F633	CAS 02, CAS-BKIN, blokk beolvasása	420
*F653	CAS 01, CAS-CHOUT, karakter kiküldése	421
*F6AD	CAS 02, CAS-BKOUT, blokk kiküldése	422
*F6DE	Magnetofonmotorok vezérlése	423
*F707	CAS-INIC, a magnetofonkezelő munkaterület inicializálása	424
*F77B	Egy tömb fizikai beolvasása szalagról	426
*F919	Egy byte beolvasása a szalagról	432
*F92B	Olvas egy fél bitet a szalagról	432
*F938	Egy bit olvasása a szalagról	432
*F972	Tömb fizikai felírása a szalagra	433
*FA31	Adott számú négyszögjel kiküldése szalagra . .	436
*FA3D	Egy byte kiírása a szalagra	436
*FA7C	Hang IT bekapcsolása, magnetofonkezeléshez . .	437
*FABD	A rendszer IT visszakapcsolása	438
*FAF7	CRC számítás	440

A *-gal jelölt címek az EXT ROM-ban található rutinokat jelentenek. Ezek felhasználásához természetesen megfelelő memórialapozási intézkedésre van szükség.

Végül még egyszer -- nem győzöm eléggé hangsúlyozni -- az előbbi rutinok csakis megfelelő előkészítés mellett, a munkájukat alkalmassá tevő programkörnyezetben használhatók fel eredményesen.

4.3 Függelék

4.3.1 A Z 80-as mikroprocesszor

Regiszterek

	Fő (aktuális) regiszterkészlet		Másodlagos (háttér-)készlet	
(0)	A	F	A'	F'
(1)	B	C	B'	C'
(2)	D	E	D'	E'
(3)	H	L	H'	L'

A (0) sor regiszterei kitüntetett szerepet töltenek be. Az akkumulátor (A) a CPU fő műveletvégző regisztere, az F jelű regiszter pedig a műveletek eredményére jellemző jelzőbiteket tartalmaz.

Az (1)-(2)-(3) sorok regiszterei az utasítások egy részében párban használhatók, BC, DE, HL jelöléssel.

Ezek -- az ún. általános regiszterek -- 8 bitesek.

Speciális regiszterek:

8 bites: I -- megszakításvektor

R -- frissítőregiszter

16 bites: IX -- indexelt címzésre használható

IY -- indexelt címzésre használható

SP -- veremmutató

PC -- programmutató (programszámláló)

Az F regiszter bitkiosztása (jelzőbitek):

b7	b6	b5	b4	b3	b2	b1	b0
S	Z	-	H	-	P/V	N	C

S előjelbit, aritmetikai, vagy logikai műveletek után a legnagyobb helyiértékű bit állapota

Z zero-bit, a műveletek 0 eredményét, vizsgálatoknál a 0 értéket, vagy az egyenlőséget jelzi, ha értéke 1

H túlcscordulást jelez a felső és alsó digitek között (felső digit: b7...b4, alsó: b3...b0 bit). Belső CPU jelzőbit

P/V aritmetikai műveleteknél előjeles túlcsordulás, logikai műveleteknél paritás (1: páros)

N összeadásnál 1, kivonáskor 0. Belső CPU jelzőbit

C carry, átvitelt jelez

A jelzőbitek működése

Utasítás	S	Z	-	H	P/V	N	C	
ADD, ADC, SUB, SBC, (8 bites), CP, NEG	F	F		F	F	F	F	
AND	F	F		0	F	0	0	
OR, XOR	F	F		F	F	F	=	
INC, DEC (8 bites)	=	=		x	=	0	F	
ADD (16 bites)	F	F		x	F	F	F	
ADC, SBC (16 bites)	=	=		0	=	0	F	
RLA, RLCA, RRA, RRCA								
RL, RLC, RR, RRC, SLA, SRA, SRL	F	F		0	F	0	F	
RLD, RRD	F	F		F	F	=	F	
DAA	=	=		1	=	1	=	
CPL	=	=		0	=	0	1	
SCF	=	=		x	=	0	F	
CCF	F	F		0	F	0	=	
IN								*
INI, IND, OUTI, OUTD, INIR, INDR, OTIR, OTDR	x	F		x	x	1	=	**
LDI, LDD, LDIR, LDDR	x	x		0	F	0	=	***
CPI, CPIR, CPD, CPDR	F	F		x	F	1	=	****
LD A,I ; LD A,R	F	F		0	F	0	=	*****
BIT	x	F		1	x	0	=	*****

A táblázatban használt szimbólumok jelentése:

F : a jelzőbit a művelet eredménye szerint áll be
 = : változatlan
 1 : a jelzőbit értéke 1 lesz
 0 : a jelzőbit 0 lesz
 x : érdektelen

* Z=1, amikor B=00
 ** P/V=0, amikor BC=0000
 *** Z=1, ha A=(HL), ezenkívül P/V=0, amikor BC=0000
 **** P/V a megszakításengedélyező flip-flop tartalmát veszi át
 ***** Z-be a vizsgált bit ellentettje kerül

Az utasítások leírására használt szimbólumok
(mnemonikok)

Megjegyzés

- a műveletek kis betűkkel jelölt operandusai a CPU regiszterei, konkrét adatok, memória- vagy portcímek;
- az egyes utasításokban ezek nem tetszőlegesen szerepeltethetők (pl. a logikai műveletek egyik operandusa mindig az akkumulátor);
- a zárójelben álló hivatkozások mindig címet jelentenek, s az operandus ilyen esetekben az adott című memóriarekesz vagy port tartalma.

ADC	r,s	r-hez hozzáadja s-et és a CY jelzőbitet
ADD	r,s	r-hez hozzáadja s-et
AND	r	logikai "és" művelet az r és az A regiszter bitjei között
BIT	b,r	az r operandus b-edik bitjét vizsgálja. A Z jelzőbit 1 lesz, ha a bit 0
CALL	f,nn	az nn című szubrutin hívása, ha az f feltétel teljesül. A megadható feltételek: C -- CY=1 NC -- CY=0 Z -- Z=1 NZ -- Z=0 P -- S=0 M -- S=1 PO -- P/V=0 PE -- P/V=1
CALL	nn	feltétlen szubrutinhívás
CCF		a CY jelzőbitet megfordítja
CP	r	az A-val összehasonlítja r-et
CPD		az A-t összehasonlítja (HL)-lél; HL-t és BC-t csökkenti 1-gyel
CPDR		az A-t összehasonlítja (HL)-lél; HL-t és BC-t csökkenti, majd a műveletet BC=0000-ig ismétli
CPI		az A-t összehasonlítja (HL)-lél; HL-t növeli, BC-t csökkenti 1-gyel
CPIR		az A-t összehasonlítja (HL)-lél; HL-t növeli, BC-t csökkenti 1-gyel, majd a műveletet ismétli BC=0000-ig
CPL		az A bitjeit megfordítja
DEC	r	r-et csökkenti 1-gyet
DI		tiltja a maszkolható megszakításokat
DJNZ	e	a B-t csökkenti 1-gyel és relatív ugrást hajt végre e-vel, ha B nem lett 0
EI		engedélyezi a megszakításokat
EX	r,s	az r és s tartalmát felcseréli
EXX		felcseréli a BC-B'C', DE-D'E' és HL-H'L' regiszterek tartalmát

HALT		várakozás megszakításra vagy reset impulzusra
IM	n	megszakításkezelési mód beállítása (n=0,1,2)
IN	r, (c)	a C. port tartalmát r-be másolja
INC	r	r-et növeli 1-gyel
IN	A, (n)	az n. port tartalmát A-ba másolja
IND		a C. port tartalmát (HL)-be másolja; HL-t és B-t 1-gyel csökkenti
INDR		a C. port tartalmát (HL)-be másolja; HL-t és B-t 1-gyel csökkenti, majd a műveletet ismétli B=00-ig
INI		a C. port tartalmát (HL)-be másolja; HL-t növeli, B-t csökkenti 1-gyel
INIR		a C. port tartalmát (HL)-be másolja; HL-t növeli, B-t csökkenti 1-gyel, majd a műveletet ismétli B=0-ig
JP	r	feltétlen ugrás az r-rel meghatározott címre
JP	f, nn	ugrás az nn címre, ha a feltétel teljesül (a feltételeket ld. az előző oldalon)
JR	f, e	relatív ugrás az e-vel meghatározott távolságra, ha a feltétel teljesül. A feltétel most csak C, NC, Z, NZ lehet
JR	e	feltétlen relatív ugrás az e-vel meghatározott távolságra
LD	e, s	az s értékét r-be másolja
LDD		a (HL)-t (DE)-be másolja; HL-t, DE-t és BC-t csökkenti 1-1-gyel
LDDR		a (HL)-t (DE)-be másolja; HL-t, DE-t és BC-t csökkenti 1-gyel majd a műveletet ismétli BC=0000-ig
LDI		a (HL)-t (DE)-be másolja; HL-t, DE-t növeli, BC-t csökkenti 1-gyel
LDIR		a (HL)-t (DE)-be másolja; HL-t, DE-t növeli, BC-t csökkenti 1-gyel, majd a műveletet ismétli BC=0000-ig
NEG		az A (-1)-szeresét képezi
NOP		nem csinál semmit
OR	r	logikai "vagy" művelet az r és az A regiszter bitjei között
OTDR		(HL) tartalmát a C. portra küldi; HL-t és B-t csökkenti 1-gyel, majd a műveletet B=00-ig ismétli
OTIR		(HL) tartalmát a C. portra küldi; HL-t növeli, B-t csökkenti 1-gyel, majd a műveletet ismétli B=00-ig
OUT	(C), r	az r-et a C. portra küldi
OUT	(n), A	az A tartalmát az n. portra küldi
OUTD		(HL) tartalmát a C. portra küldi; HL-t és B-t 1-gyel csökkenti
OUTI		(HL) tartalmát a C. portra küldi; HL-t növeli, B-t csökkenti 1-gyel
POP	r	a veremből adatot másol vissza r-be
PUSH	r	az r tartalmát a verembe másolja

RES	b,r	az r operandus b-edik bitjébe 0-t ír (törli)
RET		feltétel nélküli visszatérés szubrutinból
RET	f	visszatérés a szubrutinból, ha a feltétel teljesül
RETI		visszatérés a megszakításkezelő rutinból
RETN		visszatérés a nem maszkolható megszakítást kezelő rutinból
RL	r	az r bitjeinek forgatása a CY jelzőbiten át, balra
RLA		az A bitjeinek forgatása a CY jelzőbiten át, balra
RLC	r	az r bitjeinek forgatása balra; a kilépő 7. bit a CY-ba is bemásolódik
RLCA		az A bitjeinek forgatása balra; a kilépő 7. bit a CY-ba is bemásolódik
RLD		az A alsó és (HL) mindkét digitjének (4-es bitcsoport) forgatása balra
RR	r	az r bitjeinek forgatása a CY jelzőbiten át, jobbra
RRA		az A bitjeinek forgatása a CY jelzőbiten át, jobbra
RRC	r	az r bitjeinek forgatása jobbra; a kilépő 0. bit a CY-ba is bemásolódik
RRCA		az A bitjeinek forgatása jobbra; a kilépő 0. bit a CY-ba is bemásolódik
RRD		az A alsó és (HL) mindkét digitjének (4-es bitcsoport) forgatása jobbra
RST	c	a c című szubrutin hívása; c=00H, 08H, 10H, 18H, 20H, 28H, 30H, 38H lehet
SBC	r,s	az r-ből kivonja s-et és a CY jelzőbitet
SCF		CY=1-et állít be
SET	b,r	az r operandus b-edik bitjét 1-re állítja
SLA	r	az r bitjeit balra lépteti; a kilépő 7. bit a CY-ba másolódik, a 0. bit 0 lesz
SRA	r	az r bitjeit jobbra lépteti; a kilépő 0. bit a CY-ba másolódik, a 7. bit nem változik
SRL	r	az r bitjeit jobbra lépteti; a kilépő 0. bit a CY-ba másolódik, a 7. bit 0 lesz
SUB	r	az A regiszterből kivonja r-et
XOR	r	logikai "kizáró vagy" művelet az A regiszter és az r bitjei között

4.3.2 Hexadecimális számok

A 16-os számrendszerben használt számjegyek és tízes számrendszerbeli megfelelőjük:

HEX: 0 1 2 3 4 5 6 7 8 9 A B C D E F
DEC: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Átszámítási segédlet:

H E X A D E C I M Á L I S								S Z Á M J E G Y E K							
4				3				2				1			
magasabb helyiértékű byte								alacsonyabb helyiértékű byte							
(b7	b6	b5	b4	b3	b2	b1	b0)	(b7	b6	b5	b4	b3	b2	b1	b0)
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC		
0	0	0	0	0	0	0	0	0	0	0	0	0	0		
1	4096	1	256	1	16	1	1	1	16	1	1	1	1		
2	8192	2	512	2	32	2	2	2	32	2	2	2	2		
3	12288	3	768	3	48	3	3	3	48	3	3	3	3		
4	16384	4	1024	4	64	4	4	4	64	4	4	4	4		
5	20480	5	1280	5	80	5	5	5	80	5	5	5	5		
6	24576	6	1536	6	96	6	6	6	96	6	6	6	6		
7	28672	7	1792	7	112	7	7	7	112	7	7	7	7		
8	32768	8	2048	8	128	8	8	8	128	8	8	8	8		
9	36864	9	2304	9	144	9	9	9	144	9	9	9	9		
A	40960	A	2560	A	160	A	10	A	160	A	10	A	10		
B	45056	B	2816	B	176	B	11	B	176	B	11	B	11		
C	49152	C	3072	C	192	C	12	C	192	C	12	C	12		
D	53248	D	3328	D	208	D	13	D	208	D	13	D	13		
E	57344	E	3584	E	224	E	14	E	224	E	14	E	14		
F	61440	F	3840	F	240	F	15	F	240	F	15	F	15		

Egy adott hexadecimális szám 10-es számrendszerbeli értékét a helyi érték szerint feltüntetett decimális számok egyszerű összeadásával kapjuk.

Pl.: 0EFH=224+15=239 (dec.)

4.3.3 Portok

PORT 00		output						(0B4FH)
b o r d e r s z í n								
b7	b6	b5	b4	b3	b2	b1	b0	
I	-	G	-	R	-	B	-	

PORT 02		output						(0003)
m e m ó r i a l a p o z á s								
b7 -- b6	b5	b4 -- b3	b2	b1	b0			
3.LAP	2.LAP	0.LAP	1.LAP	-	-			
00 CART		00 SYS						
01 SYS	0 VID	01 CART	1 U1					
10 U3	1 U2	10 U0						
11 EXT								

PORT 03		output						(0B11)
I/O memória kiválasztás		billentyűzet sorkiválasztás						
b7 -- b6	b5	b4	b3	b2	b1	b0		
00 CSTL0								
01 CSTL1	-	-	a tíz közül kiválasztott egyik sor sorszáma					
10 CSTL2								
11 CSTL3								

PORT 04		output					
P I T C H , a l s ó 8 b i t							
b7	b6	b5	b4	b3	b2	b1	b0

PORT 05		output						(0B12)
! magnetofon !		! hang !		P I T H , f e l s ő				!
! motorvezérlés !		! IT ! jel !		4 b i t				!
b7	b6	b5	b4	b3	b2	b1	b0	
jobb	bal	0 KI	0 KI					
		! 1 BE ! 1 BE !						

PORT 06		output						(0B13)
! nyomtató !		- !		h a n g		! grafikus !		
! STROBE !				a m p l i t u d ó		! üzemmód !		
b7	b6	b5	b4	b3	b2	b1	b0	
						! 00 2 színű !		
0 aktív	-	(0...15) * 4				! 01 4 színű !		
						! 1x 16 színű !		

PORT 07		output					
! k u r z o r / h a n g I T n y u g t á z á s a							
b7	b6	b5	b4	b3	b2	b1	b0
a z		í r t		a d a t		k ö z ö m b ő s	

PORT 10,20,30,40		output / input	
! a d a t f o r g a l o m a s o r o s v o n a l o n			
! (a négy portcím a 0..1..2..3. csatlakozóba dugott			
! soros vonali kártyákra vonatkozik)			

PORT 11,21,31,41		(0-1-2-3 CSTL)		output			
! üzemmódkiválasztás belső RESET parancs után							
b7	b6	b5	b4	b3	b2	b1	b0
STOP bitek		paritás		karakter-		! órajel-	
száma				hossz		! -viszony	
00	-	x0	nincs	00	5 bit	! 00 szinkron	
01	1			01	6 bit	! 01 1-szeres	
10	1.5	01	páratlan	10	7 bit	! 10 16-szeres	
11	2	11	páros	11	8 bit	! 11 64-szeres	

PORT 11,21,31,41 (0-1-2-3 CSTL) output

általános parancskiküldés a soros vonalra

b7	b6	b5	b4	b3	b2	b1	b0
	belső	RTS	hiba	BREAK	vétel	DTR	adás
0	RESET	BE	törl	küld.	eng.	BE	eng.

PORT 11,21,31,41 (0-1-2-3 CSTL) input

állapobeolvasás a soros vonalról

b7	b6	b5	b4	b3	b2	b1	b0
DSR	BREAK	keret	túl	pari-	adó	kar.	adó
	vétel	hiba	futás	tás	puf.	jött	me-
				hiba	üres		het

PORT 50 output

magnetofon kimenőjel felvételhez

b7	b6	b5	b4	b3	b2	b1	b0
	a	z	í	r	t	a	d
							a
							k
							ö
							m
							b
							ö
							s

PORT 58,59,5A,5B (0-1-2-3 CSTL) output

csatoló							
IT							
b7	b6	b5	b4	b3	b2	b1	b0
0 tilt.							
1 eng.	-	-	-	-	-	-	-

PORT 58 input

a billentyűzet kiválasztott sorának beolvasása

b7	b6	b5	b4	b3	b2	b1	b0
----	----	----	----	----	----	----	----

PORT 59 input

nyomtató ACK								
szín- kapcs		magne- tofon input		függőben levő IT kérések kurzor csatolókérttyák hang 3 2 1 0				
b7	b6	b5	b4	b3	b2	b1	b0	
0 ff.								
0 aktív		1 sz.		a bitek 0 értéke aktív				

PORT 5A input

c s a t o l ó k á r t y á k a z o n o s í t ó i								
b7 - - b6		b5 - - b4		b3 - - b2		b1 - - b0		
3.cstl.		2.cstl.		1.cstl.		0.cstl.		
00: RS232 soros vonal; 01: VGB (játék); 10: floppy; 11: üres, azonosítatlan								

PORT 5B input

a hanggenerátor frekvenciaosztóját alaphelyzetbe hozza								
b7	b6	b5	b4	b3	b2	b1	b0	

PORT 60,61,62,63 output (pal.0-1-2-3)

p a l e t t a s z í n								
b7	b6	b5	b4	b3	b2	b1	b0	
-	I	-	G	-	R	-	B	

PORT 70 output

- ! CRT regiszter kijelölés								
b7	b6	b5	b4	b3	b2	b1	b0	

PORT 71 output

C R T r e g i s z t e r a d a t								
---------------------------------------	--	--	--	--	--	--	--	--

4.3.4 Rendszerváltozók, munkarekeszek, táblázatok

0003	P-SAVE, memórialapozás
0021--002E	USR-TAB, az EXTn címek táblája
0040--00FF	I/O puffer, 4x30H a csatolókarttyák számára
0100--06FF	ASCII képernyő az editor számára
0740--0AFF	A definiálható karakterek mátrixai
0E00--0E0F	Input-output hozzárendelési tábla
0E10	INT-DES, a CU/hang IT által kiszolgálendő eszközök
0E11	PORT03, a 03-as port kópiája
0E12	PORT05, a 05-ös port kópiája
0E13	PORT06, a 06-os port kópiája
0E14	Hang van folyamatban
0E15	Az új hang megszakítja a régit
0E16	STOP, CTRL+ESC lenyomása
0E17--0E18	A verem alsó határa
0E19--0E1A	HI-MEM, a használható legnagyobb memóriacím
0E1B	U3-STAT, az U3 RAM állapota
0E1C	Bővítőkarttya alaphozzárendelés
0E1D--0E1E	TIME, IT-számláló
0E1F	IRQ-STAT, az IT források engedélyezése
0E20	INT-FLAG, IT végrehajtás van folyamatban
0E21	WARM-FLAG, meleg reset van folyamatban
0E22	COLD-FLAG, hideg reset kell
0E4B	L-MODE, vonalkeresztezési mód
0E4C	L-STYLE, vonaltípus
0E4D	INK, aktuális tintaszín
0E4E	PAPER, aktuális alapszín
0E4F	BORDER, aktuális keretszín
0E50	V-FLAG, karakterfelülírási mód
0E51--0E5A	PICTURE, a billentyűzet aktuális állapota
0E5B--0E64	OLD-PICT, a billentyűzet előző állapota
0E65	DELAY-KEY, billentyűkészleltetés
0E66	LOCK-KEY, a billentyűzet LOCK-állapota
0E67	RATE-KEY, billentyűismétlési ütem
0E68	HOLD-DIS, a CTRL+P hatás tiltása
0E69	BAUD, a soros vonal sebessége
0E6A	FORMAT, a soros vonali USART üzemmódja
0E6B	BUFFER, file-típus
0E6C	REMRED, motorvezérlés
0E6D	PROTECT, írásvédelem
0E6E	EOF, file vége
0E6F--0E70	MUDDLE, CRC kezdőérték
0E71	A soros vonal órajele rossz
0E73	Üzem mód
0E74	A grafikus toll állapota
0E75	Az aktuális VID byte-ba töltendő érték
0E76--0E77	Az aktuális VID byte címe
0E78--0E79	PLOT-hoz tartozó logikai x koordináta
0E7A--0E7B	PLOT-hoz tartozó logikai y koordináta
0E7C--0E7D	PLOT-hoz tartozó fizikai x koordináta

0B7E--0B7F	PLOT-hoz tartozó fizikai y koordináta
0B83	Az L-STYLE szerinti vonaltípus bitképe
0BE5	Van lenyomva billentyű
0BE7	LOCK lenyomása
0BE8	Üzemváltó billentyűk lenyomása
0BE9	A lenyomott billentyű kódja
0BEA	Számláló a billentyűismétlés késleltetéshez
0BEB	Számláló a billentyűismétlés üteméhez
0BEC--0BED	Az aktuálisan kezelt billentyű OLD-PICT címe
0BEE	Az aktuálisan kezelt billentyűhöz tartozó bit
0BEF	Számláló a hangképzés időtartamához
0BF0	Borderszín a magnetofon-műveletek során
0BF1	VERIFY funkció jelzése
0BF2	Az IT alprogram első byte-jának tárolása
0BF3	Olvasásra nyitott file típusa
0BF4--0C04	A kért file neve
0C05--0C15	A file neve a szalagról
0C05--0D04	Input puffer a file-ok beolvasásához
0D05--0D06	A szalagról beolvasott byte-ok száma
0D07--0D08	A következő byte címe a pufferben
0D09--0D0A	A szalagról beolvasandó byte-ok száma
0D0B	Hibakód
0D0C	A beolvasott file írásvédelme
0D0D	A beolvasott szektor sorszám
0D0E	Szektorvégjel
0D0F	Tömbtípus
0D10--0D11	A beolvasott adatok kezdőcíme a memóriában
0D13	File megnyitása olvasásra
0D14	Írásra megnyitott file típusa
0D15--0D25	Megadott file-név
0D26--0E25	Output puffer a file-ok tárolásához
0E26--0E27	A tárolandó byte-ok száma
0E28--0E29	A következő tárolandó byte címe
0E2A	Hibakód
0E2C--0E2D	A tárolandó memóriaterület kezdőcíme
0E2E	A tárolandó byte
0E2F	A tárolandó file típusa
0E30	A tárolandó file írásvédelme
0E31	Az aktuális szektor sorszám
0E32	File megnyitási folyamat, írásra
0E48	Számláló a kurzor villogtatásához
0E49	A kurzor sorszám
0E4A	A kurzor soron belüli pozíciója
0E4B--0E4C	A kurzor ASCII puffer-címe
0E4D	A kurzor pozicionálásával kapcsolatos jelzés
0E4E--0E4F	A kurzorpozíció tárolása
0E50--0E67	Sorjellemzők, a sorok foglaltsága a képernyőn
0E69--0E6A	Az üzemmódnak megfelelő megjelenítő rutin címe
0E6B	Aktuális sorhossz
0E6C	Aktuális karakterszélesség
0E6D--0E94	A kurzor helyén álló karakter tárolása
0E95	INK színű vonalbyte

OE96	PAFER színű vonalbyte
OEAC-16AB	Processzor verem
1700	BASIC üzemmódjellemző
1701	Változó típusbyte
1702	Feltételes BASIC utasítás jelzése
1704	Az RST 18-hoz tartozó aktuális utasításbyte
1705	Aktuális funkcióosztály
1706	Az utasításban meghatározott periféria
1707	AUTURUN jelzése
1708	A keresett változó típusa
1709	RND változó
170A-170B	RND változó
170C-170D	Az aktuális BASIC sor kezdőcíme
170E-170F	A következő BASIC sor címe
1710-1711	A következő BASIC utasítás címe
1712-1713	DATA sormutató
1714-1715	DATA adatmutató
1716-1717	INPUT adatmutató
1718-1719	A következő RST 18 utasításbyte címe
171A-171B	IY értéke egy BASIC utasítás kezdetén
171E-171F	A HL aktuális értéke RST 18 végrehajtásakor
1720-1721	VLOMEM, a BASIC terület kezdete
1722-1723	TEXT, a BASIC program kezdőcíme
1724-1725	CHAIN, a szimbólumtábla utolsó elemének címe
1726-1727	TOP, a következő szabad byte címe a szimbólumtáblában PITCH tárolása
172A-172B	COMMAND, puffer az aktuális BASIC parancshoz
1732-1830	INPUT puffer a BASIC parancsok begépeléséhez
1831-1930	File-név puffer
19CE-19DE	Fejléc-puffer
19DF-19EE	A standard BASIC báziscím
19EF	Elsődleges tokenek ugrótáblája
C067-C0C6	RST 18 műveletek ugrótáblája
C0C7-C0E4	Kernel ugrótábla
C4AC-C4B6	Eszközinitializáló rutinok ugrótáblája
C555-C55C	A funkcióhívások ugrótáblája
C55D-C56C	Az állandó karakterek mátrixai
C5B4-C973	Video ugrótábla
C974-C98E	Editor ugrótábla
CF98-CFA2	Billentyűzet ugrótábla
D5E3-D5EB	Billentyűzet kódtáblázatok
D7BF-D8FE	Nyomtató ugrótábla
D8FF-D905	Hang ugrótábla
D92A-D932	Magnetofonkezelő ugrótábla
D9BB-D9C7	Kulcsszótáblázat a tokenek azonosításához
DE6D-DFF1	Az első beépített szimbólum címe
F094	Hibüzenetek
FDC6-FE78	Soros vonal ugrótáblája (EXT)
*F291-F29B	A definiálható karakterek mátrixa (EXT)
*FB6B-FCAA	

Z Á R S Z Ó

A könyv elején egy utazásra, tanulságos kalandra, az alkotás általános emberi csodájának átélésére invitáltam az Olvasót.

Nyilvánvaló, hogy még e könyv legutolsó lapja után is maradnak szép számmal megválaszolatlan problémák. Kár is lenne -- szerencsére nem is lehet -- egy fantasztikusan érdekes, változatosan és dinamikusan mozgó világot egyszer, s mindenkorra lezárni.

Utazásunk célja, ha úgy tetszik, éppen ezzel ellentétes volt. Kérdéseknek akár százait akartam Önben megfogalmaztatni. A TV-Computerben megadott feltételek között arra biztatom, hogy a felkínálkozó lehetőségeknek járjon utána. Programozói munkájában, a TV-Computerhez és más számítógépekhez való viszonyában keresse mindig az újat, az előrevivőt, az egyszerű és alkotóerejű gondolatokat és kívánom, hogy azokat meg is találja!

Ha e könyv révén mindehhez akár szerény mértékben is hozzájárulhattam, munkám már nem volt hiábavaló. Köszönöm, hogy Ön, kedves Olvasó, velem tartott!

TARTALOMJEGYZÉK

MINDENEKEELŐTT.....	5
1. AZ ISMERKEDÉS ALAPSZINTJE: AMIT A HARDVERRŐL TUDNI KELL.....	7
1.1 A mikroprocesszor.....	7
1.2 A memória.....	8
1.3 A képmegjelenítő hardver.....	9
1.4 A billentyűzet.....	10
1.5 A magnetofon kezelése.....	10
1.6 A nyomtató vezérlése.....	11
1.7 A soros vonali hardver.....	11
1.8 A hanggenerátor.....	12
2. AZ ISMERKEDÉS KÖVETKEZŐ SZINTJE: ELŐKÉSZÜLETEK.....	13
2.1 A memória eleje és néhány Z 80-as sajátosság.....	13
2.2 Az U0 RAM funkciói.....	14
2.3 Az U0 RAM eleje.....	15
2.4 A funkcióhívásokról.....	21
2.5 A rendszerváltozók.....	24
2.6 A nagy gondolat: ROM program a RAM-ban.....	28
2.7 A video rendszerváltozók.....	29
2.8 Még egyszer a billentyűzetről.....	31
2.9 A soros vonal két byte-ja.....	33
2.10 A magnetofonkezelés rendszerváltozói.....	34
2.11 A munkaterületek.....	37
2.12 Néhány BASIC munkaváltozó.....	44
2.13 Az előkészületek vége.....	46
3. A TV-COMPUTER ROM PROGRAMJA.....	47
3.1 Nyitány.....	47
3.1.1 A gép intelligenciája.....	47
3.1.2 A start.....	49
3.1.3 A szorzótáblák.....	51
3.1.4 BASIC előzetes.....	52
3.1.5 A BASIC veremről.....	54
3.1.6 Ismerkedés az RST 18 elemeivel.....	56
3.1.7 Számkonstansok.....	58
3.2 A rendszerinicializálás.....	61
3.2.1 Az első gombnyomás előtt.....	61
3.2.2 Az inicializálás.....	62
3.3 A funkcióhívások kezelése.....	74
3.3.1 Bevezetés.....	74
3.3.2 A funkcióhívások vezérlése.....	75

3.3.3	Az ugrótáblák.....	81
3.4	A megszakításkezelő főprogram.....	83
3.5	A funkcióhívások végrehajtása.....	89
3.5.1	A kernel funkciók.....	89
3.5.2	Az ugrótáblák.....	94
3.5.3	A video eszközmeghajtó rutinjai.....	95
3.5.3.1	A blokkfunkciók vezérlése	96
3.5.3.2	A fix karakterek definíciói	99
3.5.3.3	A video ugrótábla	102
3.5.3.4	Videorutinok	102
3.5.4	Az editor.....	141
3.5.4.1	Az editor ugrótábla	142
3.5.4.2	A vezérlőrutinok	142
3.5.4.3	A segédrutinok	168
3.5.4.4	A videomemória rutinok	169
3.5.5	A billentyűzet.....	186
3.5.5.1	A billentyűzetkezelés technikája	186
3.5.5.2	Az ugrótábla	187
3.5.5.3	A billentyűzet rutinjai	187
3.5.5.4	Kódtáblázatok	202
3.5.6	A nyomtató.....	204
3.5.6.1	A vezérlés folyamata	204
3.5.6.2	A nyomtató ugrótábla	205
3.5.6.3	A nyomtató rutinok	205
3.5.7	Hangképzés.....	207
3.5.7.1	A hangfunkciók ugrótáblája	207
3.5.7.2	A hangképzés szubrutinjai	207
3.5.8	Előzetes a magnetofonkezelő programokhoz.....	210
3.5.9	Összefoglalás.....	211
3.6	BASIC.....	213
3.6.1	Kezet nyújt a BASIC.....	213
3.6.2	A magas szintű parancsok fogadása.....	217
3.6.3	A programsorok beillesztése.....	220
3.6.4	Az utasítások végrehajtásának vezérlése.....	222
3.6.5	Az elsődleges tokenek rutinjai.....	227
3.6.5.1	A segédprogramok	228
3.6.5.2	A kulcsszavak	234
3.6.5.3	Kulcsszavak táblázata	241
3.6.5.4	Kulcsszavak rutinjai	244
3.6.6	Beépített függvények.....	287
3.6.6.1	Bevezetés	287
3.6.6.2	Hasznos segédrutinok	289
3.6.6.3	Belső szimbólumok	291
3.6.7	Alapműveletek és konverziók.....	316
3.6.7.1	Alacsony szintű rutinok	316
3.6.7.2	Még néhány "apróság"	366

3.6.8	Hibakezelés	376
3.6.8.1	Bevezetés	376
3.6.8.2	A hibakezelő program	377
3.6.8.3	Hibaüzenetek	379
3.6.9	Az utolsó segédrutinok	380
3.6.10	Híd az EXT ROM-ba	386
3.6.11	Mögöttünk a BASIC	388
3.7	Ami a rendszer ROM fő részéből kimaradt	391
3.7.1	Adalék a rendszer inicializálásához	391
3.7.2	Adalék a funkcióhívásokhoz	397
3.7.3	Adalék az IT alprogramhoz	401
3.8	Még két eszközmeghajtó	402
3.8.1	Bevezetés	402
3.8.2	A soros vonal	403
3.8.2.1	A soros vonal ugrótáblája	404
3.8.2.2	Rutinok	404
3.8.3	A magnetofon kezelése	410
3.8.3.1	Bevezetés	410
3.8.3.2	A funkciók vezérlése	412
3.8.3.3	A magnetofonkezelés alacsony szintű rutinjai	423
3.9	Az utolsó akkordok	441
3.9.1	Byte-ok az U0 RAM feltöltéséhez	441
3.9.2	Ami a PRINT-ből kimaradt	443
3.9.3	A másik hídfő: vissza a SYSTEM ROM-ba	448
4.	BEFEJEZÉS	449
4.1	Végül	449
4.2	Rutingyűjtemény	451
4.3	Függelék	457
4.3.1	A Z 80-as mikroprocesszor	457
4.3.2	Hexadecimális számok	462
4.3.3	Portok	463
4.3.4	Rendszerváltozók, munkarekeszek, táblázatok	467
ZÁRSZÓ	470

Zalai Nyomda Zalaegerszeg 88 3900 o

290,- Ft

VIDEOTON

**ELEKTRONIKAI VÁLLALAT
SZÁMÍTÁSTECHNIKAI GYÁRA**