Angerhausen - Brückmann
Englisch - Gerits

A Commodore 64 belsö felépítése

DATA BECKER - NOVOTRADE

Angerhausen · Brückmann
Englisch · Gerits

A Commodore 64-es belső felépítése

DATA BECKER – NOVOTRADE

A mű eredeti címe: 64 Intern (1984)

Fordította: DOBOSNÉ HARTYÁNYI MÁRIA
Lektorálta: DR. OBÁDOVICS J. GYULA és DR. LENGYEL JÓZSEF
A programokat lektorálta: SZERENCSI RÓBERT

A kiadásért felel: RÉNYI GÁBOR, a NOVOTRADE RT. igazgatója
Kidványmenedzser: BÉKÉS TAMÁS
Sorozatszerkesztő: ROCHLITZ ANDRÁS
Felelős szerkesztő: TARR KÁLMÁNNÉ és ROCHLITZ ANDRÁS
Műszaki szerkesztő: DÉVÉNYI ERIKA

Szedte a PREPRINT GMK, BUDAPEST
Készült a Nyírségi Nyomdában, Nyíregyházán (25 A/5 ív)
Budapest, 1985.
ISBN 963 02 40351

FONTOS TUDNIVALÓ

A jelen könyv keretén belül ismertetett kapcsolások, eljárások és programok nem tekintendők szabadalmi oltalom alá eső ipari termékeknek. Ezek elsősorban amatőr és oktatási célokat szolgálnak. A szerzők rendkívül nagy gondot fordítottak a kapcsolások, műszaki adatok és programok helyességére, a részletek kidolgozása során többszöri ellenőrzést végeztek. Mindez azonban nem zárja ki az esetleges hibalehetőségeket.
Az előforduló hibákért és az ebből adódó következményekért a DATA BECKER cég sem szavatosságot, sem jogi felelősséget nem vállal. Az esetlegesen előforduló hibák közlését a szerzők hálásan fogadják.

TARTALOMJEGYZÉK

ELŐSZÓ

A Commodore 64 egy szupergép — ez azonnal kiderült, amikor először kezdtünk vele dolgozni. Már '82 nyarán megrendeltünk egy készüléket az USA-ból, hogy az első tapasztalatokat megszerezzük a COMMODORE cég új szuperprodukciójáról.
Ezt követően karácsony előtt megérkezett az NSZK-ba az első Commodore 64-es szállítmány, — amely akkor, ugyanúgy mint az elmúlt év karácsonyán, messze alatta volt a keresletnek. Az első napoktól kezdve a Commodore 64-es abszolút sláger volt a piacon. Kimagaslóan jó minőségével teljesen megváltoztatta a számítógéppiac ár-teljesítmény viszonyát. A Commodore 64-es csak egyetlen ponton jelentett gondot az ujdonsült tulajdonosok számára: a gép kezeléséhez, programozásához nyújtott információ mondhatni egyenesen silány volt. Az olyan fontos dolgokról, mint pl. a grafika- vagy a zeneprogramozás, gyakorlatilag semmit sem lehetett megtudni a közkézen forgó kézikönyvekből, amelyek ráadásul angol nyelven íródtak.
Így határozta el a DATA BECKER cég, hogy egy kézikönyv sorozatot indít el a Commodore 64-esről, amelynek ez a könyv az első kötete.
Természetesen a könyv első kiadása óta mi magunk is sok újat tanultunk a Commodore 64-esről.
Ez az oka annak, hogy az első kiadás után másfél évvel az Olvasó a 4. bővített kiadást tarthatja kezében. A magyar fordítás is ennek alapján készült.
Az az elképzelésünk, hogy a könyv megírásában több szerző működjön közre, nagyon jól bevált.
A szerzők mindegyike arról írhat, ami őt a legjobban érdekli, és amit a legjobban ismer:
Michael *Angerhausen,* a DATA BECKER cég szoftvercsoportjának tagja, elsősorban a sprite-okkal és a grafikával foglalkozik.
Rolf *Bückmann,* legtapasztaltabb technikusunk a Commodore 64-es belsejét tárja az Olvasó elé, és közérthetően magyarázza a könyvben található kapcsolási rajzokat.
Lothar *English,* aki főként a rendszerprogramozás szakembere, a 4. kiadásban mégegyszer átdolgozta a gépi kódú programozásban olyan fontos ROM listát, továbbá bemutatja Olvasóinknak a BASIC nyelv bővítési lehetőségeit.
Végül, de nem utolsó sorban Klaus *Gerits,* a DATA BECKER fejlesztő csoportjának vezetője, a Commodore 64-es hardver-felépítésével foglalkozik mélyrehatóan. A 4. kiadásban ujdonságot jelent az Olvasók számára az IEC buszról szóló, általa készített cikk.
Reméljük, hogy ez a 4. kiadás is hozzájárul ahhoz, hogy az Olvasók mind jobban élhessenek a gép által nyújtott gazdag lehetőségekkel.

Dr. Achim Becker

1. FEJEZET
Lemezvégen a CBM 64-es

1.1 Amit a készülékről tudni kell

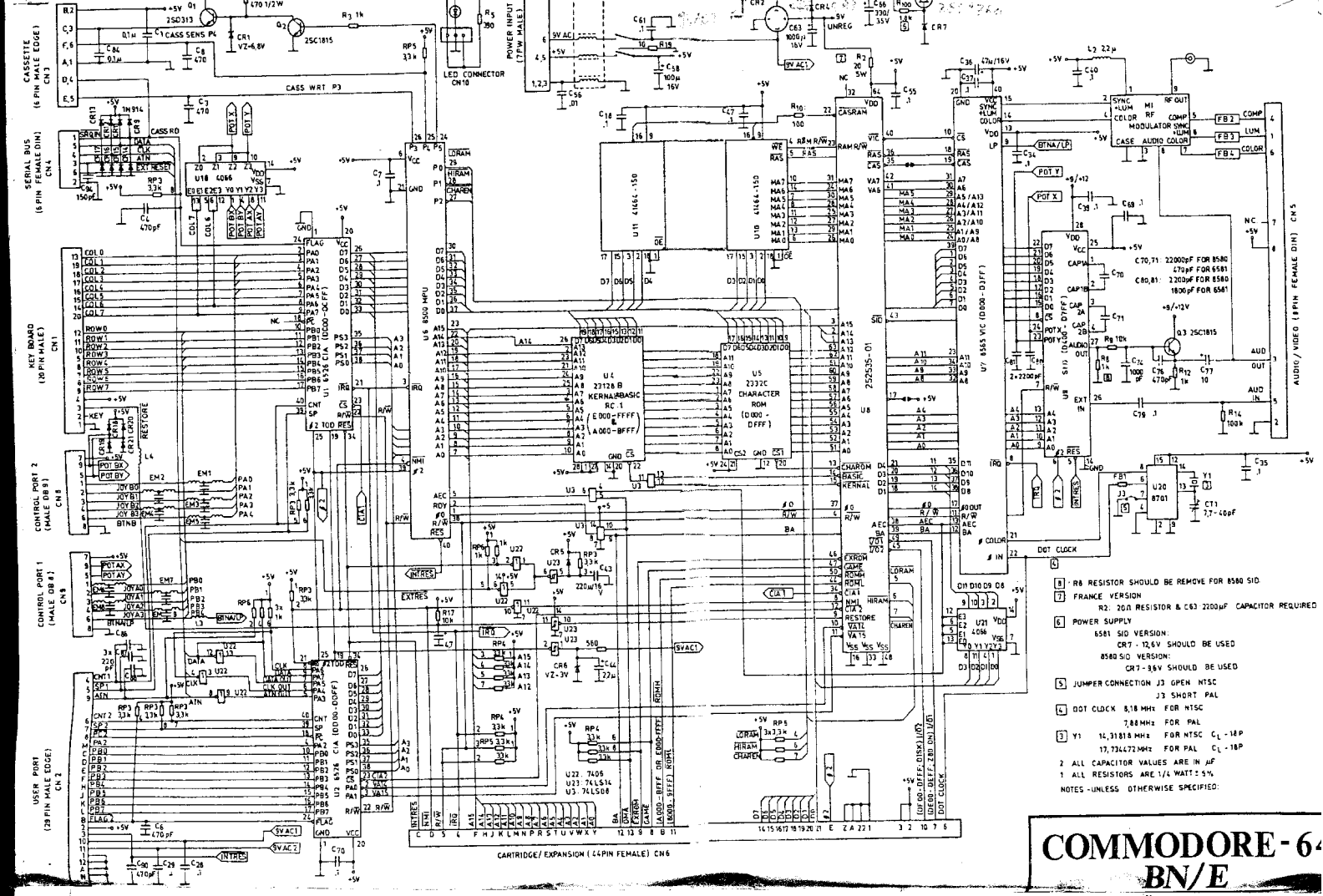
A CBM 64-es minden kétséget kizáróan egy nagyszerű alkotás, és bátran állíthatjuk, hogy a befektetett munkát sokszorosan megtéríti.
Munka közben mindenki hamarosan gyanakodni kezd, hogy a gép hardverét illetően valami titok lappanghat.
Akinek van VC 20-as gépe is, ne sajnálja a fáradtságot és kapcsolja be mindkét gépet egyszerre. Gyorsan megállapíthatjuk ugyanis, hogy a CBM 64-es kevesebb IC-t tartalmaz, mint a VC 20-as, és mégis nagyobb teljesítményt nyújt. Ez csak úgy lehetséges, hogy a CBM 64-esbe beépített IC-k magasabb integráltságúak.
A CBM 64-es tényleg tartalmaz egy sor újonnan kifejlesztett nagyintegráltságú IC-t.
A COMMODORE cég abban a szerencsés helyzetben van, hogy leányvállalatként (MOS) rendelkezik egy félvezetőgyártó részleggel.
Nehezen fogják tudni elképzelni, hogy hogyan lehetséges egy egykártyás számítógéppel (Single Board Computerrel) ilyen szintű kapcsolatot teremteni a külvilággal.
A gép gyártóinak sikerült megvalósítani egy 8 bites mikroprocesszorral és a 64 kbyte-os címtartománnyal a következőket:

- 64 kbyte dinamikus RAM
- 1 kbyte szín RAM
- 4 kbyte karaktergenerátor
- színes videochip, finom felbontású (Hi-Res) grafikával
- háromcsatornás szintetizátor
- 8 kbyte BASIC interpreter
- 8 kbyte operációs rendszer ROM
- 2 párhuzamos I/O

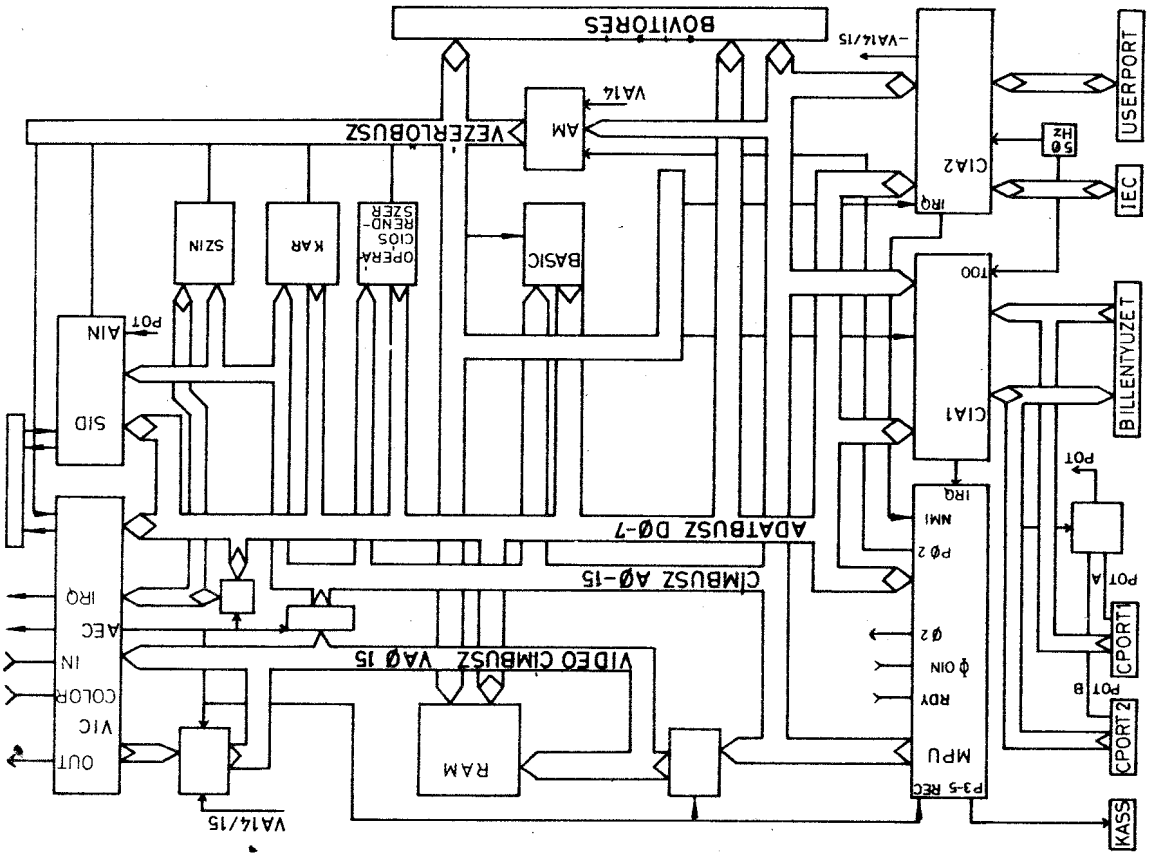
A következő oldalakon megtaláljuk a CBM 64-es kapcsolási rajzát.
Az egyes részeket külön tárgyaljuk a 3. fejezetben és az 1.4 alfejezetben.
Az Olvasó először biztosan érthetetlennek tartja, hogy egy olyan számítógép, amelyben a teljes címtartományt a RAM foglalja el, egyáltalán működhet.
A következő fejezetek szükségszerűen műszaki vonatkozásokat tartalmaznak, ezért ajánlatos előtte az idevágó irodalomban egy kicsit tájékozódni.
Az itt elemzésre kerülő ismeretek elmélyítésére különösen alkalmasak a MOS kézikönyvek. A Hardver kézikönyv ill. a Programozói kézikönyv kifejezetten a 65xx processzorok IC-ivel foglalkozik. (Ábrát lásd a következő oldalon.)

1.2 A hardverfelépítés

A beépített 6510-es processzor csak 64 kbyte-os címtartománnyal rendelkezik, így a gép tervezőinek ahhoz, hogy a felsorolt lehetőségek mindegyikét megvalósíthassák, egészen új, sajátos megoldást kellett keresniük.
Az ötlet kulcsszava: *multiplex*.
Multiplex alatt a továbbiakban azt értjük, hogy a különböző rendszerelemek időben megosztva használják ugyanazt a vezetéket.
Egyszerűen abból indulhatunk ki, hogy a hardver minden egységét egyidejűleg sosem használjuk. Ebben eddig semmi különlegesség nincs, a dolog akkor kezd érdekessé válni, amikor



COMMODORE - 64
BN/E



az egyes egységeket úgy akarjuk egyetlen logikai blokkba összeépíteni, hogy mindegyik elérhető legyen tényleges fizikai átkapcsolás nélkül.
A megoldás nyitja, a berendezés lelke, egy speciális IC, amelyet a továbbiakban AM-mel (Address Manager) jelölünk.
Erről az 1.4 fejezetben még sok szó esik.
Egy példa a különböző, de azonos címtartományban elhelyezkedő egységek átfedésére:
A karaktergenerátor és az I/O tartomány azonos címeken található. Ez azért nem vezet konfliktushoz, mert a VIC (videovezérlő, Video Interface Chip) a karaktergenerátort az órajel alacsonyszintű állapotában használja, amikor a processzor egyébként sem fordul a címbuszhoz.
Az ilyen átkapcsolási megoldás mellett természetesen még előfordulhatnának zavarok, ezeket egy beépített szoftverrel küszöbölik ki.
Ezért azoknak, akik gépi kódban programoznak, célszerű ezt a fejezetet nagyon figyelmesen elolvasni, ebben a részben ugyanis minden kényes pontra kitérünk.
A teljes ROM és a RAM bizonyos részei között 20 kbyte-os területen van átfedés. Ennek megoldásáról az 1.4 fejezetben olvashatunk.
Egy további átlapolás nem a címbusszal, hanem a külső egységekkel (perifériákkal) kapcsolatos.
A CIA egy I/O vonala szintén kétszeresen foglalt; ezen a billentyűzet és a vezérlő port osztoznak. A megvalósítást a 4.7 fejezetben részletezzük.
Magától értetődik, hogy egy ilyen bonyolult rendszernél az egyes egységeknek jóval többet kell „magukra vállalniuk", mint ami a saját feladatuk.
Nézzünk erre egy példát a 6569-es videovezérlővel (VIC) kapcsolatban:
A RAM 4164-es IC-kből épül fel. A hardver egyáltalán nem tartalmaz olyan egységet, ami – a statikus RAM-nál jóval bonyolultabb – címzést kezeli, vagy amely a tárterület rendszeres felfrissítéséről gondoskodna. Ezt mind elrendezi a VIC a saját feladata – a képernyőkép előállítása – mellett.

1.3 A 6510-es processzor és sajátosságai

A CBM egy MPU (Micro Processing Unit) 6510-es processzort tartalmaz.
Ez egy 8 bites processzor, amely a CBM 64-esben 980 kHz-es frekvenciával üzemel.
Az MPU 6510-es a 65xx processzorcsalád új tagja. A korábbi 6502-es processzorhoz képest a 6510-es által hozott legnagyobb változás az, hogy a 6510-es hat I/O kivezetéssel rendelkezik.
A programozható I/O pontok beépítése azzal az óriási előnnyel jár, hogy külön perifériális IC-k használata nélkül is kapcsolatot teremthetünk a külvilággal, vagy belső vezérlési feladatokat láthatunk el.
A CBM 64-esben az I/O pontok egy része a kazettás egységet vezérli, más része pedig a tárkezelést támogatja.
A programozóknak fontos tudniuk, hogy az I/O vonalak a 0-ás és az 1-es tárcímen keresztül érhetőek el, nevezetesen az 1-es címen az adatregiszter, a 0-ás címen pedig az adatirány-regiszter található.
A verem a $0100-$01FF-ig terjedő tárterületen mozoghat. A verem egy olyan tároló, ahol pl. egy alprogramra ugráskor a regiszterek tartalma átmenetileg megőrződik, így a főprogramba való visszatéréskor eredeti tartalmukat helyreállíthatjuk.

A lábkiosztás:

1 OIN órajel bemenet; a CBM 64-esben kb. 980 kHz
2 RDY ready; = 0, a processzor adatot vár az adatbuszról
3 – IRQ megszakítás kérés (Interrupt Request); ha 0, a processzor betölti a $FFFE címről a következő utasítás címét és innen folytatja a végrehajtást. Ez csak akkor következhet be, ha a megszakítás engedélyezett (a kapcsolóregiszterben a 2. bit értéke 0).

4	— NMI nem maszkolható megszakítás (Non-Maskable Interrupt); ha értéke 0, a processzor betölti a $FFFA címről a következő utasítás címét és onnan folytatja a végrehajtást.
5	AEC címzés vezérlés (Address Enable Control); ha értéke 0, a processzor a cím- és vezérlőbuszt nagy impedanciás állapotba hozza. Ekkor a buszt használhatják külső egységek, pl. egy második processzor.
6	VVC üzemfeszültség + 5V
7–20	A0-A13; címbusz
21	GND
22–23	A14-A15; címbusz
24–29	P5-P0; I/O kivezetések
30–37	D7-D0; adatbusz
38	R/-W; 0 = íráshozzáférés, 1 = olvasáshozzáférés. A hozzáférés csak 02 = 1 mellett!
39	02OUT; órajel kimenet a többi egység számára
40	RES Reset; ha értéke 0, a processzor nyugalmi állapotba kerül. Ha értéke 1-re változik, a processzor elindítja a $FFFC címen kezdődő programot.

1.4 A tárfelosztások

A Commodore 64-es számítógép teljes tárterülete a következő részekből áll:

64 kbyte RAM,
20 kbyte ROM,
4 kbyte I/O RAM,
8 kbyte karaktergenerátor ROM.

A 6510-es processzor egy speciális FPLA (Field Programable Logic Array) típusú IC-vel — a már említett AM — oldja meg a tárkezelést.
A tár 0-ás és 1-es címének programozásával elérhetjük, hogy az AM a RAM, a ROM, illetve az I/O területet tetszőleges kombinációban tegye elérhetővé a Commodore 64-es számára.
A CBM 64-esbe beépített IC-t 16 bemenettel és 8 kimenettel látták el. Programozással minden bemenethez (65536 különböző kombináció) tetszőleges kimenetet (256 eset) rendelhetünk.
Mivel az AM a gép egyik legfontosabb része, érdemes megismerkedni az egyes ki- és bemeneti jelek jelentésével.

A bemeneti jelek:

— CAS	A VIC-től érkező jel a RAM címek vezérlésére
— LORAM	a processzor port 0. bitjéről érkező jel
— HIRAM	Ua. mint előbb az 1. bitre vonatkozóan
— CHAREN	Ua. mint előbb a 2. bitre vonatkozóan. Lehetővé teszi az olvasást a karaktergenerátorból. BASIC programból nem tudjuk elérni a karektergenerátort, mert a karaktergenerátor és a timer a CIA chip azonos címén található. Ha mégis megkíséreljük, a megfelelő megszakítást ezen a címen a timer helyett a karaktergenerátort találja, ami hibás elágazáshoz vezet és így az operációs rendszer nem tudja tovább folytatni a munkát.
— VA14	A CIA 2 A portjának 0. bitjéről érkező jel. Előállítja a karaktergenerátor és a videoram báziscímeinek egy részét.
A13-A15	Címbusz jelek: az I/O tartomány dekódolására szolgálnak.
BA	A VIC-ről érkező jel: a processzor RDY vonalát befolyásolja.
— AEC	Ha értéke 0, a buszt a processzor, egyébként a VIC vezérli. A jel a VIC AEC jelének inverze.

R/-W	A processzor vezérlőjele
— EXROM	A bővítő port bemeneti jele
— GAME	mint fent
VA12–13	A VIC címbusz

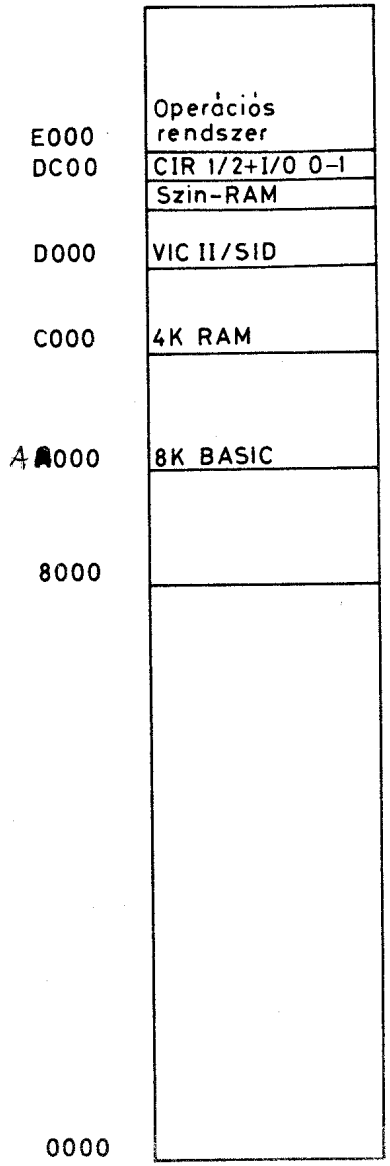
A kimeneti jelek:

— CASRAM	0 = a RAM átveszi a tárcím felső byte-ját
— BASIC	0 = a BASIC ROM kiválasztása
— KERNAL	0 = az operációs rendszer kiválasztása (ROM)
— CHAROM	0 = a karaktergenerátor kiválasztása
GR/-W	0 = a szín-RAM írásának engedélyezése
— I/O	0 = I/O dekódoló kiválasztása
— ROML	Jel a bővítő portra
— ROMH	ua. mint előbb.

A továbbiakban közreadjuk a bemeneti kombinációk és a kimeneti kombinációk egymáshoz rendelésének csaknem teljes listáját.
A „csaknem teljes" a következőkre utal:
A szerzők számára az lett volna a legegyszerűbb, ha megadják a 65536 bemeneti kombinációhoz tartozó lehetséges kimeneteket. A szóbanforgó IC egyik tulajdonsága azonban, hogy vannak redundáns bitjei, ami azt jelenti, hogy ezek semmilyen hatást nem gyakorolnak a kimenetre. Az ilyen biteket a táblázatban x-szel jelöltük. Az ilyen bitek kiszűrésére írtunk egy programot, amely négy heti folyamatos működéssel sem tudta befejezni a keresést.
Emiatt a táblázat végéről néhány kombináció hiányzik, pl. a $FE és a $FF értékekhez tartozó kombinációk. Ezeknek azonban egyébként sincs nagy jelentősége, ugyanis ahogyan azt majd látni fogják, ahhoz, hogy egy kimenet befolyásoljon valamit, alacsony szintűnek kell lennie.
Ha az Olvasó nem szándékozik „végigküzdeni" magát a táblázaton, tekintse meg a következő oldalakon található ábrákat, amelyek bemutatják az AM hatását a tárfelosztásra.
(Táblát és ábrákat lásd a következő oldalakon.)

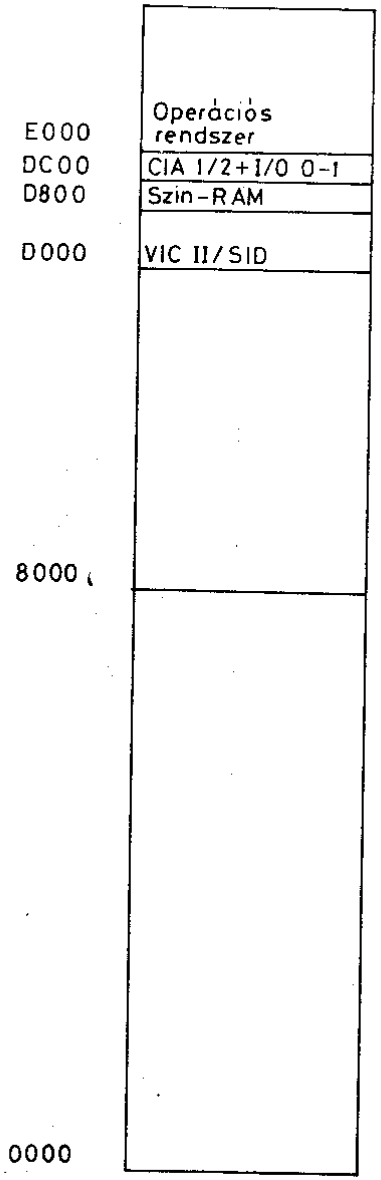
*** PAL CBM 64 ***

VA13	VA12	GAME	EXROM	R/W	AEC	BA	A12	A13	A14	A15	VA14	CHAREN	HIRAM	LORAM	CAS	ROMH	ROML	I/O	GR/W	CHAROM	KERNAL	BASIC	CASRAM
0	0	X	X	1	0	X	X	1	0	1	X	X	1	X	X	0	1	1	1	1	1	1	1
1	0	X	X	X	0	X	X	1	1	1	X	X	X	X	X								
1	0	1	1	X	1	X	X	X	X	X	X	X	X	X	X								
X	0	X	X	1	0	X	X	0	0	1	X	X	1	1	X	1	0	1	1	1	1	1	1
0	X	X	X	1	0	X	X	0	0	1	X	X	1	1	X								
1	0	X	X	X	0	X	X	0	0	1	X	X	X	X	X								
X	X	X	X	0	0	X	1	0	1	1	X	1	X	1	0	1	1	0	0	1	1	1	1
X	X	X	X	0	0	X	1	0	1	1	X	1	1	X	0								
1	0	X	X	0	0	X	1	0	1	1	X	X	X	X	0								
X	X	X	X	0	0	X	1	0	1	1	X	1	X	1	1	1	1	0	1	1	1	1	1
X	X	X	X	1	0	1	1	0	1	1	X	1	X	1	X								
X	X	X	X	1	0	1	1	0	1	1	X	1	X	1	X								
X	X	X	X	1	0	1	1	0	1	1	X	1	1	X	X								
1	0	X	X	0	0	X	1	0	1	1	X	X	X	X	1								
1	0	X	X	1	0	1	1	0	1	1	X	X	X	X	X								
0	X	X	X	0	0	X	1	0	1	1	X	0	X	X	0	1	1	1	0	1	1	1	0
0	X	X	X	0	0	X	1	0	1	1	X	X	0	0	0								
X	1	X	X	0	0	X	1	0	1	1	X	X	0	0	0								
X	1	X	X	0	0	X	1	0	1	1	X	X	0	0	0								
X	1	X	X	1	0	X	1	0	1	1	X	0	1	X	X	1	1	1	1	0	1	1	1
X	1	0	1	X	1	X	X	X	X	X	1	X	X	X	X								
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	1	1	1	1	0	1	1
X	1	X	X	1	0	X	X	1	1	1	X	X	1	X	X								
X	1	X	X	1	0	X	X	1	0	1	X	X	1	1	X	1	1	1	1	1	1	0	1
0	X	X	X	X	X	X	X	X	X	0	0	X	X	X	0	1	1	1	1	1	1	1	0
0	X	X	X	X	0	X	X	X	X	0	X	X	X	X	0								
0	X	X	X	X	X	X	X	X	X	0	X	0	X	0	X	0							
0	X	X	X	0	X	X	0	X	X	X	0	X	X	X	0								
0	X	X	X	X	0	X	0	X	X	X	X	X	0	X	0								
0	X	X	X	0	0	X	0	X	X	X	X	X	X	X	0								
0	X	X	X	X	X	X	X	1	X	X	0	X	X	X	0								
0	X	X	X	X	0	X	X	1	X	X	X	X	0	X	0								
0	X	X	X	0	0	X	X	1	X	X	X	X	X	X	0								
0	X	X	0	X	X	X	X	X	X	0	X	X	X	X	0								
0	X	X	0	X	1	X	X	X	X	X	X	X	X	X	0								
0	X	X	X	X	X	X	X	0	0	X	0	X	X	0	0								
0	X	X	X	X	0	X	X	0	0	X	X	X	X	0	0								
0	X	X	X	X	X	X	0	0	1	X	0	X	X	X	0								
0	X	X	X	X	0	X	0	0	X	X	X	X	X	0	0								
0	X	X	X	X	0	X	0	0	1	X	X	X	X	X	0								
0	X	X	X	1	X	X	X	X	X	X	0	X	0	0	0								



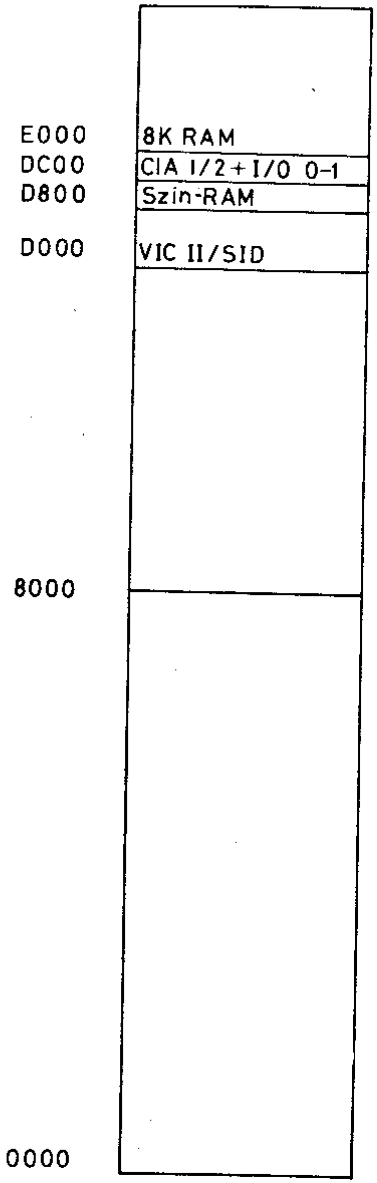
A bekapcsolás utáni tárfelosztás, bővítés nélkül.

- LORAM =1
- HIRAM =1
- EXROM =1
- GAME =1
- CHAREN=1



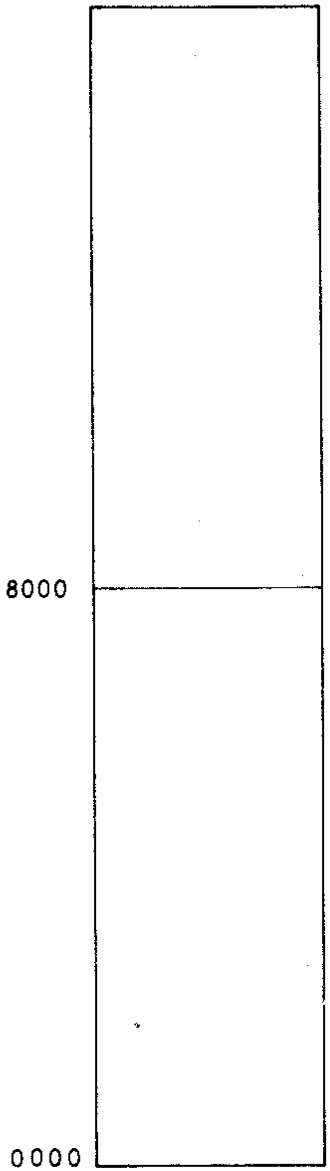
Ez a tárfelosztás pl.akkor képzelhető el,amikor egy módosított BASIC rendszert másoltunk az áthelyezhető RAM-ba

- LORAM =0
- HIRAM =1
- EXROM =X
- GAME =1
- CHAREN=1



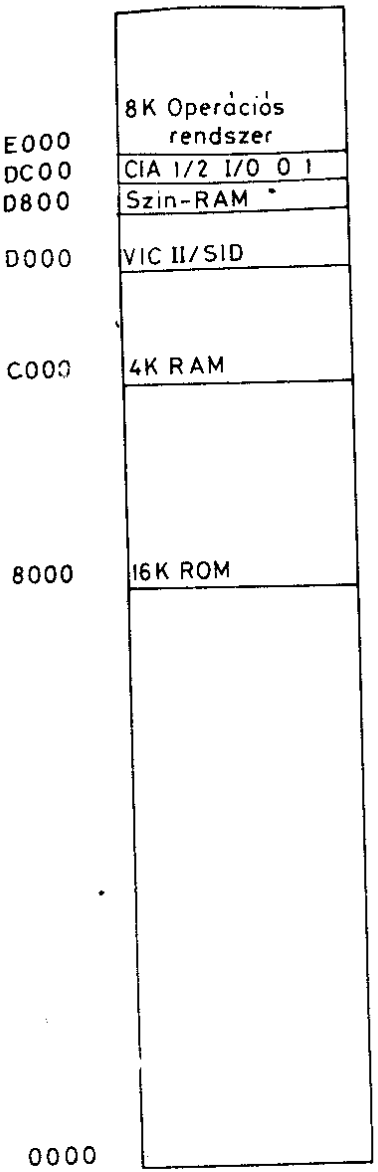
Egy külső processzor működik a tárban, miközben az I/O egységeket is használjuk.

- LORAM =1
- HIRAM =0
- EXROM =X
- GAME =1
- CHAROM=1



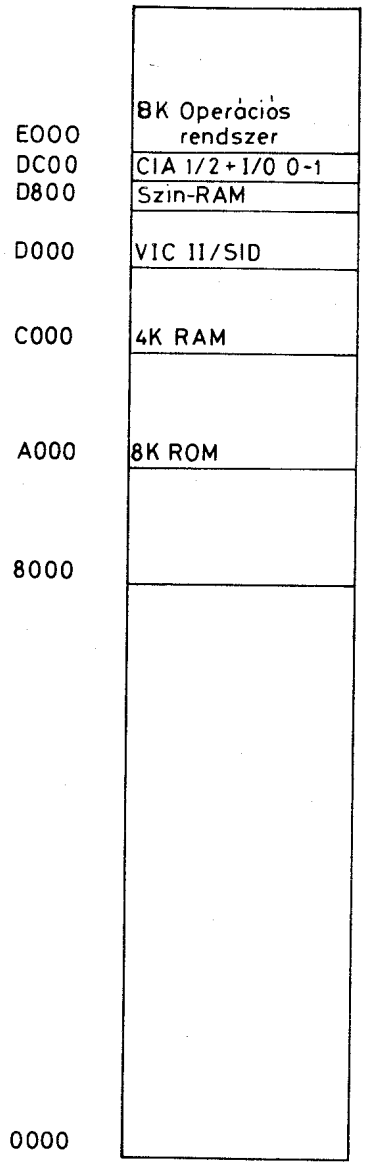
Azonos eset az előzővel,
I/O elemek nélkül.

- LORAM = 0
- HIRAM = 0
- EXROM = 1
- CAME = 1



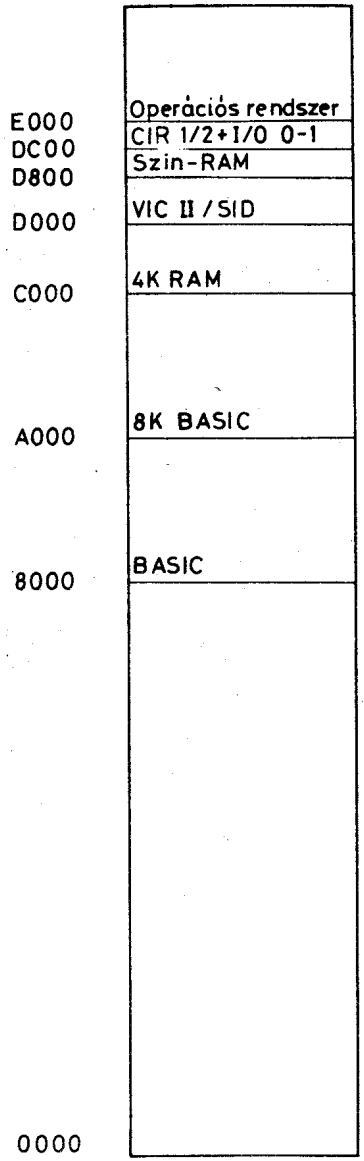
Tipikus eset, amikor nem BASIC,
hanem valami más programnyel-
vet használunk.
A másik nyelv (pl. a PASCAL) a
külső ROM területen helyezke-
dik el

- LORAM = 1
- HIRAM = 1
- EXROM = 0
- GAME = 0
- CHAREN = 1



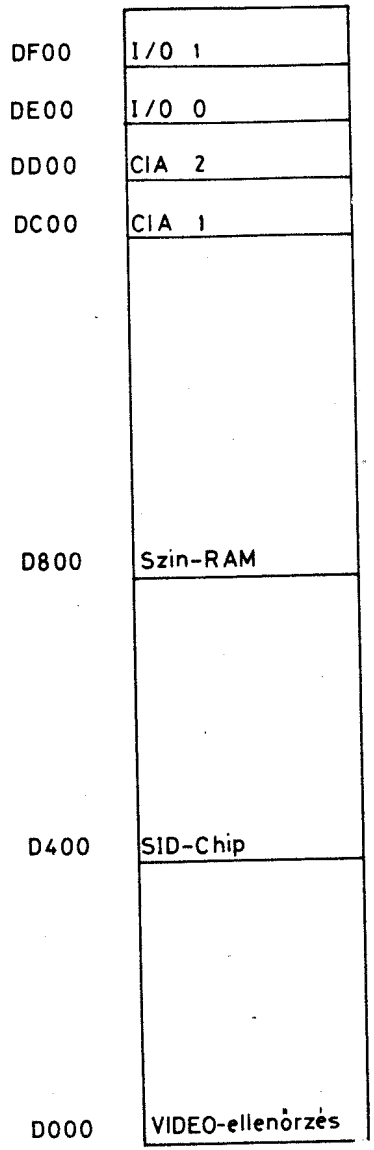
A játékokra alkalmas tárkiosztás

- LORAM = 0
- HIRAM = 1
- EXROM = 0
- GAME = 0
- CHAREN = 1



Egy ROM egységgel bővített BASIC. (pl. a HELP+)

- LORAM = 1
- HIRAM = 1
- EXROM = 0
- GAME = 1
- CHAREM = 1



Ezen a címtárfelosztáson látható az I/O terület, ha a CHAREN=1. A CHAREN=0-nál a teljes területet a karaktergenerátor foglalja el.

1.5 A bővítő port

A CBM 64-es hátoldalán található egy 44 lábú csatlakozó, amit bővítő (expansion) portnak nevezünk.
Ezen a bemeneten elérhetjük a teljes rendszerbuszt és a rendszerbuszhoz csatlakozó vezérlő vonalakat.
Bővíthetjük az operációs rendszert, a BASIC-et, csatlakoztathatunk a géphez játékokat, I/O egységeket (pl. IEC buszt). Külső proceszorral elérhetjük a rendszer belső egységeit stb.
Megfelelő berendezésekkel egyszerre több bővítő egység használatára is módunk van.
Azonban a VC-20-as számára készített modulok sem logikailag, sem fizikailag nem illeszthetőek a bővítő portra. Ha ezt valaki mégis megkísérli, elő kell állítania a VC-20-asnál jól ismert blokk-kiválasztó jelet, a 13-as, 14-es, 15-ös címbitek dekódolásával.

Az alábbiakban ismertetjük a lábkiosztást:

1	GND
2–3	+5 V
4	– IRQ; a processzor IRQ-val összekötve
5	CR/-W, a processzor R/-W-vel összekötve
6	DOT CLOCK; a VIC pontraszter-üteme, kb. 7,83 MHz
7	– I/01; általában = 0 a $DE00-tól $DEFF-ig terjedő tárterületen
8	– GAME; bemenet az AM-hez, ld. az 1.4 alfejezetben
9	– EXROM; mint fent
10	– I/02; általában = 0 a $DF00-tól a $DFFF-ig terjedő tárterületen
11	– ROML; kimenet az AM-ről, ld. az 1.4 alfejezetben
12	BA; a VIC-ről érkező jel, az olvasott adatok érvényességére utal
13	– DMA; bemenet. Ha 0, a rendszerbusz külső hozzáférésre van fenntartva
14–21	CD7-CD0; adatbusz
22	GND
A	GND
B	– ROMH; kimenet az AM-ről, ld. az 1.4 alfejezetben
C	– RESET
D	– NMI
E	02; órajel kimenet
F–Y	CA15-CA0; címbusz
Z	GND

Jelentősebb eltérések a VC-20-as bővítő portjához viszonyítva: Az érintkezők távolsága itt 2,54 mm, a VC-20-asnál 3,96 mm. A blokk- és tárkiválasztó jelek, – RAM1–3 és – BLK1–5 teljesen hiányoznak, mivel itt a teljes címbuszt kivezették (a VC-20-asnál csak a 0-tól 12-ig terjedő címbitek vannak kivezetve).
Eltérő az I/O tárterület mérete; itt 256 byte, a VC-20-asnál 1 kbyte.

Ha az Olvasó figyelmesen áttanulmányozta ezt és az előző alfejezetet, feltehetően világosan látja, hogy saját maga is befolyásolhatja a számítógép tárkiosztását. Különösen az – EXROM és a – GAME jelek megválasztásával, ekkor ugyanis úgy megváltozhat a tár felépítése, hogy bekapcsolás után még a megszokott READY üzenet sem jelenik meg a képernyőn. Célszerű tehát ezeket a jeleket kellő óvatossággal kezelni.

1.6 A user port

A user port a CBM 64-es sokoldalú interface-e, amely, mint ahogyan azt a neve is tükrözi, elsősorban a használók igénye szerinti külső egységek csatlakoztatására szolgál.
Ez a külső egység lehet egy egyszerű LED-től kezdve egy nyolcbites párhuzamos interface-szel ellátott Centronics nyomtatóig bármi, amit szívesen csatlakoztatnánk a CBM 64-eshez.
Az illesztéshez egy kétsoros, azaz 2 X 12 egymástól 3,96 mm-re levő érintkezővel ellátott csatlakozót használhatunk.
Az user port lényegében egy nyolcbites portból és a következőkben felsorolt vezérlővonalakból épül fel:

1	GND
2	+ 5 V; max. 100 mA-rel terhelhető
3	— RESET; a hasonnevű processzorvonallal összekötve
4	CNT1; a CIA 1 CNT vonalával összekötve
5	SP1; a CIA 1 SP vonalával összekötve
6	CNT2; a CIA 2 CNT vonala
7	SP2; a CIA 2 SP vonalával összekötve
8	— PC2; a CIA 2 handshake kimenete
9	ATN OUT; a soros busz vezérlő csatornája
10	9 V AC; váltófeszültség, max. 100 mA-rel terhelhető
11	9 V AC; a fenti, ellenfázisban
12	GND
A	GND
B	— FLAG2; a CIA 2 handshake bemenete
C—L	PB0-PB7; a CIA 2 I/O vonalai
M	PA2; a CIA 2 I/O vonala. Más CBM gépeken a VIA 6522-es CB2 vonalaként ismert
N	GND

A fenti vonalak közül néhánynak rögzített feladata van a CBM 64-esen. Ezeket és a CIA 6526-os részletes leírását megtalálják Olvasóink a 4. fejezetben.
Ha a user portra „olyan külső egységeket akarunk csatlakoztatni, amelyeket eredetileg a VC-20-ra vagy pl. a CBM 8032-re terveztek, ez csak abban az esetben lehetséges, ha a külső egység működéséhez elegendőek az A-N, az 1-es és a 12-es lábak.
Ugyanakkor nem szabad megfeledkezni a B és az M lábak programozástechnikailag eltérő voltáról. Idevonatkozó részleteket ugyancsak a 4. fejezetben közlünk.

1.6.1 A USER PORT PROGRAMOZÁSA BASIC-BEN

Ezt a fejezetet főként azoknak az Olvasóinknak szántuk, akik a BASIC nyelvet nagyon mélyrehatóan ismerik, de nincsenek tisztában az olyan külső egységek felhasználási lehetőségeivel, amelyeket nem lehet egyszerű fizikai csatlakoztatással az alapgéphez illeszteni. Bemutatjuk, hogy miként lehet egy egyszerű felépítésű áramkört a user porthoz csatlakoztatni és BASIC programból kezelni.
Tegyük fel, hogy az áramkör mindössze négy kapcsolóból, négy fénykibocsátó diódából, nyolc ellenállásból és egy IC-ből áll.
A user porton keresztüli adatbevitel és adatkihozatal alapelveit az Olvasó itt meg fogja érteni és a tapasztalatai alapján a saját elképzeléseit is meg tudja valósítani. A kapcsolási rajzot, amely annyira egyszerű, hogy nem szorul különösebb magyarázatra, a fejezet végén közöljük.
A CBM 64-es user portján sokkal több foglalt kimenet van, mint a többi CMB gépnél, ezért célszerű utalnunk arra, hogy a kimenetek közül melyek a tényleges „user" kimenetek.

Ha pl. nem dolgozunk RS-232-es illesztővel, akkor szabadon (anélkül, hogy bármit is elrontanánk) használhatjuk pl. a következő lábakat: 1—2, 4—8, 10—12, A—N (ezeknek a jelentését az előző fejezetben megadtuk).
Térjünk vissza a kiindulásunkhoz:
A PB0-PB7 adatvonalakat szabadon programozhatjuk adatbevitelre és adatkihozatalra.
Használjuk például a PB0-tól PB3-ig terjedő vonalakat bevitelre, a PB4-től PB7-ig terjedő vonalakat pedig kihozatalra. Az adatáramlás irányát az 56579-es tárcímen levő B adatportra vonatkozó adatirányregiszter tartalmának beállításával szabályozhatjuk.
Ha egy bit értéke 1, a B adatport (56577-es cím) megfelelő bitjének outputjára, ha 0, az inputjára utal. Esetünkben tehát a 0-tól 3-ig terjedő biteket 0-ra (IN), a 4-től 7-ig terjedő biteket pedig 1-re (OUT) kell állítani az adatirány-regiszterben. A megfelelő BASIC utasítás:

```
POKE 56579,240
```

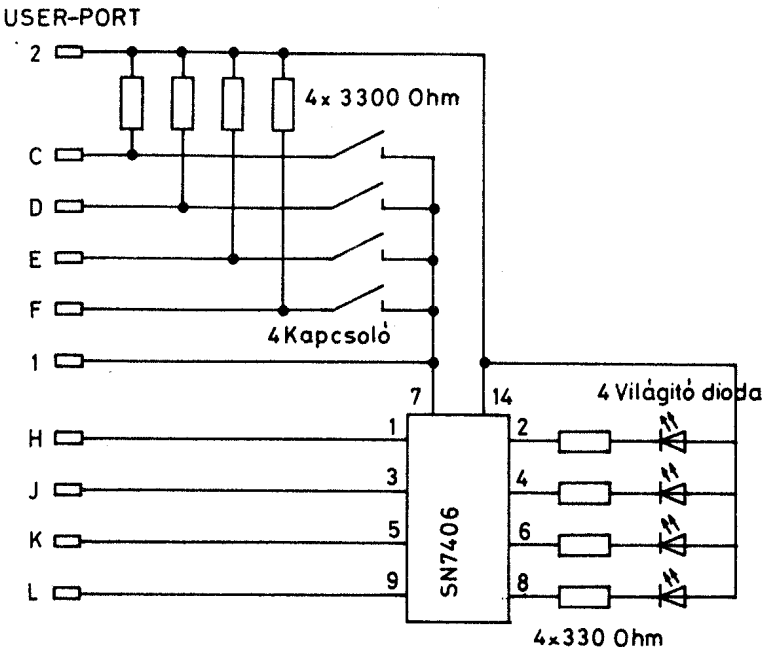
Hogyan programozzuk a kapcsolásunkat? Pl. a következő igen egyszerű utasítással

```
PRINT PEEK (5657) AND 15
```

leolvashatjuk a négy kapcsoló állását, ill. a

```
POKE 56577,X
```

utasítással tetszőlegesen beállíthatjuk a kimenetet, azaz a fénykibocsátó diódákat ki- és bekapcsolhatjuk, X értéke 16, 32, 64, 128 vagy 0 lehet (pl. 0, ha minden lámpát ki akarunk kapcsolni).
Ha az Olvasó saját tervezésű egységeket akar a user portra csatlakoztatni, kérjük vegye figyelembe az alábbiakat, ha el akarja kerülni azt, hogy esetleg a számítógép megsérüljön.
A bemeneti feszültségnek 0 . . . 5 V között kell lennie. A 0 . . . 0,6 V közé eső feszültséget a rendszer az adatportról való olvasásnál 0-nak, az 1,6 . . . 5 V közötti értékeket pedig 1-nek értelmezi. A 0,7 . . . 1,5 V-ig terjedő tartomány határozatlan, ezért véletlenszerűen 1 vagy 0 értéket képviselhet.
Ha kimenetként akarjuk használni a user portot, vegyük figyelembe, hogy a kimeneteket csak egy TTL bemenetszintű terhelésre méretezték. Semmiképpen ne csatlakoztassunk tehát közvetlenül pl. egy fénykibocsátó diódát, ezzel ugyanis előbb-utóbb a CIA-t biztosan tönkretennénk. Célszerű mindig egy vonalmeghajtó fokozatot beépíteni, mint ezt az ismertetett példában láttuk.
Egy kimenetre programozott portbitre sohase kapcsoljunk idegen feszültséget, ez ugyanis egészen biztosan sérüléshez vezetne. Alaposan fontoljuk meg, hogy hogyan állítjuk be az adatirányregiszter értékét, nehogy egy beolvasásra kijelölt bitet véletlenül kimenetként programozzunk.
Ha egy készülék áramellátását a user portról akarjuk megoldani, ne feledkezzünk meg arról, hogy a két rendelkezésre álló feszültség egyike sem terhelhető 100 mA-nél nagyobb árammal.
Ha ezt az értéket túllépjük, először leáll a kazettás egység, majd kiég a C 64-es belső biztosítékja, végül esetleg a transzformátor primer biztosítékja. További sérülésekkel nem kell számolnunk.
Természetesen a fenti példával csak egy kis betekintést nyújtottunk a user port egyszerű felhasználási lehetőségeibe.
Ha bonyolultabb feladatokat szeretne megoldani a többi vonal felhasználásával, úgy támaszkodjon a 4. fejezet részletes leírására.
A fentiek vonatkoznak azokra a vonalakra is, amelyekre most nem tértünk ki, és természetesen mindegyikük PEEK-POKE utasítással programozható.



2. FEJEZET
A SZINTETIZÁTOR ÉS PROGRAMOZÁSA

2.1 A 6581-es hangvezérlő

2.1.1 A 6581-ES ÁLTALÁNOS ISMERTETÉSE

A CBM 64-esbe beépített szintetizátorral a fuvolahangtól a mozdony pöfögésig minden elképzelhető hangot megszólaltathatunk. Míg a kereskedelmi forgalomban kapható szintetizátorok általában csak egy hangon szólnak, addig a CBM 64-es szintetizátora három hangot tud egyszerre kezelni.
A hangzáshoz szükséges elemek egyetlen IC-be, nevezetesen a SID (Sound Interface Device) 6581-esbe vannak beépítve, amely a 65xx-es család egyik új tagja.

A SID felépítése a következő:

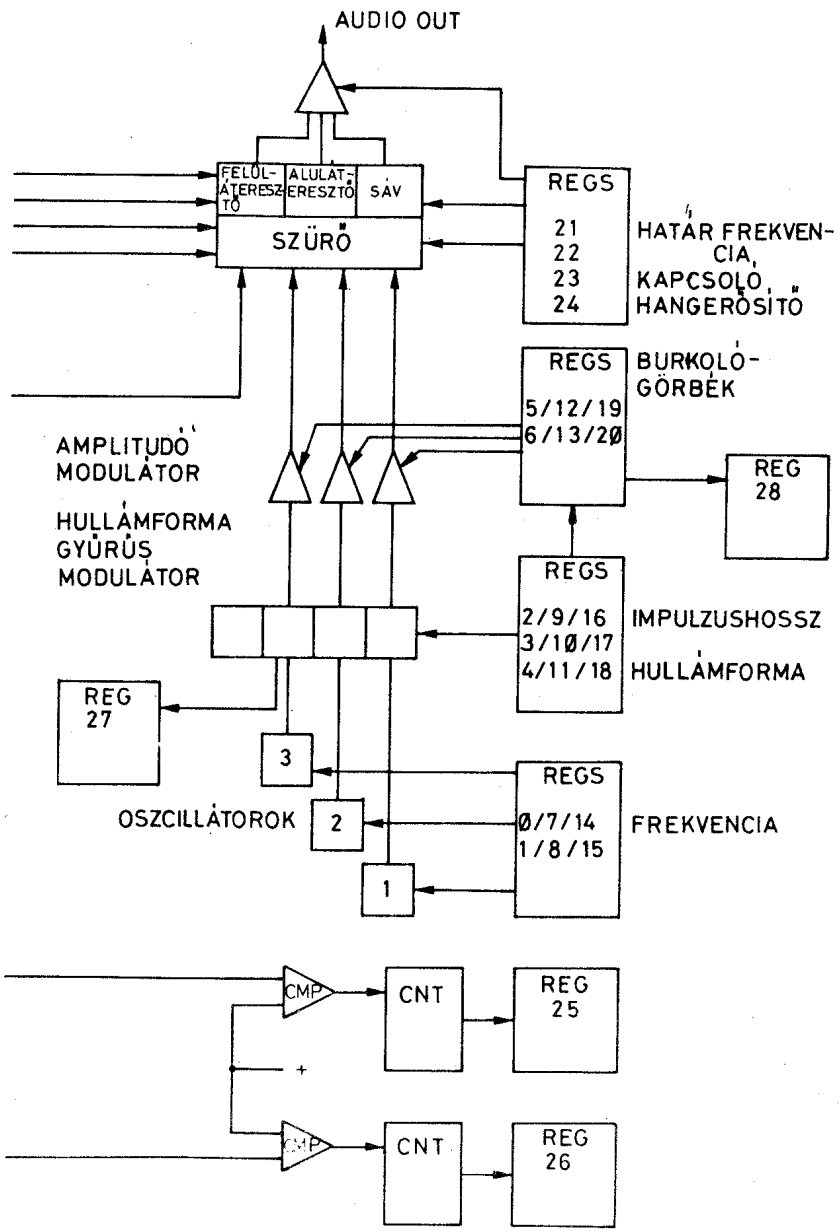
- három külön-külön programozható oszcillátor (hang)
- hangonként négy keverhető rezgésfajta
- hangonként három keverhető szűrő (felüláteresztő, aluláteresztő, ill. sávszűrő)
- hangonként egy-egy burkológenerátor [ADSR (Attack/Decay/Sustain/Release) vezérlés]
- két kaszkadolható gyűrűsmodulátor
- külső jelforások leválasztási lehetősége
- két nyolcbites A/D átalakító

A lábak kiosztása:

1–2	CAP1A, CAP1B; a programozható szűrők kondenzátorainak csatlakozása. Javasolt kapacitás: 2200 pF
3–4	CAP2A, CAP2B; ua. mint az 1–2
5	– RES (Reset); = 0 alapállapotba hozza a SID-et
6	02 (órajel); az adatbusz műveletek csak a 02 = 1 mellett megengedettek
7	R/W (Read/-Write); 0 = hozzáférés írásra, 1 = hozzáférés olvasásra
8	– CS (Chip Select); 0 = az adatbusz érvényes, 1 = az adatbusz nagy impedanciájú (Tri-state)
9–13	A0-A4 (0–4 adatbitek); a SID egyik regiszterének kiválasztása a 29-ből.
14	GND (föld); figyelem: ahhoz, hogy elkerüljük a rendszerelemek nem kívánatos kölcsönhatását, a SID áramellátásához egy saját földre van szükség
15–22	D0-D7; adatvonalak a processzortól/-hoz
23	A2IN (analog Input 2); ld. részletesen a 4.1.3. alfejezetben!
24	A1IN; ua. mint a fenti, csak az 1-es A/D átalakítóhoz
25	VCC; + 5 V tápfeszültség
26	EXT IN (External Input); egy külső audiojel bemenete, a jelet a SID leválasztja
27	AUDIO OUT; a SID által előállított vagy kevert jelek együttes kimenete enete
28	VDD; + 12 V tápfeszültség

A SID 6581-esbe beépített három hangszintetizátort megszólaltathatjuk egyszerre, külön-külön, illetve külső hangforrásokkal együtt, és így egészen bonyolult hullámformákat is előállíthatunk. Minden egyes hang egy oszcillátorból, egy hullámforma-generátorból (négy különböző hullámformával), egy burkológenerátorból és egy amplitudómodulátorból áll.
Az oszcillátor egy 16 biten ábrázolható alapfrekvenciát ad, amely a 0-tól 8200 Hz-ig terjedő tartományba esik (1 MHz-es órajel mellett).

A SID kapcsolási rajza:



A lehetséges hullámformák: háromszög, fűrészfog, négyszög (változtatható szünetviszonyokkal) és zaj.
A hullámforma megválasztása is alapvetően meghatározza a hangzást. A háromszög hullámforma a szinuszgörbéhez közelítő alakjából adódóan lágy, fafuvolára emlékeztető hangot ad.
A fűrészfog hullámforma egyenletes eloszlású teljes spektrumával élesen szólal meg, mint egy trombita.
A négyszög hullámformában hiányoznak a felharmonikusok, így kissé tompán a klarinéthoz hasonlóan hangzik.
Végül a zaj hullámforma a megengedett tartományon belül különböző frekvenciák véletlen egymásutánjával éri el a zavaros, zajos hanghatást.
A hangerőt az amplitudómodulátor szabályozza, amit a burkológenerátorral vezérelhetünk.
Ha a burkológenerátor egy ún. trigger-bitet (szerepe hasonló a zongora billentyűjének szerepéhez) érzékel, az előzetesen programozott hang megszólal, a jól ismert három fázisban: felerősödés, kitartás, lecsengés.
A hangszínt tovább szabályozhatjuk a szűrők programozásával. Végül az 1-es és 2-es hangot a 3-as hanggal modulálhatjuk, ami azt eredményezi, hogy az alaphang mellett erősebben észlelhető az együtt- ill. különhangzás.
Ha egy hang megszólalása közben hangszínt akarunk változtatni (mint pl. a WAH-WAH effektusnál), a 3-as hanggal azt is megtehetjük, hogy leolvassuk a burkológenerátor pillanatnyi állását és ettől függően megváltoztatjuk a szűrőfrekvenciát, esetleg az oszcillátorfrekvenciát.
A 3-as hang zajgenerátorát is bármikor lekérdezhetjük és a kapott értéket nagyszerűen felhasználhatjuk véletlen számként. A SID tartalmaz két nyolcbites A/D (analóg/digital) átalakítót is. Ezekkel lekérdezhetjük a vezérlő portra csatlakoztatott két potenciométer mindenkori értékét. Ezt a lehetőséget leggyakrabban játékprogramoknál használják, bár alkalmas hangerő, hangmagasság szabályozására, vagy pl. egy hidegvezeték csatlakoztatásával hőmérséklet mérésére is.

2.1.2 A SID REGISZTEREI

A SID a $D400 (54272)-es tárcímen kezdődik.
A vezérlő-regiszterek szerepe a következő:

REG 0 az 1-es hang oszcillátorfrekvenciája (alsó byte)
REG 1 az 1-es hang oszcillátorfrekvenciája (felső byte)
REG 2 az 1-es hang impulzusszélessége (alsó byte)
REG 3 az 1-es hang impulzusszélessége (felső byte)
A 2-es és 3-as regiszterek meghatározzák az 1-es hang négyszögkimenetének impulzus/szünet viszonyait.
A 3-as regiszter bitjei közül csak a 0-tól 3-ig terjedő bitek tartalma meghatározó.

REG 4 az 1-es hang vezérlőregisztere

0. bit — KEY: a burkológörbe-generátort vezérli. Ha értéke 0-ról 1-re nő, akkor a hangerő eléri az 5-ös regiszterben programozott ATTACK időtartam alatt a maximumot, majd ugyancsak az 5-ös regiszterben megadott DECAY időtartam alatt lecsökken a 6-os regiszterben megadott SUSTAIN szintre, és mindaddig ezen a szinten marad, míg a vezérlőbit értéke újra 0 nem lesz. Ekkor a 6-os regiszterben rögzített RELEASE időtartam alatt a hangerő leesik nullára.

1. bit — SYNC: 1 = az 1-es oszcillátor háromszög hullámforma kimenetét egy kevert frekvencia adja meg (az 1-es és a 3-as oszcillátor frekvenciáinak összege és különbsége). Hatása akkor is érvényesül, ha a 3-as hang ki van kapcsolva.

2. bit – RING: 1 = az 1-es oszcillátor háromszög hullámforma kimenetét egy kevert frekvencia adja meg (az 1-es és a 3-as oszcillátor frekvenciáinak összege és különbsége).

3. bit – TEST: ha valamelyik oszcillátornál a zajgenerátor mellett hullámformát is választunk, előfordulhat, hogy a zajgenerátor leblokkol. Ezt kiküszöbölhetjük a 3-as bittel.

4. bit – TRI: 1 = háromszög hullámforma kiválasztása

5. bit – SAW: 1 = fűrészfog hullámforma kiválasztása

6. bit – PUL: 1 = négyszög hullámforma kiválasztása

A hullámforma impulzus/szünet viszonyait a 2-es és a 3-as regiszterekben kell beállítani.

7. bit – NSE: 1 = a zajgenerátor kiválasztása
Megjegyzés a 4–7 bitekhez: egyszerre több hullámformát is választhatunk, de az eredmény az egyes hullámformáknak nem összege, hanem logikai ÉS kapcsolata lesz. Ne feledkezzünk meg a 3-as bitnél említett problémáról sem.

REG 5 ATTACK/DECAY
0–3. bitek: meghatározák azt az időtartamot, amely alatt a hangerő a maximumról leesik a SUSTAIN szintre.
Az időtartam értéke a 6 ms-tól 24 s-ig terjedő tartományba eshet.

4–7. bitek: meghatározzák azt az időtartamot, amely alatt (a KEY bit magasra állítása után) a hangerő nulláról a maximális szintre emelkedik.
Megengedett tartomány: 2 ms-től 8 s-ig.

REG 6 SUSTAIN/RELEASE
0–3. bitek: meghatározzák azt az időtartamot, amely alatt (a KEY bit alacsonyra állítása után) a hangerő a SUSTAIN szintről leesik nullára.

4–7. bitek: meghatározzák a SUSTAIN szintet, a hang leesése (DECAY) után kitartott hangerőt.

REG 7–REG 13 A 2-es hang vezérlőregiszterei. Jelentésük a 2-es hangra vonatkozóan az alábiakban felsorolt eltérésektől eltekintve azonos a 0–6. regiszterek jelentésével.
SYNC: a 2-es oszcillátort az 1-es oszcillátorral szinkronizálja.
RING: a 2-es oszcillátor háromszögkimenetét a 2-es és az 1-es oszcillátorok gyűrűmodulált frekvenciájából állítja elő.

REG 14–REG 20 A 3-as hang vezérlőregisztere. Jelentésük a 3-as hangra vonatkozóan az alábbi eltérésektől eltekintve ua. mint a 0–6. regiszterek jelentése.
SYNC: a 3-as oszcillátort a 2-essel szinkronizálja.
RING: a 3-as oszcillátor háromszögkimenetét a 3-as és a 2-es oszcillátorok kevert frekvenciájából állítja elő.

REG 21 A szűrőfrekvencia alsó byte-ja. Csak a 0–2 biteket használjuk.

REG 22 A szűrőfrekvencia felső byte-ja.
A 21-es és a 22-es regiszterek 11 bitje határozza meg a szűrők levágási frekvenciáját.
A frekvencia kiszámítási módja:
F = (30 + W * 5,8) Hz, ahol W a 11 bites számérték.

REG 23 Szűrőrezonancia és -kapcsoló
0. bit: 1 = az 1-es hang szűrése
1. bit: 1 = a 2-es hang szűrése
2. bit: 1 = a 3-as hang szűrése
3. bit: 1 = a külső hangforrás szűrése
4–7. bitek: meghatározzák a szűrő rezonancia frekvenciáját, segítségükkel a frekvenciaspektrum bizonyos részei kiemelhetőek. Különösen érezhető ez a hatás a fűrészfog hullámformánál.

REG 24 A regiszter feladatai:
0–3 bitek: összhangerő
4. bit: aluláteresztő szűrő
5. bit: sávszűrő
6. bit: felüláteresztő szűrő
A felül- és aluláteresztő szűrő meredeksége 12 dB/oktáv, a sávszűrője 6 dB/oktáv.
Egyszerre több szűrőt is bekapcsolhatunk. Ha például egyszerre alkalmazzuk az alul- és felüláteresztő szűrőt, sávzárat kapunk.
Általában a szűrőt arra használjuk, hogy egy frekvenciaspektrumból egy meghatározott tartományt kiemeljünk. Szűréssel sokkal árnyaltabb hangzásokat kaphatunk, mintha pusztán a hullámformát változtatjuk.
Ha egy hang lefutása közben változtatjuk a szűrőfrekvenciát (egy kis gépi kódú programmal), a legkülönfélébb hangszereket is utánozhatjuk.
7. bit: a 3-as hang nem hallható. Ezt a lehetőséget a többi hang paramétereinek meghatározására használhatjuk. (ld. a 27-es és 28-as regisztereket).

A fenti regiszterek a csak írható, az alábbiak pedig a csak olvasható regiszterek:

REG 25 A/D átalakító (1)

REG 26 A/D átalakító (2)

REG 27 Zajgenerátor (3). Ez a regiszter tartalmaz egy, a 3-as zajgenerátor állapotának megfelelő véletlen számot. A generátort be kell kapcsolni, de a 3-as hang kikapcsolt állapotban is lehet (a 24. regiszter 7. bitje magas).

REG 28 3-as burkológenerátor. A regiszterből leolvashatjuk a 3-as hang relatív hangerejét. A relatív hangerő pillanatnyi értékének megfelelően változtathatjuk a frekvenciát, vagy a szűrő paramétereit (ld. 2.2 fejezet).

2.1.3 AZ ANALÓG/DIGITÁLIS ÁTALAKITÓ

Az A/D átalakító az analóg jelet (pl. feszültséget) digitális jellé alakítja. Az eljárás elvi nehézségét az okozza, hogy a végtelen finom fokozatú analóg jelnek (digitális) véges sok fokozatú jelet (pl. egy intervallumot) kell megfeleltetni. Az átalakítás során szükségszerűen számolnunk kell bizonyos mértékű +/– eltérésekkel.
A SID 6581-es két A/D átalakítót tartalmaz (POTX, POTY). Mindkettő fix 5 V referenciafeszültséget használ.
Az olvasási folyamat lényege a következő: egy külső kapacitás kisül, majd a 25-ös, illetve 26-os regiszterekbe egy, a kapacitás újratöltési idejével arányos érték kerül. Ez a folyamat ciklikusan ismétlődik.

2.1.3.1 Az A/D átalakító kezelése

A fentiek alapján világos, hogy az átalakítóhoz csak potenciométeres kapcsolót lehet használni. Az érzékelők csak változtatható ellenállások lehetnek, pl. fotoellenállások, melegvezetők, hidegvezetők stb.
Ha például feszültséget akarunk mérni, a feszültséget először át kell alakítani pl. egy egyenirányító transzformátor segítségével.
A mérőberendezés nagyon egyszerű, a mérőellenállás egyik vége az + 5 V-ra (a Commodore 64-es vezérlő portján), a másik vége pedig a SID chip analóg bemenetére (szintén elérhető a vezérlő porton, POTX és POTY-nal jelölve) van kötve. A 25-ös és a 26-os regiszterekből leolvasható érték megfelel az ellenállás mértékének. Ahhoz, hogy a teljes 8 bitet kihasználhassuk, az ellenállásnak a 200 Ω-tól 200 kΩ-ig terjedő tartományba kell esnie.
Az A/D átalakító programozástechnikai kezelését a következő fejezetben tárgyaljuk.

2.1.3.2 A vezérlőjátékok (paddle-k) használata

A CBM 64-eshez csatlakoztathatjuk a kereskedelmi forgalomban kapható vezérlőjátékokat. Maximum két pár, azaz négy játékot használhatunk egyszerre, a gép jobb oldalán levő 1-es, illetve 2-es portról. A játék nem egyéb, mint egy potenciométer (amit a fent leírt módon kötünk a SID-hez) és egy nyomógomb, amely az egyik játéknál a baloldali, a másiknál a jobboldali botkormány-pozíciót vezérli.
A játék pillanatnyi értékének leolvasása programból egy kicsit nehézkes (ld. a 8.7 alfejezetben), mivel a játékok és a billentyűzet a CIA1 és a CIA2 azonos vonalait használják. Mint már említettük, egyszerre két pár játékot használhatunk és ugyanakkor a Commodore 64-esen csak két analóg átalakító van, egy átalakítóra tehát két játékot kell rákötni. Ehhez felhasználjuk a CIA1 további két bitjét.
Az A/D átalakító lekérdezésének ideje alatt le kell tiltanunk a billentyűzetet, de csak erre az időre, hogy közben azért a billentyűzet használható legyen. A megoldást a következő gépi kódú rutin szolgáltatja, amelyhez mellékeljük a DATA sorokat, a BASIC betöltőprogrammal együtt. A rutin kényelmes hozáférést biztosít mind a négy játék állapotának lekérdezéséhez.
A tárban olyan helyet kerestünk, amelyet az operációs rendszer nem használ. Az érkező adatokat azokra a tárcímekre helyeztük, amelyeket egyébként csak a kazettás műveletek vesznek igénybe. A gépi kódú program BASIC betöltőprogramját az alábbiakban közöljük. Csatlakoztassuk a játékot a géphez, indítsuk el a programot és nézzük meg mi történik.

P1/1.ASS

```
CFBE                100       * = $CFBE
CFBE                110       ;
CFBE 78             120       SEI          ;BILLENTYUZET LEKERDEZES MEGTILTASA
CFBF A9 80          130       LDA #$80     ;AZ 'A' PADDLE PARAMETERE
CFC1 20 EC CF       140       JSR $CFEC    ;AZ A/D ATALAKITO ERTEKEINEK BEOLVASASA
A1,A2)
GFC4 8E 3C 03       150       STX $033C    ;ES TAROLASA
CFC7 8C 3D 03       160       STY $033D
CFCA AD 00 DC       170       LDA $DC00    ;AZ 'A' PADDLE NYOMOGOMB ALLASANAK BEOLV.
SASA CIA1-BOL
CFCD 29 0C          180       AND #$0C     ;A SZUKSEGES BITEK KISZURESE
CFCF 8D 9F 02       190       STA $029F    ;ES TAROLASA
CFD2 A9 40          200       LDA #$40     ;A 'B' PADDLE PARAMETERE
CFD4 20 EC CF       210       JSR $CFEC    ;AZ A/D ATALAKITO ERTEKEINEK BEOLVASASA
B1,B2)
CFD7 8E 3E 03       220       STX $033E    ;ES TAROLASA
CFDA 8C 3F 03       230       STY $033F
CFDD AD 01 DC       240       LDA $DC01    ;A 'B' PADDLE NYOMOGOMB ALLASANAK BEOLV.
```



```
ASA CIA2-BOL
CFE0 29 0C          250       AND #$0C     ;A SZUKSEGES BITEK KISZURESE
CFE2 8D A0 02       260       STA $02A0    ;ES TAROLASA
CFE5 A9 FF          270       LDA #$FF     ;OSSZES BIT KIMENETRE A CIA1-BE
CFE7 8D 02 DC       280       STA $DC02    ;A BILLENTYUZET LEKERDEZES
CFEA 58             290       CLI          ;ENGEDELYEZESE
CFEB 60             300       RTS          ;VISSZA A BASIC PROGRAMBA
CFEC 8D 00 DC       310       STA $DC00    ;A PADDLE KESZLET KIVALASZTASA
CFEF 09 C0          320       ORA #$C0     ;ES A MEGFELELO BITEK
CFF1 8D 02 DC       330       STA $DC02    ;KIMENETRE ALLITASA
CFF4 A2 00          340       LDX #$0      ;KESLELTETO CIKLUS
CFF6 CA             350       DEX          ;AZ A/D INPUT
CFF7 D0 FD          360       BNE $CFF6    ;BIZTONSAGOS ELVEGZESEHEZ
CFF9 AE 19 D4       370       LDX $D419    ;1. A/D INPUT
CFFC AC 1A D4       380       LDY $D41A    ;2. A/D INPUT
CFFF 60             390       RTS          ;VISSZATERES A FOPROGRAMBA
D000                65535     .END
```

P1/2

```
10 DATA 120,169,128, 32,236,207,142
15 DATA  60,  3,140, 61,  3,173
20 DATA   0,220, 41, 12,141,159,  2
25 DATA 169, 64, 32,236,207,142
30 DATA  62,  3,140, 63,  3,173,  1
35 DATA 220, 41, 12,141,160,  2,169
40 DATA 255,141,  2,220, 88, 96,141
45 DATA   0,220,  9,192,141,  2
50 DATA 220,162,  0,202,208,253,174
55 DATA  25,212,172, 26,212, 96
60 FOR M=53182 TO 53247
70 READ A:POKE M,A:NEXT:REM A GEPI KODU PROGRAM TOLTESE
80 AX=830: REM 1. PADDLE AZ 1. PORTON
90 AY=831: REM 2. PADDLE AZ 1. PORTON
100 BA=672 :REM 'A' PADDLE TUZGOMBJA
110 BX=828 :REM 1. PADDLE A 2. PORTON
120 BY=829 :REM 2. PADDLE A 2. PORTON
130 BB=830 :REM 'B' PADDLE TUZGOMBJA
140 SYS 53182:REM ALLAPOTOLVASAS
150 PRINT PEEK(AX)" "PEEK(AY)" "PEEK(BA)" ";
160 PRINT PEEK(BX)" "PEEK(BY)" "PEEK(BB)
170 GOTO 140
```

2.2 A SID 6581-es programozása

Az előző alfejezetekben ismertettük a SID legfontosab regisztereit, azok jelentését, illetve az A/D átalakító kezelését. Most rátérünk a SID programozására, bemutatjuk, hogy hogyan lehet a hangokat BASIC program segítségével megszólaltatni.
Ha alaposan áttanulmányoztuk az egyes regiszterek jelentését, és ismerjük a tárcímeket, ahol a regiszterek találhatóak, néhány POKE utasításból felépíthetjük a hangkezelő programot.
A regiszterek értékeinek betöltésekor célszerű az alábbi sorrendet követni:

1) betöltjük a 24-es regisztert;
2) rögzítjük a megfelelő ADSR (5,6,12,13,19,20) regiszterek értékét;
3) a hangok sorrendjében betöltjük a többi regisztert, a 4-es, a 11-es és a 18-as regiszterek kivételével, ezeknek utoljára adunk értéket.

Ha nem feledkeztünk meg a 0. bitről, a kiválasztott hang sikeresen megszólal.
Az alábbi rövid BASIC program szemlélteti a szintetizátor lehetséges hullámformáit és a hangterjedelmet.

P2

```
10 S1=54272 :REM 1. HANG
20 S2=54279 :REM 2. HANG
30 S3=54286 :REM 3. HANG
40 FL=54293 :REM A SZURO ALSO BYTE-JA
50 FH=54294 :REM A SZURO FELSO BYTE-JA
60 RS=54295 :REM RESONANCIA + KAPCSOLO
70 PL=54296 :REM PASS MOD  + HANGERO
80 POKE S1+4,0:POKE S2+4,0:POKE S3+4,0
100 POKE S1+2,0:POKE S1+3,8
110 POKE S1+5,0:POKE S1+6,240
120 POKE RS,0:POKE PL,15
130 PRINT"HAROMSZOG"
140 T=16:GOSUB 300
150 PRINT"FURESZFOG"
160 T=32:GOSUB 300
170 PRINT"NEGYSZOG"
180 T=64:GOSUB 300
190 PRINT"ZAJOK"
200 T=128:GOSUB 300
210 END
300 POKE S1,0:POKE S1+1,0
310 POKE S1+4,T+1:REM HANG BEKAPCSOLASA
320 FOR I=0 TO 255:FOR J=0 TO 255 STEP50
330 POKE S1,J:POKE S1+1,I
340 NEXT J,I
350 POKE S1+4,T:REM HANG KIKAPCSOLASA
360 RETURN
```

A program 10-től 80-ig terjedő sorai a regiszterek tárcímeit tartalmazó értékadó utasítások. Ezt a néhány sort illesszük be minden hangkezelő program elejére, konstans tárcímek helyett ugyanis könnyebb a program további részében változókra hivatkozni.
A következő mintaprogrammal a burkológenetárorral elérhető hanghatásokat mutatjuk be. A 10-től 80-ig terjedő sorokat átvettük az előző programból:

P3

```
100 A=9:D=9:S=8:R=9:H=400
110 POKE S1+5,16*A+D:POKE S1+6,16*S+R
120 POKE RS,0:POKE PL,15
130 POKE S1,37:POKE S1+1,17
140 POKE S1+4,33
150 FOR I=0 TO H: NEXT
160 POKE S1+4,32
```

Ha a 100-as sor értékadó utasításaiban változtatjuk a konstansokat, megfigyelhetjük, hogy az egyes paraméterek hogyan befolyásolják a megszólaló hang jellegét.
Az A, D, S, R változók értéke csak 0 és 15 közé eshet, egyébként az ILLEGAL QUANTITY ERROR hibaüzenetet kapjuk.
Az Olvasó már bizonyára rájött a 100-as sorban elhelyezett változók jelentésére. Az A változó a hang felfutási idejét, a D a kitartási szintre való visszaesés időtartamát, a H változó a kitartás időtartamát az S szinten, végül az R azt az időtartamot határozza meg, amely alatt a hangerő ismét nullára esik vissza, ha a KEY bitek értéke ismét 0.
Ha az R változó értéke nagyobb, mint 0, akkor a hangot a 4-es regiszter 0-ra állításával nem szabad kikapcsolni, így ugyanis a hang azonnal elhallgatna, és az R változónak semmi hatása nem lenne. Használjuk fel inkább a hangerő vezérlésére a 0. bitet, ezt állítsuk 0-ra úgy, hogy közben a többi bitjének értéke megőrződjön, azaz a 0. bit törlésével kapott páros számot töltsük vissza a regiszterbe.
Erre vonatkozóan a CBM 64-es kézikönyvben találunk szemléltető programokat.

A következő program (a 10-től 80-ig terjedő sorokkal kiegészítve) megszólaltat egy három akkordos hangzatot a spinéten (a spinét régi húros ütőhangszer, a zongora őse).

P4

```
100 A=0:D=1:S=13:R=10:H=100
110 POKE S1+5,16*A+D:POKE S1+6,16*S+R
120 POKE S2+5,16*A+D:POKE S2+6,16*S+R
130 POKE S3+5,16*A+D:POKE S3+6,16*S+R
140 POKE RS,0:POKE PL,15
150 POKE S1,37:POKE S1+1,17
160 POKE S2,154:POKE S2+1,21
170 POKE S3,177:POKE S3+1,25
180 POKE S1+4,33:POKE S2+4,33:POKE S3+4,33
190 FOR I=0 TO H: NEXT
200 POKE S1+4,32:POKE S2+4,32:POKE S3+4,32
```

A következő példában a hang frekvenciáját változtatjuk a 3-as hang burkológörbéjének leolvasásával, módosításával (54300). A 100-as sorbeli adatok megváltoztatásával ismét más-más eredményre jutunk.

P5

```
100 A=9:D=9:S= 9:R= 9:H= 30
110 POKE RS,0:POKE PL,15
120 POKE S3+5,16*A+D:POKE S3+6,16*S+R
130 POKE S3+4,33
140 FOR I=0 TO H:POKE S3+1,PEEK(54300):NEXT
150 POKE S3+4,32
160 FOR I=0 TO R*4:POKE S3+1,PEEK(54300):NEXT
```

Utolsó példánk legyen mindenki számára kedvcsináló!
Hallgassuk meg egy űrbéli vállalkozás akkusztikus hangrevűjét!

P6

```
100 A=15:D=0:S= 8:R=13:H=8000
110 POKE RS,0:POKE PL,15
120 POKE S1,0:POKES1+1,30
130 POKE S2,0:POKES2+1,1
140 POKE S3,0:POKES3+1,100
150 POKE S1+5,16*A+D:POKE S1+6,16*S+R
160 POKE S1+4,129:POKE S3+4,23
170 FOR I=0 TO H:NEXT
180 POKE S1+4,128:POKE S3+4,16
```

Reméljük, hogy mintapéldáinkkal sikerült mindannyiuk érdeklődését felkelteni és legalább néhányukat a szintetizátor adottságainak kiaknázására ösztönözni! Az utóbbihoz jó szórakozást kívánunk!

2.3 A SYNTHIMAT 64

A SYNTHYMAT 64 szoftvertermék tulajdonosai a Commodore 64-es számítógépet kiváló minőségű szintetizátorrá varázsolhatják. Mi ez a szintetizátor tulajdonképpen? Egy sajátos hangszer, az elektronikus zene pontos előállításának és vezérlésének eszköze.
Míg a „természetes" hangszerek, a gitár vagy pl. a fuvola csak egyetlen hangtípus, hangzás megszólaltatására képesek, addig a szintetizátor önmagában a hangszerek széles skáláját tudja utánozni. Mindez azért lehetséges, mert a szintetizátor több modulból épül fel, melyek

mindegyike az előállított hang egy-egy tulajdonságát határozza meg.
Az egyidejűleg felcsendülő harmónikus hangok mindig újabb és újabb meglepetésekkel örvendeztetik meg a zene barátait. A szintetizátort támogató programok között két alapvetően eltérő típust különböztetünk meg.
Az egyik típus tulajdonképpen egy BASIC bővítés, amely egy sajátos utasításkészlettel megkönnyíti a zenekezelő programok elkészítését.
A SYNTHIMAT 64 nem ehhez a típushoz tartozik!
Ez a program párbeszédes formában bekéri a billentyűzetről a zene paramétereit. A választást színes, áttekinthető kép segíti; a program a képernyőn ábrázolja a szintetizátor egységeit.
Először a hangszínt kell megválasztanunk. Ez nem nehéz feladat, mert minden leütött érték azonnal hallható változást okoz a hangszínben. A választott hangszínt előre elkészített kis zenebetétek segítségével próbálhatjuk ki. Ha kedvünkre választottunk, a továbbiakban valódi szintetizátorként használhatjuk a billentyűzetet.
A SYNTHIMAT 64 polifonikus zenét szolgáltat. A polifonikus szintetizátor több billentyű egyidejű lenyomásakor akkordokat szólaltat meg.
A legnagyobb gondot az okozhatja, hogy egy valódi hangszer billentyűit képtelenség reprezentálni a billentyűzeten. Az igazi zenészek kezdetben kicsit körülményesnek fogják tartani a játékot „ezen a hangszeren", de hamarosan biztosan hozzászoknak majd az eltérésekhez. A SYNTHIMAT 64 igen nagy előnye, hogy két szólamot is eljátszhatunk vele. Az egyik szólam a kíséret, a másik a dallam (szóló) lehet, melyeket bármely hangzásra rákapcsolhatunk.
Így megtehetjük például azt, hogy egyszerre szólaltatunk meg egy zongorát és egy nagybőgőt úgy, hogy a zongora játssza a szólamot, a nagybőgő pedig a kíséretet.
A SYNTHIMAT 64 az alapmodulokon kívül 8 hanggenerátort tartalmaz, amelyekkel tovább finomíthatjuk az alaphangokat.
Az ún. TREMELO effektus például akkor keletkezik, amikor a hangerőt moduláljuk. A szűrőfrekvencia modulálása nagyon érdekes hanghatásokat eredményez, pl. a békabrekegést. Ez az ún. WAH-WAH effektus. A bővítő generátorokkal modulálhatjuk az oszcillátorunk hangfrekvenciáját, a négyszög hullámforma impulzusszélességét, stb. – az előállítható hanghatások száma szinte végtelen.
A szoftver optimálisan kihasználja a CBM 64-es tárkapacitását. A saját magunk által komponált zenedarabokat (a zongoraműtől az űrzenéig) tárolhatjuk és bármikor újra megszólaltathatjuk, összesen 256 tárcím áll a rendelkezésünkre. Alkotásainkat a gép kikapcsolása után is megőrizhetjük, ugyanis a 256 tárcím tartalma lemezen is tárolható. Ez utóbbi tulajdonsága különösen megkedvelteti a programot használóival. Lehetővé teszi ugyanis, hogy az utókor (de legalábbis közvetlen baráti körünk) számára készített zeneműveket lemezen tároljuk, és napok múltán vendégeink megérkezésekor lejátszuk anélkül, hogy egyetlen újjal is a billentyűkhöz érnénk.
Olyan Olvasóink is, akik esetleg sem a programozáshoz, sem a zenéhez nem értenek, a SYNTHIMAT 64 segítségével csodálatos zenegéppé varázsolhatják Commodore 64-esüket. (Ehhez hozzájárul a program kezelési útmutatója és viszonylag alacsony ára.)

3. FEJEZET
A GRAFIKA ÉS PROGRAMOZÁSA

3.1 A VIC 6569-es videovezérlő

A VIC szintén a 65xx processzorcsalád tagja. Azonkívül, hogy a képszerkesztéssel járó összes feladatot ellátja, a processzort is tehermentesíti a dinamikus tárkezelésből adódó időzítési feladatok kezelésével.
Röviden a 8 legfontosabbat:
- 16 szín
- 40 sor / 25 oszlop
- Hi-Res (High Resolution) grafika 320 X 200-as felbontással
- 5 üzemmód
- 8, egyenként 24 X 21 pontból álló sprite
- standard PAL jel
- eltolható karaktergenerátor
- eltolható videoram

A 6569-es lábkiosztása:

1–7	D6-D0; processzor adatbusz
2–8	– IRQ; 0, ha az IMR és az IRR egy bitje azonos
9	– LP; bemenet, fényceruza (light-pen strobe)
10	– CS; processzorbusz művelet csak CS = 0 mellett!
11	R/-W; 0 = az adat átvitele az adatbuszról
12	BA; 0, ha az adat még nem áll készen az olvasásra
13	VDD; + 12 V
14	COLOR; színinformáció kimenet
15	SYNC; jel- és kép szinkron-impulzus
16	AEC; 0 = a VIC használja a rendszerbuszt, 1 = a busz szabad
17	0OUT; a rendszerütem kimenete
18	– RAS; dinamikus RAM vezérlőjel
19	– CAS; mint fent
20	– GND
21	0COLOR; színfrekvencia bemenet
22	0IN; pontfrekvencia bemenet
23	A11; processzor címbusz
24–29	A0/A8-A5/A13; multiplexelt (video) RAM címbusz
30–31	A6-A7; (video-)RAM címbusz
32–34	A8-A10; processzor adatbusz
35–38	D11–8; adat a szín-RAM-ból
39	D7; processzor adatbusz
40	VCC; + 5 V

3.1.2 A REGISZTEREK LEÍRÁSA

A VIC47 regiszterrel dolgozik, amelyek jelentését az alábbiakban foglaljuk össze.

REG 0 A 0. sprite X koordinátája
A sprite képernyőpozíciójának X koordinátáját tartalmazza. A 9. bit a 16-os regiszterben van.

REG 1 A 0. sprite Y koordinátája
Ua. mint fent, az Y koordinátára. Ennek a regiszternek nincs átvitele.

A 2-től a 15-ig terjedő regiszterek leírása ugyanaz, mint a fenti két regiszter jelentése. A 2/3 regiszterpár az 1-es sprite-ra, a 4/5 regiszterpár a 2-es sprite-ra vonatkozik stb.

REG 16 az X koordináta MSB-je
Ebben a regiszterben találhatóak a sprite-ok X regisztereinek átvitelei. A 0. bit a 0. sprite-ra, az 1. bit az 1. sprite-ra vonatkozik stb.

REG 17 1. vezérlőregiszter
0–2. bit:a felső képkeret ábrázolása a rasztersorokban
3. bit 0:= 24 sor, 1 = 25 sor
4. bit 0:= képernyő kikapcsolása
5. bit 1:= standard bittérkép üzemmód
6. bit 1:= bővített színes üzemmód
7. bit:a 18-as regiszter átvitele

REG 18 annak a rasztersornak a sorszáma, amelyben a fényceruza IRQ-t váltott ki. Átvitele a 17-es regiszterben található.

REG 19 annak a képernyőpozíciónak az X koordinátája, amelyben a fényceruza a STROBE jel kiváltásakor található (a fényceruza észlelésének pozíciója, LP = 0).

REG 20 Ua. mint előbb, az Y koordinátára vonatkoztatva.

REG 21 a sprite engedélyezése
Minden sprite-hoz tartozik egy bit. Ha ennek értéke 1, a sprite be van kapcsolva (látható).

REG 22 2. vezérlőregiszter ISZTER
0–2. bit:a baloldali képernyőkeret ábrázolása
3. bit 0:= 38 karakter, 1 = 40 karakter
4. bit 1:= színes üzemmód

REG 23 A sprite X irányú nyújtása
Minden sprite-hoz tartozik egy bit. Ha ennek értéke 1, a megfelelő sprite kétszeres nagyításban jelenik meg.

REG 24 A karaktergenerátor és a videoram báziscíme
1–3. bit: a karaktergenerátor báziscímének 11–13-as bitjei
4–7. bit: a videoram báziscímének 10–13-as bitjei.

A 24-es regiszter leírása a CBM 64-es kézikönyvben megtalálható

REG 25 IRR Interrupt Request Register
0. bit: raster megszakítás (18-as regiszter)
1. bit: sprite-háttér megszakítás (31-es regiszter)
2. bit: sprite-sprite megszakítás (30-as regiszter)
3. bit: fényceruza megszakítás (LP láb)
7. bit: 1 = ha a fentiek közül legalább egy bit értéke 1.

REG 26 IMR Interrupt Mask Register
kiosztása azonos a fentivel. Ha az IRR és az IMR bitjei közül legalább egy megegyezik, akkor az IRQ = 0.

REG 27 Minden sprite-hoz tartozik egy bit. 1 = a háttérpontok takarják a sprite pontjait (a háttér prioritása nagyobb)

REG 28 Minden sprite-hoz tartozik egy bit. 1 = a sprite színes üzemmódban látható (Multicolor Mode)

REG 29 Y irányú nyújtás
Minden sprite-hoz tartozik egy bit. 1 = a sprite függőlegesen kétszeresére nyúlik.

REG 30 Sprite-Sprite ütközés
Minden sprite-hoz tartozik egy bit. Ha egy sprite hozzáér egy másikhoz, a megfelelő bit értéke 1 lesz. Ezzel egyidejűleg az IRR 2. bitje is 1-re változik. Ütközés után a regiszter bitjét törölni kell, egyébként nem tudjuk követni a későbbi ütközést.

REG 31 Sprite-háttér ütközés
Ua. mint előbb, a sprite-háttér ütközésére vonatkoztatva.

REG 32 Keretszín

REG 33– Háttérszín (0–3 bitek)
REG 36
REG 37– Többszínű sprite (0–1 bitek) (multicolor)
REG 38
REG 39– A sprite-ok (0-tól 7-ig) színe (sprite color)
REG 46

3.1.3 A VIC ÁBRÁZOLÁSI MÓDJAI

A VIC három különböző ábrázolásmódot ismer. A karakteres-, a pontonkénti-, és a figurális (sprite-os) ábrázolási módot.

A figurális ábrázolási mód:

A sprite a programozás által meghatározott, a képernyő bármely pontján megjeleníthető, mozgatható figura. (A sprite szó jelentése magyarul: szellem). A mozgatást egy 512 X 256 pontból álló mezőben két regiszterpárral vezérelhetjuk. A mozgás tartománya nagyobb a látható képernyőterületnél, így a figurát a látható területen kívülre is pozicionálhatjuk.
Az ábrázolási mód fontos jellemzői:

1) Egyidejűleg maximum nyolc, 0-tól 7-ig számozott figurával dolgozhatunk.
2) A figurákat vízszintes-, függőleges-, esetleg mindkét irányba megnyújthatjuk.
3) A figurákat egymástól függetlenül elhelyezhetjük a képernyőn.
4) A figurákat kétféleképpen ábrázolhatjuk; finom felbontásban vagy többszínű grafikában.
5) A figurák elsődlegessége egymással szemben rögzített. A legmagasab prioritású a 0. figura. Ez azt jelenti, hogy ha a képernyőn fedésbe kerül pl. a 0. és az 1. figura, a 0. figura lesz látható.
6) A figurák elsődlegesek a háttérben ábrázolt alakzatokkal szemben. A háttéralakzatok karakterek vagy grafikus ábrák lehetnek, ezek színe azonos a háttér színével.
7) A figurák egymás közötti átfedését, az ütközést a Commodore 64-esen automatikusan figyelhetjük.
8) A háttér és a figura ütközését szintén figyelhetjük.

A figurák finom felbontású ábrázolásmódját választva, a figurák 24 X 21 pontból állnak. Ez annyit jelent, hogy egy figura meghatározásához 504 bitre, azaz 63 byte-ra van szükségünk. Ha egy bit értéke 1, a megfelelő pont a figura színét (39—46. regiszterek), egyébként a háttér színét veszi fel, tehát ebben az esetben a 24 X 21-es téglalap két különböző színt vehet fel.

A többszínű (multicolor) figura 12 X 21 pontból áll úgy, hogy a vízszintesen elhelyezhető pontok dupla szélességűek.
Ebben az esetben is 504 bitet használunk egy figura meghatározására, de a bitek páronként határoznak meg egy-egy pontot. A figura színezésére ekkor négy különböző színt használhatunk.
Ha ezt az ábrázolásmódot választjuk, a 28-as regiszter megfelelő bitjét 1-re kell állítanunk.

A bitpárok jelentése:

Bitpár	Szín	Cím
00	változó	képernyő színe
01	0. MC regiszter	$D025 (53285)
10	sprite color	$D027—$D02E
11	1. MC regiszter	$D026 (53286)

A VIC-nek tudnia kell, hogy egy adott figura leírása hol található a tárban. Ezt az információt a VIC mindig a képernyőtár utolsó (legfelső) nyolc byte-ján keresi. Azaz, ha normál üzemmódot használunk, a tár $7F8 (2040) címétől kezdve $7FF6 (2047)-ig terjedő byte-jaiba kell tárolni az egyes figurákat leíró tárterületek kezdőcímét. A 0-s figura mutatója a 2040-es címen, az 1-es figura mutatója a 2041-es címen található stb.

Karakteres ábrázolási mód:

Ebben az üzemmódban a VIC betölt egy byte-ot a videotárból (RAM) és a byte tartalmát a karaktergenerátoron belüli mutatóként értelmezi. A mutató által megadott tárcímen tárolt bitmintát megjeleníti a képernyőn. A karaktergenerátor összesen 256 különböző karaktert tartalmaz. A karakterek színét a színtárból határozza meg. Minden karakterhez, azaz minden képernyőpozícióhoz négy bit tartozik, és minden négybites szám a 16 lehetséges szín valamelyikére utal. A karakter 8 X 8 pontja közül a látható pontok (a megfelelő bit értéke 1) a kiválasztott színben, a többi pont pedig a háttér színében jelenik meg.

Karakterek megjelenítése színes üzemmódban (multicolor):
(22-es regiszter 4. bitje 1)

Ebben az üzemmódban a megjelenítés a szín-RAM 4 bitjének vizsgálatával kezdődik. Ha a legmagasabb helyiértékű bit értéke 0, a VIC a karaktert a megszokott 8 X 8-as pontmátrix formájában ábrázolja.
A látható pontok színét (ahol a bitmintákban 1 áll) ekkor a maradék három bit értéke határozza meg, a többi pont színe pedig ismét azonos a háttér színével.
Ha a legnagyobb helyiértékű bit értéke 1, akkor a színt bitpárok határozzák meg. A karakter ekkor 4 X 8-as méretű, ismét dupla szélességű pontokkal. A színt ekkor a VIC a következő hozzárendelés szerint keresi:

00 = 0. háttérszín regiszter
01. = 1. háttérszín regiszter
10. = 2. háttérszín regiszter
11. = a színtár alsó három bitje

Ez az ábrázolásmód négyszínű karaktereket eredményez:

Bővített színes üzemmód (extended color mode):
(A 17. regiszter 6. bitje = 1)

A bővített színes üzemmód hasonló a normál karakteres ábrázolásmódhoz, azzal az eltéréssel, hogy ebben az esetben a háttér színe karakterenként eltérő is lehet.
A látható pontok színét most is a színtár tartalma határozza meg.
A háttérpontok színét azonban a VIC a videotár két felső bitjéből az alábbi hozzárendeléssel választja ki:

00 = 0. háttérszín regiszter
01.= 1. háttérszín regiszter
10.= 2. háttérszín regiszter
11 = 3. háttérszín regiszter

Minthogy azonban így a videotár adott byte-jának két bitjét elhasználtuk a szín meghatározására, a maradék hat biten csak 64 különböző címet, azaz 64 különböző karaktert tudunk megadni.

Finom felbontású ábrázolási mód (high resolution bit mode)
(A 17-es regiszter 5. bitje = 1)

Ez az ábrázolási mód a képernyő és a tár közvetlen egymáshozrendelésen alapszik. A tár egy bitje megfelel a képernyő egy pontjának és megfordítva. A teljes leképezéshez 8 kbyte tárterületre van szükség.
A videotárat ekkor színtárként használjuk, a normál színtárnak pedig nincs feladata.
Az ábrázoláskor a képernyőt 320 X 200 pontra bontjuk fel. A videotár minden byte-ja meghatározza a képernyő egy 8 X 8-as egységének színét úgy, hogy a felső félbyte a látható pontok színét, az alsó félbyte pedig a többi pont színét adja meg.

Színes finom felbontású ábrázolásmód (multicolor bit mode)
(A 17-es regiszter 5. bitje 1, a 22-es regiszter 4. bitje = 1)

A teljes képernyőt ekkor 160 X 200 pontra bontjuk, mivel a tár minden bitpárja egy dupla szélességű ponthoz tartozik.
A pontok színét a VIC az alábbi hozzárendelés alapján keresi:

00 = 0. háttérszín regiszter
01 = a videotár adott címének felső félbyte-ja
10 = a videotár adott címének alsó félbyte-ja
11 = színtár

A videotár és a karaktergenerátor áthelyezésének lehetőségéről a 3.3 és 3.4 fejezetekben lesz szó.

3.2 Illesztés a processzorhoz

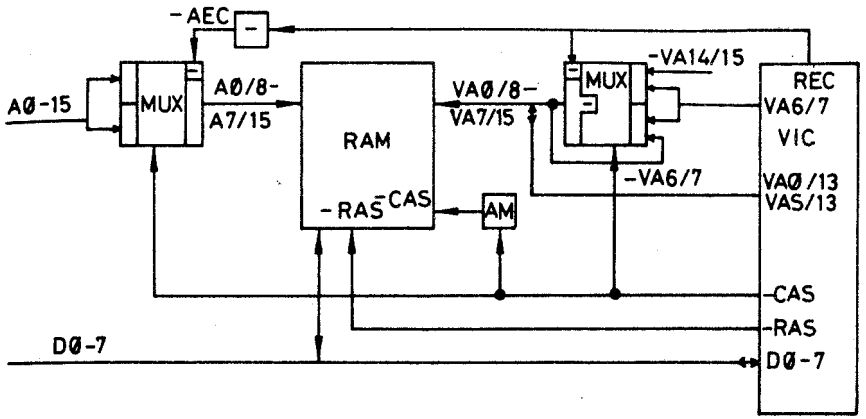
A VIC chipet szabályos külső egységként illesztették a processzorhoz, így a külső egységekhez hasonlóan korlátozás nélkül írhatunk bele, illetve olvashatunk belőle.
Minthogy azonban a VIC állítja elő az órajelet, jóval szélesebb hatáskörrel rendelkezik, mint egy egyszerű külső egység. Kézben tartja a rendszerbusz ütemezését, ezáltal engedélyezheti, letilthatja a hozzáférést, ki tudja szolgálni a dinamikus RAM-ot stb., minden időpillanatban tudja, hogy éppen milyen műveletek megengedettek, sőt a rendszerbusz vezérlését el is veheti a processzortól. Az utóbbira mindig szükség van, valahányszor a VIC a videoramot, a karaktergenerátort, illetve a szín-RAM-ot használja. Alapvető feladatát, a képernyőn megjelenő kép állandó felfrissítését csak úgy tudja ellátni, ha a videoramhoz, a karaktergenerátorhoz ill. a szín-RAM-hoz ciklikusan hozzáfér.
Működésével nem zavarja a processzor munkáját, ugyanis ügyesen azokat a pillanatokat használja ki, amikor a processzornak nincs szüksége a rendszerbuszra.
Az összehangolás nagyon egyszerű: 02 = 1-nél a processzor, egyébként (02 = 0) a VIC foglalja le a rendszerbuszt.
Fizikailag a buszt a VIC az AEC láb 0-ra állításával teszi szabaddá, míg a processzor nagy ellenállású állapotba (Tri-State) hozza. Így egyik sem zavarja a másikat.

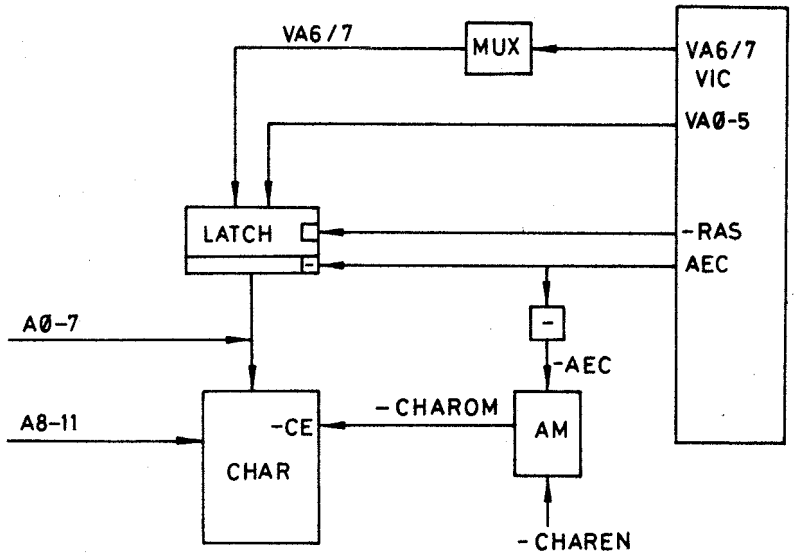
3.3 Illesztés a RAM-hoz

A VIC csak 16 kbyte-ot tud önállóan megcímezni. A videoram, illetve a karaktergenerátor áthelyezése után a VIC ezekhez csak akkor tud hozzáférni, ha a címzéshez még két további bitet kívülről felhasználhat. A CIA 2 ilyen esetekben a VIC rendelkezésére bocsájtja a PA0-1 biteket. Ha ezt a két bitet megváltoztatjuk, a videoram 16 kbyte-tal tolódik el, ugyanis a két bit a videoram 14. és 15. címbitjét adja.
Ha elegendő tárterület áll rendelkezésre, a PA0–1 bitekkel több képernyőlapot kezelhetünk egyszerre.
Figyelnünk kell persze arra, hogy ezek a bitek L-aktívak, ha tehát az alsó 16 kbyte-ot akarjuk választani, akkor mindkét bitet 1-re, a felső 16 kbyte választása esetén pedig mindkét bitet 0-ra kell állítani. Ez lehetőséget ad pl. arra, hogy Hi-Res üzemmódban, míg az egyik képet megjelenítjük, egy másik kép már készen várakozzon a tárban.
A videoram finomabb, 1 kbyte-os eltolásait a VIC 24-es regiszterével vezérelhetjük. A regiszter 4–7. bitjei ugyanis megfelelnek a 10–13-as címbiteknek.
Az elmondottakat szemlélteti a következő oldal felső ábrája.

3.4 A karaktergenerátor illesztése

Az alábbiakat a következő oldal alsó ábrája szemlélteti.
A CIA bitek értékének módosítása a videoram eltolásával egyidejűleg a karaktergenerátor azonos eltolását is eredményezi. A 24-es regiszter (VIC) 1–3. bitjei ugyanakkor 2 kbyte-os

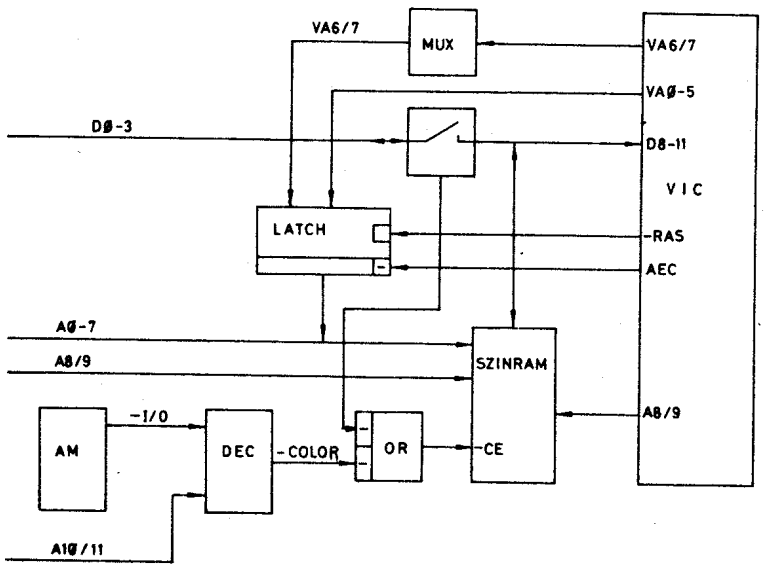




eltolást jelentenek, ugyanis a 11–13-as címbiteknek felelnek meg. A CBM 64-esbe beépített karakter-ROM különleges helyzetben van az AM-nek (address manager) köszönhetően. A VIC csak akkor éri el a karaktergenerátort, ha az a $1000, illetve $9FFF relatív címen található. Valójában a karaktergenerátor a processzor SD000-s címén kezdődik.
Minthogy a karakter ROM nem a VIC címtartományában fekszik, az eléréshez ismét további címbitekre van szükség, amelyeket egy különleges pufferben kell tárolni. A hiányzó bitek a 0–7. címbitek A 24-es regiszter fontos szerepet tölt be: nemcsak a generátor kezdőcímét rögzíti, hanem finom felbontás esetén a 3-as bitje határozza meg a képernyőtár helyzetét is Ekkor az 1-es és a 2-es biteknek nincs jelentősége.

3.5 Illesztés a szín-RAM-hoz

A szín-RAM-ot, szemben a videorammal és a karaktergenerátorral, nem lehet eltolni. A címzése éppúgy, mint a karakter-ROM címzése, kétféleképpen történhet: a processzorbuszról, illetve a VIC buszról.
A címzéshez most is hiányoznak a 0–7-es címbitek, amelyeket a VIC ugyanolyan módon pótol, mint a karakter ROM esetében. A szín-RAM-ot a VIC-kel egy saját adat-interface köti össze. Ez az interface egy kapcsolóval leválasztható a processzor adatbuszról. A megoldás a következő:
Ha AEC = 0 (a VIC vezérli a buszt) a kapcsoló a VIC irányába,
ha AEC = 1, ellenkező irányba mutat Az írás/olvasás hozzáférést a processzor oldaláról az AM vezérli.
Nézzük meg a következő kapcsolási rajzot.



3.6 A szín- és grafika programozása

A gazdag grafikai lehetőségek és a beépített szintetizátor révén a Commodore 64-es kiemelkedik a személyi számítógépek sorából. A grafika és a zene azonban mindaddig csak lehetőség marad, amíg meg nem ismerkedünk a kezelésükhöz szükséges programozási technikával.
Szeretnénk, ha az Olvasó áttanulmányozva ezt a fejezetet, olyan ismeretek birtokába jutna, melyekkel életre keltheti a gépbe épített gazdag lehetőségeket.
A Commodore 64-es a vele egy ár- és teljesítmény-kategóriába tartozó személyi számítógépek között messze a legjobb grafikus adottságokkal rendelkezik.
Ebben a vonatkozásban a VC 20-as, a kisebb testvér is mögötte marad a Commodore 64-esnek.
A Commodore 64-es legsajátosabb adottsága a többi gépekhez képest a sprite-kezelés, az a

hardveresen megalapozott lehetőség, hogy a képernyőn egymástól függetlenül 8 különböző sprite-ot mozgathatunk.
A sprite-ok különböző színűek lehetnek. Összesen 16 szín áll rendelkezésre. Sokszínű üzemmódban egy sprite színezésére is három különféle színt használhatunk. A sprite-ok 24 X 21 pontot tartalmazó téglalapok, tehát a méretük kb. megfelel egy 3 X 3 betűt tartalmazó blokknak.
A sprite-ok programozási lehetőségeire még visszatérünk, előzetesen csak annyit jegyzünk meg, hogy a programból lekérdezhető pl. a képernyőn elfoglalt helyzetük, megvizsgálhatjuk, hogy két különböző sprite ütközik-e a képernyőn stb. Mielőtt a sprite-kezelés részleteibe belemélyednénk, foglalkozzunk a grafika és a színek programozásával.

A grafika és a színek programozása

Mint már említettük, a Commodore 64-esen 16 különböző színnel dolgozhatunk. Tetszőlegesen választhatunk, hogy a képernyőn a háttér és az előtér színe milyen legyen. A karakterek színét változtathatjuk a billentyűzetről is. A CTRL és egy színbillentyű egyidejű megnyomásával a 0-tól 7-ig, a SHIFT és egy színbillentyű egyidejű megnyomásával a 8-tól 15-ig számozott színeket kapjuk:

Billentyű	Szín	Ismertetőszám
CTRL–1	Fekete	0
CTRL–2	Fehér	1
CTRL–3	Vörös	2
CTRL–4	Türkiz	3
CTRL–5	Lila	4
CTRL–6	Zöld	5
CTRL–7	Kék	6
CTRL–8	Sárga	7
C= –1	Narancs	8
C= –2	Barna	9
C= –3	Rózsaszín	10
C= –4	1. Szürke	11
C= –5	2 Szürke	12
C= –6	Világoszöld	13
C= –7	Világoskék	14
C= –8	3. Szürke	15

A színek változtatása közben előfordulhat hogy a képernyőnkön a karakterek nem elég jól, vagy egyáltalán nem láthatóak. Az eredmény ugyanis mindig a háttér és a karakterek színének összhatásától függ. Nagyon rosszul látható például egy zöld írás fekete előtérben fekete kerettel. Hosszabb rövidebb kísérletezés után mindenki megtalálja a számára legkedvezőbb színösszeállítást. Az előtér és a keret színét az 53280-as illetve az 53281-es tárcímeken keresztül módosíthatjuk. A két tárcímre be kell írni POKE utasítással a kiválasztott szín (színek) kódját. Például a

```
POKE 53280,0
POKE 53281,0
```

utasítások végrehajtása után teljesen fekete képernyőt kapunk. A fenti POKE utasítások nem befolyásolják a karakterek színét. A karakterek színét is megváltoztathatjuk úgy, hogy a karakternek megfelelő tárcímre írjuk be a megfelelő színkódot Pl. a

```
POKE 55296,0
```

utasítás a képernyő bal felső sarkában levő karakter színét feketére változtatja. Az összes többi karakter színe változatlan. A fenti POKE utasítás első argumentuma a Commodore 64-es szín-RAM-jának az első címe. A szín-RAM tartalmazza a képernyőn ábrázolt összes karakter színét. A teljes szín-RAM 55296-tól 56295-ig terjed A képernyőn ábrázolható 1000 karakter mindegyike számára egy byte van fenntartva. Itt az 55296 cím az 1. sorra és az 1. oszlopra, az 55297 cím az 1. sorra és a 2. oszlopra stb. vonatkozik. Végül az 56295 cím a 25. sor és a 40. oszlop színinformációját tartalmazza.
Ugyancsak a képernyő karaktereinek ábrázolására szolgál a képernyőtár. A képernyőtár tartalmazza a tulajdonképpeni karaktereket a szín meghatározása nélkül A képernyőtár az 1024-től 2023-ig terjedő RAM területen található.
A tárcímek jelentését ismerve, ha pl a teljes képernyőt ki akarjuk tölteni A betűkkel, a következő programot kell elkészítenünk

```
P?,

10 PRINTCHR$(147);:REM KEPERNYO TORLESE
20 FOR I=1024 TO 2023: REM KEPERNYO TERULET
30 POKE 54272+I,0:REM SZINTERULET (54272+1024=55296)
40 POKE I,1:REM 'A'KOD=1,'B'KOD=2, STB
50 NEXT:REM A TELJES KEPERNYO ES SZINTERULET
60 GOTO 60:REM EZZEL A KEPERNYOT NEM IRJUK FELUL
```

A program végrehajtása során a képernyő balról jobbra megtelik A betűvel.
Ha azt akarjuk, hogy az A betűk színe véletlenszerűen változzon, módosítsuk a 30-as sort a következőképpen:

```
30 POKE 54272+I, INT(RND(1) * 16)
```

A program most is ugyanúgy fut le, mint az első változatnál, de az A betűk különböző színekben jelennek meg. A programsor megértéséhez ismernünk kell a beépített véletlenszám generátort
Véletlen számok generálására nagyon gyakran szükség van, különösen a játékprogramok készítéséhez.
Az RND (random) függvény alkalmazása rendkívül egyszerű. Ha például egy véletlen szín előállításához egy 0 és 15 közötti véletlenszámot szeretnénk kapni, a gép által előállított 0 és 1 közé eső számot a 0 és 15 közé eső egész számok tartományára kell leképezni, amelyre kiválóan alkalmas az INT függvény (NT(RND(1) *16)).

Ezzel természetesen a Commodore 64-es grafikus ábrázolási lehetőségeit még korántsem merítettük ki. A finom felbontású színes grafika kezelését a fejezet végén található programmal illusztráljuk. A program bemutatja néhány, a grafika programozásához fontos tárcím alkalmazását és ezek hatását.
Tekintsük először át a grafika-programozással kapcsolatos legfontosabb kérdéseket.
A Commodore 64-esen sajnos a körök, vonalak és egyéb geometriai alakzatok megjelenítése programozástechnikailag nem olyan egyszerű, mint néhány más típusú gépen. Nincs például olyan utasítás, amellyel az X és az Y pont között egy egyenest húzhatnánk, vagy amellyel meghatározott középpontú, meghatározott sugarú kört rajzolhatnánk. Az említett utasítás hiánya azonban természetesen nem jelenti azt, hogy a Commodore 64-es minderre nem képes. Csak meg kell értetni vele, hogy tulajdonképpen mit is akarunk.
A grafikával foglalkozó fejezet végén egy részletes segédprogramot találunk, amely megkönnyíti a grafikus ábrázolást A szerzők a programnak a HI-RES GRAFIKA SEGÉDLETE nevet adták, segítségével ugyanis gyorsan és kényelmesen állítnatunk elő egyszerűbb grafikát Komolyabb feladatokra, összetettebb alakzatok előállítására is készítettek a szerzők egy programot, amely SUPERGRAPHIK 64 néven a szaküzletekben beszerezhető.

A dolgokat azonban nem szabad elhamarkodni; az a javaslatunk, hogy az Olvasó csak az elsőként említett szerényebb programmal kísérletezzen.
Térjünk most vissza az említett rajzoló programhoz.
Ennek a programnak egyrészt az a feladata, hogy egy szinuszgörbét állítson elő a képernyőn, ezzel demonstrálva a Commodore 64-es grafikus ábrázolási lehetőségét.
A programban szereplő regiszterek jelentését megtalálja az Olvasó a grafika-processzor regisztereivel foglalkozó fejezetben.

```
!B

10 REM SINUS-PLOT PROGRAM
20 V=53248:REM GRAFIKA-PROCESSZOR KEZDOCIME
30 AD=8192:REM HI-RES BIT TERKEP KEZDOCIME
40 POKE V+17,59:REM A NAGYFELBONTASU GRAFIKA BEKAPCSOLASA
50 POKE V+24,24
60 FOR I=1024 TO 2023:REM A HIRES SZIN-RAM BEALLITASA
70 : POKE I,16: REM SZINKOD
80 NEXT I
90 FOR I=8192 TO 16383:REM HI-RES BIT TERKEP TORLESE
100 : POKE I,0
110 NEXT I
120 FOR X=0 TO 319:REM AZ X TENGELY MEGRAJZOLASA
130 : Y=100:REM AZ X TENGELY HELYZETE
140 : GOSUB 1000 :REM A RAJZOLO RUTIN MEGHIVASA
150 NEXT X
160 FOR Y=0 TO 199:REM AZ Y TENGELY MEGRAJZOLASA
170 : X=160:REM AZ Y TENGELY POZICIOJA
180 : GOSUB 1000 :REM A RAJZOLO RUTIN MEGHIVASA
190 NEXT Y
200 X=0
210 FOR I= TO STEP/160
220 REM AZ INTERVALLUMOK HATARAI
230 Y=100+99*SIN(I): REM A FUGGVENY EGYENLETE
240 : GOSUB 1000
250 : X=X+1
260 NEXT I
270 GOTO 270:REM A KEPERNYO VEDELME
1000 OY=320*INT(Y/8)+(Y AND 7):REM A PONT HELYZETENEK KISZAMITASA
1010 OX=8*INT(X/8)
1020 MA=2^((7-X) AND 7)
1030 AV=AD+OY+OX
1040 POKE AV,PEEK(AV) OR MA:REM A PONT KIRAJZOLASA
1050 RETURN
```

Ha az Olvasó megértette a program felépítését, rövidesen hasonló programmal különböző matematikai függvényeket is ábrázolhat. Érdemes megkísérelni egy általános függvény-rajzoló program elkészítését, amely a függvény egyenletének és az ábrázolandó tartomány határainak beolvasása után kirajzolja a képernyőre vagy nyomtatóra a függvény képét A „nyomtató" és a „grafika" témánál valamelyest időznünk kell, a grafika nyomtatása ugyanis meglehetősen bonyolult programozási feladat Ez a könyv két ún hardcopy programot tartalmaz, amelyekkel a képernyő teljes tartalmát kinyomtathatjuk. Az első változat a HI-RES GRAFIKA SEGÉDLETE c. programban található és EPSON nyomtatóra dolgozik. A másik változat HARDCOPY név alatt a 6.3.2 fejezetben található.
Térjünk rá a rajzoló program elemzésére. A 20-as és a 30-as sorban beállítjuk a grafika-oldalak két kezdőcímét. Ezután a

```
POKE V+17,59 és a POKE V+24,24
```

utasításokkal bekapcsoljuk a finomfelbontású grafikát, azaz átkapcsolunk karakteres üzemmódról grafikus üzemmódra. Erre azért van szükség, mert alapállapotban a gép mindig

karakteres üzemmódot feltételez. A szöveg és a grafika egyaránt a mindenkori képernyőtárban található. A Commodore 64-es természetesen egyszerre csak egy képernyőtárral tud dolgozni, következésképpen karakteres üzemmódban nem lehet grafikát, illetve megfordítva, grafikus üzemmódban nem lehet szöveget megjeleníteni.
Ha a program egy későbbi munkafázisában vissza akarunk kapcsolni karakteres üzemmódra, a két cím (V +17 és V + 24) eredeti tartalmát vissza kell állítanunk. Ez csak úgy lehetséges, ha ezeket az értékeket megőrizzük:

A1=PEEK (V+17) és A2=PEEK (V+24)

Ahhoz, hogy a grafika-feldolgozás után ismét visszatérhessünk karakteres üzemmódba, a tárolt változókat — a mi példánkban az A1 és A2 változót — vissza kell írni a V+17 és V+24 címre:

POKE V+17,A1 és POKE V+24,A2

Az utasítás végrehajtása után a megszokott módon folytathatjuk a munkát.
Most következhet a képernyő és a grafika színének kiválasztása. A színek meghatározásához két ciklusra van szükség. Az első ciklusban a színt állítjuk be, a második ciklusban pedig töröljük a HI-RES grafika tárterületét.
Ezen a ponton érdemes egy kicsit részletesebben foglalkozni a tárolási móddal.
A képtárban minden képernyőponthoz tartozik egy bit, amelynek értéke, mint tudjuk, 0 vagy 1 lehet. Ha egy adott ponthoz tartozó bit értéke 1, a pont látható a képernyőn, egyébként nem. Mivel 8 bit alkot egy byte-ot a képernyőtár minden byte-ja 8 képernyőpontot határoz meg. A képernyőtár 1000 byte-ja (40 sor és 25 oszlop), soronként 40 X 8 = 320, oszloponként pedig 25 X 8 = 200 pontnak felel meg. A teljes képernyő felbontása tehát 320 X 200, azaz 64 000 pont.
De menjünk tovább. Vizsgáljuk meg külön-külön az egyes karaktereket. Egy karakter 8 X 8, azaz 64 pontra bontható fel. A pontok mindegyikét külön programozhatjuk, azaz eldönthetjük, hogy látható legyen vagy sem Csak a színek meghatározása okoz gondot A színtárban ugyanis csak a 8 X 8-as egységeket ábrázolhatjuk, külön-külön a pontokat nem! Ha minden egyes pont színét külön meg akarnánk szabni, a színkódok tárolásához 64000 byte-ra, azaz 64 kbyte-ra lenne szükség, ekkora színtárral pedig a C 64-es nem rendelkezik.
Meg kell elégednünk azzal, hogy a 8 X 8-as egységeket, azaz az egy karakternek megfelelő ponthalmazokat egyetlen színkóddal jellemezzük. Ez a képernyős ábrázolás szempontjából is előnyös, hiszen nagyobb színfelbontás teljesen elmosódó képhez vezetne.
Egy programozható színegység, azaz egy karakter tehát a következő szerkezetű ponthalmaz:

. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .

A 8 X 8 biten, azaz 8 byte-on tárolt információ normál üzemmódban vonatkozhat egy, a képernyőtárban található karakterre, HI-RES üzemmódban pedig a képernyő adott pozíciójú pontjaira.
Lehet, hogy első hallásra ez az ábrázolási mód kicsit bonyolultnak tűnik, de rövid töprengés után kiderül, hogy valójában nem az. Az elmondottakból következik, hogy a képernyőn

egyszerre szöveget és grafikát nem tudunk megjeleníteni (legalábbis nagy programozási ráfordítás nélkül). Első pillantásra ez nagy hátrány. Azonban a C 64-essel megegyező ár- és teljesítmény-kategóriájú személyi számítógépek egyikénél sincs lehetőség a szöveg és a grafika együttes kezelésére. Ellenkező esetben nem lenne szükség külön-külön szöveg- és grafika tárra.
A grafikus gépek ára azonban jóval meghaladja azt az összeget, amelyet azok az emberek megengedhetnek maguknak, akik nem kifejezetten számítóközpontot akarnak lakásukban berendezni. Van azonban egy kis trükk, amely lehetővé teszi a grafikák feliratozását. A szöveg egyes karaktereit grafikus jelekké kell átalakítani a karaktergenerátorban tárolt információ alapján (lásd a 2.4 fejezetben ismertetett példa programot).
Tegyük fel, hogy az első képernyőpozícióban, és ennek megfelelően az 1024-es tárcímen éppen egy karakter található. Az 1024-es tárpozíció azonban a grafikus üzemmódban a karakternek megfelelő ponthalmaz színét tartalmazza A grafikus jel nagysága ugyanakkora, mint egy normális betűé (8 X 8). Ha tehát a képernyőt HI-RES módban használjuk, betű helyett csak egy színes négyzetet látunk. Ha a Commodore 64-es például ekkor egy hibaüzenetet próbálna kiírni, helyette a képernyőn természetesen olvashatatlan színes négyzetek jelennének meg.
Visszakapcsolva karakteres üzemmódba, a szöveg olvashatóvá válik és a grafika eltűnik.
Az Olvasó könnyebben megérti az elmondottakat, ha figyelembe veszi a következő tárfelosztást:

üzemmód	Képernyő-RAM	szín-RAM
Karakteres	1024 — 2023	55296 — 56295
Grafika	8192 — 16823	1024 — 2023

A táblázatból kitűnik, hogy a szöveg és a grafika miért nem keverhető egymással. Egy adott tárcím tartalma karakteres üzemmódban a karakterkódot, grafikus üzemmódban pedig a négyzet színkódját tartalmazza.
Hasonló gondot jelent a pontok generálása. Az egyes karakterekhez, illetőleg az egyes grafikus jelek pontjai, valamint a 8 X 8-as bitmátrixok közötti hozzárendelés elve a következő:
Vizsgáljuk meg, milyen információt tartalmaznak a grafikus képernyőtár egyes byte-jai. Az első byte a 8192-es tárcímen, az első 8 X 8 as egység legfelső sorára, a 8123-as byte ugyanezen négyzet második sorára stb. vonatkozik. A második 8 X 8-as négyzet pontjaira vonatkozó információt a 8200-as címtől kezdődő 8 byte tartalma adja:

8192 —>. . . . . 8200 —>. . . . . .
8193 —>. . . . . 8201 . . . . . .
tovább! stb.
8199 —>. . . . . .

A pontok megjelenítését POKE utasításokkal vezérelhetjük. Minthogy a POKE utasítás nem egy bit, hanem 1 byte tartalmát módosítja, egyidejűleg 8 pont megjelenítéséről kell gondoskodnunk. A pontok megjelenítéséhez először meg kell határoznunk azt a tárcímet, amely a kiválasztott pontokra vonatkozik, másrészt a 8 pont láthatóságának megfelelő bitmintát is ki kell választanunk.
Az első négyzet felső sorában álló pontok befolyásolására pl. a POKE 8129, utasítást használhatjuk, persze előzetesen kiszámítva a bitmintához tartozó decimális számértéket, amely az utasítás második argumentumát adja.
Például ha az adott sor baloldali és jobboldali pontját akarjuk megjeleníteni, a kívánt bitminta:

1 0 0 0 0 0 0 1

amit természetesen nem lehet POKE 8192, 10000001 formában megadni Emlékezzünk vissza az egyes bitek helyiértékére: a hetedik bit helyiértéke 2^7 azaz 128, a nulladik bit helyiértéke

pedig 2^0 azaz 1. Az utasítás helyes formában tehát POKE 8192,129. Az utasítás végrehajtása után (feltéve, hogy átkapcsoltunk grafikus üzemmódra) a legfelső sorban két pont láthatóvá válik.
Ha a grafika programozása során az Olvasó a fenti alapelveket figyelmen kívül hagyja, ugyancsak meglepő eredményekre juthat.
A következő oldalakon bemutatjuk azt a már korábban említett A HI-RES GRAFIKA SEGÉDESZKÖZE c. programot, amely a grafika programozását lényegesen megkönnyíti.
Végezetül egy javaslat: aki egy nagyon kényelmes grafikus programcsomagot szeretne vásárolni, annak feltétlenül ajánljuk a DATA BECKER cég által kifejlesztett SUPERGRAPHIK 64 assembler nyelven írt szoftvertermékét. A SUPERGRAPHIK 64 rendkívül gyors, nem fogjai el BASIC tárterületet, minden olyan utasítást tartalmaz, amit a nagyobb gépeken a SUPERGRAPHIK programcsomagból megismerhettünk, sőt néhány egészen sajátos utasítással újabb bizonyítékát adja annak, hogy a C 64-es képes versenyre kelni nagyobb testvéreivel. Míg a nagyobb Commodore gépeken a finomfelbontású lehetőségeket a hardver eszközök teremtik meg, addig a C 64-esen hasonló feladatokat egy programmal meg lehet oldani.
Az alábbiakban ismertetjük a SUPERGRAPHIK 64 legfontosabb utasításait:
A SUPERGRAPHIK 64 olyan BASIC bővítés a Commodore 64-hez, amely a finomfelbontású színes grafikák programozását támogatja. Utasításkészlete:

PLOT x, y	– egy pontot megjelenít a képernyő x, y koordinátájú pontjában
PLOT x1, y1, TO x2, y2	– egy egyenes szakaszt rajzol az x1, y1 pontból az x2, y2 pontba
PLOT TO x2, y2	– egyenes szakaszt rajzol a kurzor pozíciójától az x2, y2 pontba
CIRCLE	– kör, ellipszis, ív
PAINT változó FROM x, y	– zárt terület befestése a változóban megadott színkóddal
PAINT változó	– festés a kurzorpozíciótól kezdve
FRAME d, x1, y1, TO x2, y2	– d szélességű keret rajzolása
FILL x1, y1 TO x1, y2	– mezőmegadás
TEXT szöveg, x, y, m	– szöveg a grafikában
GMODE	– grafikus oldal és mód
GCLEAR	– grafika törlése
GMOVE	– grafika eltolása
INVERS	– grafika invertálása
ROT	– festett figura elforgatása
SIZE	– festett figura nagyítása
FCOL	– a keret színe
BCOL	– a háttér színe
SCOL	– a karakter színe
PCOL	– a pont színe
GSAVE	– a grafika tárolása
GLOAD	– a grafika betöltése
HCOPY	– nyomtatás
SREAD	– 63 byte olvasása
SDEFINE	– definiálás
SMODE	– sprite-mód
SSET (TO)	– a sprite mozgatása
IF # par THEN v, d, r, l, cn, c, cc	– elágazás
IRETURN	– a megszakítás vége
VOLUME	– hangerő
SOUND	– hang megszólaltatása
FILTER	– szűrő mód
TUNE	– hangszín

A programot szalagon vagy lemezen tárolhatjuk, közvetlenül betölthetjük és futtathatjuk. Érdemes feltárni a C 64-es és a SUPERGRAPHIK 64 együttműködéséből megszülető grafikus csodavilágot!

Az alábbiakban bemutatjuk a GRAFIK AID (A HI-RES GRAFIKA SEGÉDESZKÖZE) nevű program assemblerlistáját:

P9.ASS

```
000D                    110 CR      =  13       ; RETURN ASCII ERTEKE
001B                    120 ESC     =  27       ; ESCAPE
0014                    130 XCOORD  = $14
0097                    140 FLAG    = $97       ; PONT BEALLITASA/TORLESE
0097                    150 MASKE   = FLAG      ; HARDCOPY-MASZK
00B9                    160 SA      = $B9       ; MASODLAGOS CIM
00FD                    170 TMP     = $FD
00FD                    180 ADR     = TMP
00FD                    190 AV      = TMP
0014                    200 CODE    = XCOORD
0015                    210 SPALTE  = CODE + 1
8400                    220 COLLO   = $400      ; HI-RES SZINTAR KEZDETE
8800                    230 COLHI   = $800      ; HI-RES SZINTAR VEGE
2000                    240 GRALO   = $2000     ; HI-RES BIT TERKEP KEZDETE
4000                    250 GRAHI   = $4000     ; HI-RES BIT TERKEP VEGE
B7EB                    260 GETCOR  = $B7EB     ; BETOLTI AZ X ES AZ Y KOORDINATAKAT
AEFD                    270 CHKCOM  = $AEFD     ; A VESSZO ELLENORZESE
B79E                    280 GETBYT  = $B79E     ; EGY BYTE BETOLTESE
D000                    290 VIDEO   = $D000     ; VIDEOVEZERLO
E1D4                    300 GETPAR  = $E1D4     ; BETOLTI A FILENEVET ES AZ EGYSEGSZAMOT
E544                    310 CLRSCR  = $E544     ; TORLI A KEPERMYOT
FFC9                    320 CHKOUT  = $FFC9     ; BEALLITJA AZ OUTPUT EGYSEGET
FFCC                    330 CLRCH   = $FFCC     ; CSATORNA ZARASA
FFD2                    335 PRINT   = $FFD2     ; OUTPUT RUTIN
FFD5                    340 LOAD    = $FFD5     ; LOAD   RUTIN
FFD8                    350 SAVE    = $FFD8     ; SAVE   RUTIN
1000                    360         ;
C000                    370         *= $C000    ; UGRASTABLAZAT
C000 4C 1E C0           380         JMP INIT    ; GRAFIKA BEKAPCSOLASA
C003 4C 3C C0           390         JMP CLEAR   ; GRAFIKA TORLESE
C006 4C 51 C0           400         JMP COLOR   ; SZIN BEALLITASA
C009 4C 6D C0           410         JMP REVERS  ; GRAFIKA INVERTALASA
C00C 4C 89 C0           420         JMP SET     ; PONT BEIRASA
C00F 4C 86 C0           430         JMP RESET   ; PONT TORLESE
C012 4C 52 C1           440         JMP GLOAD   ; GRAFIKA BETOLTESE
C015 4C 39 C1           450         JMP GSAVE   ; GRAFIKA TAROLASA
C018 4C 75 C1           460         JMP HARD    ; HARDCOPY
C01B 4C 61 C1           470         JMP GOFF    ; GRAFIKA KIKAPCSOLASA
C01E                    480         ;
C01E 20 3C C0           490 INIT    JSR CLEAR     ; GRAFIKA TORLESE
C021 AD 11 D0           500         LDA VIDEO + 17      ; GRAFIKA BEKAPCSOLASA
C024 8D 70 C1           510         STA STORE1
C027 AD 18 D0           520         LDA VIDEO+24
C02A 8D 71 C1           530         STA STORE2
C02D A9 3B              540         LDA #27+32
C02F 8D 11 D0           550         STA VIDEO + 17
C032 A9 18              560         LDA #16+8
C034 8D 18 D0           570         STA VIDEO + 24
C037 A2 10              580         LDX #$10            ; A PONTOK SZINE FEHER
C039 4C 57 C0           590         JMP COL             ; A HATTER SZINE FEKETE
C03C                    600         ;
C03C A0 00              610 CLEAR   LDY #0        ; A GRAFIKA-TAR TORLESE
C03E A2 20              620         LDX #GRALO>
C040 84 FD              630         STY TMP
C042 86 FE              640         STX TMP+1
C044 98                 650         TYA
C045 EA                 655         NOP
```

```
C046 91 FD      660 CLR    STA (TMP),Y
C048 C8         670        INY
C049 D0 FB      680        BNE CLR
C04B E6 FE      690        INC TMP+1
C04D CA         700        DEX                    ; KOVETKEZO LAP
C04E D0 F6      710        BNE CLR
C050 60         720        RTS
C051            730        ;
C051 20 FD AE   740 COLOR  JSR CHKCOM   ; SZIN BEALLITASA
C054 20 9E B7   750        JSR GETBYT           ; SZINKOD TOLTESE
C057 A0 00      760 COL    LDY #0
C059 A9 04      770        LDA #COLLO>
C05B 84 FD      780        STY TMP
C05D 85 FE      790        STA TMP+1
C05F 8A         800        TXA                    ; SZINKOD
C060 A2 04      810        LDX #COLHI-COLLO>  ; LAPOK SZAMA
C062 91 FD      820 COL1   STA (TMP),Y
C064 C8         830        INY
C065 D0 FB      840        BNE COL1
C067 E6 FE      850        INC TMP+1
C069 CA         860        DEX                    ; KOVETKEZO LAP
C06A D0 F6      870        BNE COL1
C06C 60         880        RTS
C06D            890        ;
C06D A0 00      900 REVERS LDY #0       ; GRAFIKA INVERTALASA
C06F A9 20      910        LDA #GRALO>
C071 84 FD      920        STY TMP
C073 85 FE      930        STA TMP+1
C075 A2 20      940        LDX #GRAHI-GRALO>  ; KOVETKEZO LAP
C077 B1 FD      950 REV1   LDA (TMP),Y
C079 49 FF      960        EOR #$FF             ; AZ OSSZES BIT KONVERTALASA
C07B 91 FD      970        STA (TMP),Y
C07D C8         980        INY
C07E D0 F7      990        BNE REV1
C080 E6 FE     1000        INC TMP+1
C082 CA        1010        DEX                  ; KOVETKEZO LAP
C083 D0 F2     1020        BNE REV1
C085 60        1030 ILL    RTS             ; HIBAS KOORDINATAK BEIRASNAL
C086           1040        ;
C086 A9 80     1050 RESET  LDA #$80  ; PONT TORLESE
C088 2C        1060        .BYTE $2C
C089 A9 00     1070 SET    LDA #0       ; PONT BEIRASA
C08B 85 97     1080        STA FLAG
C08D 20 FD AE  1090        JSR CHKCOM       ; VESSZO
C090 20 EB B7  1100        JSR GETCOR       ; XCOORD-BAN AZ X, X-BEN AZ Y KOORDI
NATA
C093 E0 C8     1110        CPX #200
C095 B0 EE     1120        BCS ILL          ; Y KOORDINATA >199-NEL - VIZSGALAT
C097 A5 15     1130        LDA XCOORD+1
C099 C9 01     1140        CMP #320>
C09B 90 08     1150        BCC OK
C09D D0 E6     1160        BNE ILL
C09F A5 14     1170        LDA XCOORD
C0A1 C9 01     1180        CMP #320>        ; Y KOORDINATA >320-NAL - VIZSGALAT
C0A3 B0 E0     1190        BCS ILL
C0A5 8A        1200 OK     TXA              ; Y KOORDINATA AKKU-BAN
C0A6 4A        1210        LSR A
C0A7 4A        1212        LSR A
C0A8 4A        1214        LSR A
C0A9 0A        1216        ASL A            ; 8-SZOR
C0AA A8        1220        TAY
C0AB           1230        ; OFFY = 320 * INT(Y/8) + (Y AND 7)
C0AB B9 FF C0  1240        LDA MULT,Y
C0AE 8D 73 C1  1250        STA OFFY
C0B1 B9 00 C1  1260        LDA MULT+1,Y     ; 320-SZOR
C0B4 8D 74 C1  1270        STA OFFY+1
```



```
C0B7 8A        1280        TXA              ; Y-KOORDINATA
C0B8 29 07     1290        AND #7
C0BA 18        1300        CLC
C0BB 6D 73 C1  1310        ADC OFFY
C0BE 8D 73 C1  1320        STA OFFY
C0C1           1330        ; OFFX = 8* INT(X/8)
C0C1 A5 14     1340        LDA XCOORD
C0C3 29 F8     1350        AND #%11111000
C0C5 8D 72 C1  1360        STA OFFX
C0C8 18        1370        CLC              ; AV=GRALO+OFFY+OFFX
C0C9 A9 00     1380        LDA #GRALO<
C0CB 6D 73 C1  1390        ADC OFFY
C0CE 85 FD     1400        STA AV
C0D0 A9 20     1410        LDA #GRALO>
C0D2 6D 74 C1  1420        ADC OFFY+1
C0D5 85 FE     1430        STA AV+1
C0D7 18        1440        CLC
C0D8 A5 FD     1450        LDA AV
C0DA 6D 72 C1  1460        ADC OFFX
C0DD 85 FD     1470        STA AV
C0DF A5 FE     1480        LDA AV+1
C0E1 65 15     1490        ADC XCOORD+1
C0E3 85 FE     1500        STA AV+1
C0E5           1510        ; MA  = 2^((7-X)AND7)
C0E5 A5 14     1520        LDA XCOORD
C0E7 29 07     1530        AND #7
C0E9 49 07     1540        EOR #7
C0EB AA        1550        TAX
C0EC BD 31 C1  1560        LDA GRBIT,X      ; TABLAZATI ERTEK OLVASASA
C0EF A0 00     1570        LDY #0
C0F1 24 97     1580        BIT FLAG
C0F3 10 05     1590        BPL SET1
C0F5 49 FF     1600        EOR #$FF
C0F7 31 FD     1610        AND (AV),Y       ; PONT TORLESE
C0F9 2C        1620        .BYTE $2C
C0FA 11 FD     1630 SET1   ORA (AV),Y ; PONT BEIRASA
C0FC 91 FD     1640        STA (AV),Y
C0FE 60        1650        RTS
C0FF           1655        ;
C0FF           1660        ; SZORZOTABLA N*320, N=0-24
C0FF           1670        ;
C0FF           1675 MULT   = *
C0FF 00 00 40  1680        .BYTE 00,00,$40,01,$80,02,$C0,03
C107 00 05 40  1685        .BYTE 00,05,$40,06,$80,07,$C0,08
C10F 00 0A 40  1690        .BYTE 00,10,$40,11,$80,12,$C0,13
C117 00 0F 40  1695        .BYTE 00,15,$40,16,$80,17,$C0,18
C11F 00 14 40  1700        .BYTE 00,20,$40,21,$80,22,$C0,23
C127 00 19 40  1705        .BYTE 00,25,$40,26,$80,27,$C0,28
C12F 00 1E     1710        .BYTE 00,30
C131           1715        ;
C131 01 02 04  1720 GRBIT  .BYTE 1,2,4,8,$10,$20,$40,$80 ; 2 HATVANYAI
C139           1730        ;
C139 20 FD AE  1740 GSAVE  JSR CHKCOM; GRAFIKA TAROLASA
C13C 20 D4 E1  1750        JSR GETPAR       ; FILENEV ES KESZULEKCIM
C13F A2 00     1760        LDX #GRAHI<
C141 A0 40     1770        LDY #GRAHI>
C143 A9 00     1780        LDA #GRALO<
C145 85 FD     1790        STA TMP
C147 A9 20     1800        LDA #GRALO>
C149 85 FE     1810        STA TMP+1
C14B A9 FD     1820        LDA #TMP
C14D 85 B9     1830        STA SA           ; ABSZOLUT CIMZESU TAROLAS
C14F 4C D8 FF  1840        JMP SAVE
C152           1850        ;
C152 20 FD AE  1860 GLOAD  JSR CHKCOM; GRAFIKA TOLTESE
C155 20 D4 E1  1865        JSR GETPAR       ; FILENEV ES KESZULEKCIM
```

```
C158 A9 01      1870        LDA #1          ; ABSZOLUT CIMZESU TOLTES
C15A 85 B9      1880        STA SA
C15C A9 00      1890        LDA #0          ; LOAD FLAG
C15E 4C D5 FF   1900        JMP LOAD
C161            1910        ;
C161 AD 70 C1   1920 GOFF   LDA STORE1 ; GRAFIKA BE
C164 8D 11 D0   1930        STA VIDEO+17
C167 AD 71 C1   1940        LDA STORE2
C16A 8D 18 D0   1950        STA VIDEO+24
C16D 4C 44 E5   1960        JMP CLRSCR      ; KEPERNYO TORLESE
C170            1970        ;
C171            1980 STORE1 *= *+1
C172            1990 STORE2 *= *+1
C173            2000 OFFX   *= *+1
C175            2010 OFFY   *= *+2
C175            2020        ;
C175            2030        ; HARDCOPY
C175            2040        ; EPSON FX 80-HOZ, VC-INTERFACE-VEL
C175            2050        ;
C175 20 FD AE   2060 HARD   JSR CHKCOM ; VESSZO
C178 20 9E B7   2070        JSR GETBYT      ; LOGIKAI FILE-SZAM
C17B 20 C9 FF   2080        JSR CHKOUT      ; KIIRAS A NYOMTATORA
C17E A9 00      2090        LDA #GRALO
C180 A0 20      2100        LDY #GRALO
C182 85 FD      2110        STA ADR         ; A GRAFIKA CIME
C184 84 FE      2120        STY ADR+1
C186 A2 19      2130        LDX #25         ; A SOROK SZAMA
C188 A0 07      2140        LDY #7
C18A 2C         2150        .BYTE $2C
C18B A0 05      2160 ZEILEN LDY #5
C18D B9 D6 C1   2170 GMD    LDA GMOD,Y
C190 20 D2 FF   2180        JSR PRINT       ; NYOMTATO ATKAPCSOLASA BIT-TERKEP M
ODRA
C193 88         2190        DEY
C194 10 F7      2200        BPL GMD
C196 A9 28      2210        LDA #40
C198 85 15      2220        STA SPALTE
C19A A9 80      2230 SPALT1 LDA #$80
C19C 85 97      2240        STA MASKE
C19E A9 00      2250 BYTES  LDA #0
C1A0 85 14      2260        STA CODE
C1A2 A0 07      2270        LDY #7
C1A4 B1 FD      2280 BITS   LDA (ADR),Y; A BITMINTA OSSZEALLITASA
C1A6 25 97      2290        AND MASKE
C1A8 F0 07      2300        BEQ TT2         ; A BIT ERTEKE = 0
C1AA A5 14      2310        LDA CODE
C1AC 19 DE C1   2320        ORA GBIT,Y      ; A BIT ERTEKE = 1
C1AF 85 14      2330        STA CODE
C1B1 88         2340 TT2    DEY
C1B2 10 F0      2360        BPL BITS
C1B4 A5 14      2370        LDA CODE
C1B6 20 D2 FF   2380        JSR PRINT       ; KOD A NYOMTATORA
C1B9 46 97      2390        LSR MASKE
C1BB 90 E1      2400        BCC BYTES
C1BD A5 FD      2410        LDA ADR
C1BF 69 07      2420        ADC #7          ; PLUS CARRY = 8
C1C1 85 FD      2430        STA ADR         ; A CIM NOVELESE
C1C3 90 02      2440        BCC TT3
C1C5 E6 FE      2450        INC ADR+1
C1C7 C6 15      2460 TT3    DEC SPALTE  ; KOVETKEZO OSZLOP
C1C9 D0 CF      2470        BNE SPALT1
C1CB CA         2480        DEX             ; KOVETKEZO SOR
C1CC D0 BD      2490        BNE ZEILEN
C1CE A9 0D      2500        LDA #CR
C1D0 20 D2 FF   2510        JSR PRINT
C1D3 4C CC FF   2520        JMP CLRCH       ; NYOMTATO CSATORNA ZARAS
```



```
C1D6 01 40      2530 GMOD   .BYTE 320>,320<  ; GRAFIKUS PONTOK SZAMA
C1D8 06 2A 1B   2540        .BYTE 6,42,ESC,CR       ; GRAFIKUS MOD
C1DC 31 1B      2560        .BYTE 49,ESC            ; 8 FONT SORONKENT
C1DE 80 40 20   2570 GBIT   .BYTE $80,$40,$20,$10,0,4,2,1 ; 2 HATVANYAI
C1E6            2600        .END
```

```
ADR    =00FD  AV     =00FD  BITS   =C1A4  BYTES =C19E  CHKCOM=AEFD  CHKOUT=FFC9
CLEAR  =C03C  CLR    =C046  CLRCH  =FFCC  CLRSCR=E544  CODE   =0014  COL    =C057
COL1   =C062  COLHI  =0800  COLLO  =0400  COLOR  =C051  CR     =000D  ESC    =001B
FLAG   =0097  GBIT   =C1DE  GETBYT=B79E  GETCOR=B7EB  GETPAR=E1D4  GLOAD  =C152
GMD    =C18D  GMOD   =C1D6  GOFF   =C161  GRAHI  =4000  GRALO  =2000  GRBIT  =C131
GSAVE  =C139  HARD   =C175  ILL    =C085  INIT   =C01E  LOAD   =FFD5  MASKE  =0097
MULT   =C0FF  OFFX   =C172  OFFY   =C173  OK     =C0A5  PRINT  =FFD2  RESET  =C086
REV1   =C077  REVERS=C06D  SA     =00D9  SAVE   =FFD8  SET    =C089  SET1   =C0FA
SPALT1=C19A  SPALTE=0015  STORE1=C170  STORE2=C171  TMP    =00FD  TT2    =C1B1
TT3    =C1C7  VIDEO  =D000  XCOORD=0014  ZEILEN=C18B
```

A BASIC betöltő program;

```
P10

100 FOR I=49152 TO 49637
110 READ X :POKE I,X : S=S+X : NEXT
120 DATA  76, 30,192, 76, 60,192, 76, 81,192, 76,109,192
130 DATA  76,137,192, 76,134,192, 76, 82,193, 76, 57,193
140 DATA  76,117,193, 76, 97,193, 32, 60,192,173, 17,208
150 DATA 141,112,193,173, 24,208,141,113,193,169, 59,141
160 DATA  17,208,169, 24,141, 24,208,162, 16, 76, 87,192
170 DATA 160, 0 ,162, 32,132,253,134,254,152,234,145,253
180 DATA 200,208,251,230,254,202,208,246, 96, 32,253,174
190 DATA  32,158,183,160,  0,169,  4,132,253,133,254,138
200 DATA 162,  4,145,253,200,208,251,230,254,202,208,246
210 DATA  96,160,  0,169, 32,132,253,133,254,162, 32,177
220 DATA 253, 73,255,145,253,200,208,247,230,254,202,208
230 DATA 242, 96,169,128, 44,169, 0 ,133,151, 32,253,174
240 DATA  32,235,183,224,200,176,238,165, 21,201,  1,144
250 DATA   8,208,230,165,20 ,201, 64,176,224,138, 74, 74
260 DATA  74, 10,168,185,255,192,141,115,193,185,  0,193
270 DATA 141,116,193,138, 41,  7, 24,109,115,193,141,115
280 DATA 193,165, 20, 41,248,141,114,193, 24,169,  0,109
290 DATA 115,193,133,253,169, 32,109,116,193,133,254, 24
300 DATA 165,253,109,114,193,133,253,165,254,101, 21,133
310 DATA 254,165, 20, 41,  7, 73,  7,170,189, 49,193,160
320 DATA   0, 36,151, 16,  5, 73,255, 49,253, 44, 17,253
330 DATA 145,253, 96,  0,  0, 64,  1,128,  2,192,  3,  0
340 DATA   5, 64,  6,128,  7,192,  8,  0, 10, 64, 11,128
350 DATA  12,192, 13,  0, 15, 64, 16,128, 17,192, 18,  0
360 DATA  20, 64, 21,128, 22,192, 23,  0, 25, 64, 26,128
370 DATA  27,192, 28,  0, 30,  1, 2 ,  4,  8, 16, 32, 64
380 DATA 128, 32,253,174, 32,212,225,162,  0,160, 64,169
390 DATA   0,133,253,169, 32,133,254,169,253,133,185, 76
400 DATA 216,255, 32,253,174, 32,212,225,169,1  ,133,185
410 DATA 169,  0, 76,213,255,173,112,193,141,17 ,208,173
420 DATA 113,193,141, 24,208, 76, 68,229,  0,  0,  0,  0
430 DATA   0, 32,253,174, 32,158,183, 32,201,255,169,  0
440 DATA 160, 32,133,253,132,254,162, 25,160,  7, 44,160
450 DATA   5,185,214,193, 32,210,255,136,16 ,247,169, 40
460 DATA 133, 21,169,128,133,151,169,  0,133,20 ,160,  7
470 DATA 177,253, 37,151,240,  7,165, 20, 25;222,193,133
480 DATA  20,136,208,240,165, 20, 32,210,255, 70,151,144
490 DATA 225,165,253,105,  7,133,253,144,  2,230,254,198
500 DATA  21,208,207,202,208,189,169, 13, 32,210,255, 76
510 DATA 204,255,  1, 64,  6, 42, 27, 13, 49, 27,128, 64
520 DATA  32, 16,  8,  4,  2,  1,
530 IF S<>60459 THEN PRINT"HIBA AZ ADATOKBAN":END
540 PRINT"RENDBEN   !"
```

A program használata nagyon egyszerű. A BASIC programból SYS utasítással hívhatjuk meg az egyes rutinokat, amelyben esetenként paramétereket is átadhatunk. A program elején célszerű a rutinok címeit változókban tárolni. A SYS utasításban a megfelelő ugrási címet tartalmazó változó után vesszővel elválasztva beírhatjuk a szükséges paramétereket.
Az alábbi példa egyes változóinak jelentése:

X — A grafikus pont vízszintes koordinátája, 0-tól 319-ig
Y — Függőleges koordináta 0-tól 199-ig. A (0,0) koordinátájú pont a bal felső sarokban van.
PF — A grafikus pont színeinek kódja, 0-tól 15-ig
HF — A háttér színe, 0-tól 15-ig
LF — A nyomtató logikai file-száma 1-től 255-ig.

```
P11

100 IN=12*4096 : REM SYS IN - GRAFIKA BEKAPCSOLASA
110 CL=IN+3    : REM SYS CL - GRAFIKA TORLESE
120 CO=IN+6    : REM SYS CO,PF*16+HF - SZIN BEALLITAS
125 RV=IN+9    : REM SYS RV - GRAFIKA INVERTALAS
130 SE=IN+12   : REM SYS SE,X,Y - PONT BEIRAS
140 RS=IN+15   : REM SYS RS,X,Y - PONT TORLES
150 GL=IN+18   : REM SYS GL,"NAME",1 ODER 8 - GRAFIKA BETOLTES
160 GS=IN+21   : REM SYS GS,"NAME",1 ODER 8 - GRAFIKA MENTESE
170 HD=IN+24   : REM SYS HD,LF  - HARDCOPY A NYOMTATORA
180 OF=IN+27   ,: REM SYS OF - GRAFIKA KIKAPCS
200 SYS IN : REM GRAFIKA BE
210 PF=1 : REM PONTSZIN= FEHER
220 HF=0 : REM HATTERSZIN = FEKETE
230 SYS CO,16*PF+HF : REM  SZIN BEALLITAS
240 REM X TENGELY RAJZOLASA
250 FOR I=0 TO 319 :SYS SE,X,100 : NEXT
260 REM Y TENGELY RAJZOLASA
270 FOR I=0 TO 199 :SYS SE,160,Y : NEXT
280 REM SINUS-GORBE RAJZOLASA
290 PI:X=0
300 FOR I=-PI TO PI STEP 2/319
310 SYS SE,X,100+99*SIN(I)
320 X=X+1:NEXT
330 SYS GS,"SINUS-GORBE",8:REM GRAFIKA MENTESE
340 OPEN 1,4,1 :REM NYOMTATO NYIT - EPSON MODUS
350 SYS HD,1 :REM HARDCOPY A NYOMTATORA
360 CLOSE1:REM NYOMTATOFILE ZAR
370 SYS OF :REM GRAFIKA KIKAPCS
```

3.7 A sprite — a Commodore 64-es varázsszava

Bevezetés

A finomfelbontású grafika mellett kétségkívül az ún. sprite-ok jelentik a Commodore 64-es adottságainak csúcsát. A sprite-ok önálló kis grafikák, amelyek egymástól függetlenül programozhatók. Használatukkor nem kell finomfelbontású grafikára átkapcsolnunk, mivel a Commodore 64-es a sprite-okat egészen sajátos módon kezeli.

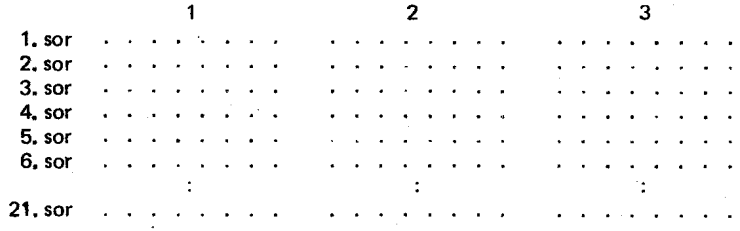
Lehetőségek

A Commodore 64-es nyolc sprite-ot (0-tól 7-ig számozva) képes egyszerre megjeleníteni a képernyőn. Ha a sprite-ot elhelyeztük a tárban és elláttuk egy sorszámmal, igényünk szerint ki-/bekapcsolhatjuk, azaz a képernyőn láthatóvá, illetve nem láthatóvá tehetjük. Az X, Y

koordináták megadásával a sprite-ot áthelyezhetjük a képernyő egyik oldaláról a másikra anélkül, hogy a korábbi pozícióról töröltük volna. Munka közben sem a képernyőtárat, sem a színtárat nem kell megváltoztatni. Mindez automatikusan megvalósul a 6569-es VIC vezérlésével.
A grafikus képek szerkesztésének további nagyszerű eszközei: a sprite-okat függőleges és/vagy vízszintes irányban megnagyíthatjuk; ütközést idézhetünk elő közöttük és közben a hátteret rögzíthetjük; meghatározhatjuk a grafikák prioritását, pl azt, hogy a sprite-ok a háttér előtt vagy mögött jelenjenek meg. A prioritás vezérlésével térhatású képeket is ábrázolhatunk. A Commodore 64-es tehát háromdimenziós grafikák előállítására képes, és ezzel az első olyan gép a személyi számítógépek sorában, amely viszonylag kis programozási ráfordítás árán egészen fantasztikus grafikus eredményeket szolgáltat.

Felépítés

A sprite-ok programozásához elengedhetetlenül fontos a bináris aritmetika és a 6569-es processzor regisztereinek ismerete (lásd az 1.1 és a 3.1.1 fejezetet). Ha ezeket a fejezeteket az Olvasó még nem nézte át, vagy tartalmukat nem értette meg egész pontosan, célszerű visszalapoznia és a homályos részeket mégegyszer gondosan áttanulmányoznia. Ez biztosíték arra nézve, hogy a következő fejezetek megértése nem okoz majd gondot.
A sprite-ok generálása során a legfontosabb feladat a 46 regiszterre hárul (lásd a 3.1.1 alfejezetet). Ezekkel a regiszterekkel vezérelhetjük ugyanis a sprite-ok mozgását, színét, és egyéb sajátos tulajdonságait Minden regiszter 8 bitből áll, amelyeket a programozó saját igényeinek megfelelően beállíthat vagy törölhet. A sprite-ok ábrázolásának legfontosabb lépése a sprite pontjainak meghatározása. Mint már tudjuk, a sprite egy olyan téglalap alakú ponthalmaz, amely vízszintesen 21, függőlegesen pedig 24 pontot tartalmaz. Miután azonban egy POKE utasítással maximálisan csak 8 pontot állíthatunk be vagy törölhetünk, egyetlen sorra vonatkozó információ tárolásához 3 byte-ra van szükség.
Végül is minden sprite három oszlopból építhető fel, amelyek mindegyike egy 3 X 21-es pontmátrixból áll. A programozásnál ügyelnünk kell arra, hogy a három oszlopot mindenkor egymás mellett tároljuk.
Az elmondottakat a következőképpen szemléltethetjük:



A sprite-ok megjelenítésének kétféle módja van:

1) Az egyszínű megjelenítés
Ebben az esetben a sprite minden pontjának pontosan egy bit felel meg. A pontok színét vagy a sprite-hoz tartozó regiszter határozza meg, vagy a háttér színe. Ha a ponthoz rendelt bit értéke 1, akkor a pont a sprite színében, egyébként a háttér színében jelenik meg.

2) A sokszínű (multicolor) megjelenítés
Ebben az esetben minden ponthoz egy bitpárt rendelünk. A két bit meghatározza azt a

regisztert, ahol a pont színét tároljuk. Igy egy sprite maximálisan háromféle színből tevődhet össze, hiszen bár két biten összesen négyféle információt tárolhatunk, de a 0-t a nem látható pontok meghatározására használjuk.

A következő fejezetben részletesen foglalkozunk a sprite-ok programozásával.

3.7.1 A SPRITE-OK PROGRAMOZÁSA

Bevezetés

A sprite-ok programozásának legfontosabb BASIC utasítása a POKE utasítás.
Először meg kell határozni a sprite látható pontjainak pozícióját, majd ki kell számítani az így kapott bitmintához tartozó decimális számot, amely a POKE utasítás második argumentuma lesz. Szemben a finomfelbontású grafika programozásával, itt nem kell állandóan változtatni a bitmintát, így a kapott értékeket elhelyezhetjük DATA sorokban, és egy FOR NEXT ciklus segítségével beolvashatjuk. Végül a decimális számokat POKE utasítással rendre a megfelelő tárcímekre töltve készen van a sprite. Nem szabad persze megfeledkezni arról, hogy a sprite nem látható pontjaihoz rendelt 0 értékeket is be kell írni a megfelelő tárcímekre. A programozás megértésének megkönnyítése érdekében elemezzük az alábbi DATA sorokat:

```
1000 DATA 000,000,000
1010 DATA 000,000,000
1020 DATA 000,000,000
1030 DATA 000,000,000
1040 DATA 000,000,000
1050 DATA 000,000,000
1060 DATA 000,000,000
1070 DATA 003,255,255
1080 DATA 000,002,000
1090 DATA 192,170,128
1100 DATA 194,150,080
1110 DATA 234,150,080
1120 DATA 194,170,168
1130 DATA 192,170,168
1140 DATA 000,032,128
1150 DATA 000,170,160
1160 DATA 000,000,000
1170 DATA 000,000,000
1180 DATA 000,000,000
1190 DATA 000,000,000
1200 DATA 000,000,000
```

A DATA sorok felépítése megfelel a sprite-ok 21 X 3-as szerkezetének. A számok mindegyike 1 és 255 közé esik, és egy-egy bitmintát reprezentál. Ezen a módon egy tetszőleges 21 X 24 pontból álló alakzat programozható.
Mielőtt elmélyednénk a sprite-ok programozásának részleteiben, érdemes elgondolkodni azon, hogy mi indokolja a kétféle ábrázolási mód, a finomfelbontású-, és sprite-os grafika létjogosultságát, milyen szempontok alapján dönthetünk egy-egy feladat megvalósítása során az egyik, illetve a másik mellett.

Alkalmazás

A sprite-ok előre megtervezett figurák, leglényegesebb tulajdonságuk a mozgathatóság. Alakjuk sohasem változik meg, legfeljebb a méretük. Jellegükből következően elsősorban a játékprogramokban és videotrükkök megvalósításában játszanak fontos szerepet.

A finomfelbontású képernyőn ábrázolt grafikák alakját általában nem lehet előre megtervezni. Gondoljunk arra, hogy egy függvény képe mindig a függvény egyenletében szereplő paraméterektől, vagy pl. egy választási eredményeket ábrázoló oszlopdiagram formája az egy-egy jelöltre beérkezett szavazatok gyakoriságától függ.
Jól érzékelhető, hogy az eltérő feladattípusok más grafikus megoldást kívánnak, és nagyon jó, hogy a C 64-es mindkét lehetőséget a rendelkezésünkre bocsájtja.
A sprite-okat reklámcélokra is kiválóan felhasználhatjuk.
Szerkeszthetünk pl. színes termékbemutatót, ami különösen hatásos lehet hangeffektusokkal párosítva.

A programozás alapgondolata

Térjünk vissza a sprite-ok programozására.
Elemezzük részleteiben, hogy mit is tartalmaznak a fenti DATA sorok?
Alakítsuk vissza bináris számokba:

	SOROZAT		
	1	2	3
1. sor	00000000	00000000	00000000
2. sor	00000000	00000000	00000000
3. sor	00000000	00000000	00000000
4. sor	00000000	00000000	00000000
5. sor	00000000	00000000	00000000
6. sor	00000000	00000000	00000000
7. sor	00000000	00000000	00000000
8. sor	00000000	11111111	11111111
9. sor	00000000	00000010	00000000
10. sor	11000010	10010110	10000000
11. sor	11000010	10010110	01010000
12. sor	11101010	10010110	01010000
13. sor	11000010	10101010	10101000
14. sor	11000000	10101010	10101000
15. sor	00000000	00100000	10000000
16. sor	00000000	10101010	10100000
17. sor	00000000	00000000	00000000
18. sor	00000000	00000000	00000000
19. sor	00000000	00000000	00000000
20. sor	00000000	00000000	00000000
21. sor	00000000	00000000	00000000

Lehet, hogy első látásra nem szembetűnő, de valójában a 0-kkal és 1-esekkel egy helikoptert akartunk ábrázolni.
A figura papíron persze nem egészen úgy mutat, mint ahogyan az a képernyőn majd megjelenik.

Bekapcsolás

A sprite alakját persze nem kell véglegesnek tekinteni, ha nem sikerült úgy, ahogyan terveztük, bármikor módosíthatjuk.
Megjelenítéshez a sprite-ot be kell kapcsolni. Bekapcsolás előtt a 21. regiszterbe betöltjük a sprite sorszámát. A regiszter minden bitje megfelel egy sprite-nak.

Sprite:	7	6	5	4	3	2	1	0
Bit:	b7	b6	b5	b4	b3	b2	b1	b0

A POKE 21,1 utasítás tehát a 0. sprite-ot kapcsolja be, a POKE 21,3 utasítás a 0. és az 1. sprite-ot, a POKE 21,255 utasítás pedig az összes sprite-ot aktivizálja.

A sprite-ot tartalmazó tárterület

A bekapcsoláshoz meg kell adni azt a tárterületet, ahol tároltuk a sprite-hoz rendelt bitmintát. A sprite-okat tartalmazó tárterületek kezdőcímét a képernyőtár legfelső nyolc tárcíme, nevezetesen a 2040-től 2047-ig terjedő tárcímek tartalmazzák. Minden sprite 63 byte-ot foglal el (21 sor, soronként 3 byte), és ehhez elválasztó byte-ként még egy 0 tartalmú byte-ot is csatolunk, így ez összesen 64 byte.
A 16 k-s képtár 256, egyenként 64 byte-os blokkra van felosztva. A kezdőcímek, vagy másképpen sprite-mutatók tartalma azt a blokkot adja meg a 256 blokkon belül, ahol a sprite kezdődik.
Tegyük fel, hogy a sprite a 832-es tárcímen kezdődik. Ekkor a sprite mutatója 13, (832/64 = 13) Ha történetesen ez a 0. sprite, akkor a sprite-mutató értékét a POKE 2040,13 utasítással határozhatjuk meg. A 13 azt jelenti hogy a VIC chip által címezhető 16 k-s terület 13. 64 byte-os egysége határozza meg a sprite-ot.
A fentiek alapján világos, hogy minden sprite egy 64 byte-os blokk elején kell, hogy kezdődjön. Ha a képernyőtár alaphelyzetban van (azaz a $400-as címen kezdődik), akkor a sprite-mutatók értékei a következők lehetnek:

Cím	Mutató
704	11
768	12
832	13
896	14
960	15

Minthogy azonban a fenti tárterület csak négy sprite ábrázolásához elegendő, további sprite-okat csak úgy tudunk bekapcsolni, ha a BASIC kezdetet eltoljuk, és így további tárterületeket szabadítunk fel:

```
POKE 44,10        (BASIC kezdet a $0A00 címre)
POKE 10*256,0     (Az első BASIC byte értéke 0)
NEW               (A mutató visszaállítása)
```

A következő táblázat tartalmazza a sprite-mutatókhoz tartozó sprite-okat.

Mutató	2040	2041	2042	2043	2044	2045	2046	2047
Sprite	0	1	2	3	4	5	6	7

Ha tehát a 2040-es tárcímre 13-at írunk, ez azt jelenti hogy a 0. sprite a 832-es tárcímen kezdődik. Ha a 2041-es címre is 13-at írunk akkor a 0. és az 1. sprite alapja azonos lesz, hiszen mindkettőt azonos tárterület definiálja.
Természetesen ebben az esetben is lehetnek a 0. és az 1. sprite-ra vonatkozó egyéb információk (pl. helyzet, szín) eltérőek, és ez érthető módon megkönnyíti a programozási munkát.
Ezzel be is fejeztük a sprite megjelenítéséhez szükséges programozási teendőket. Minden egyebet elvégez helyettünk a VIC. Az egyetlen hátralevő feladat a sprite adatok tényleges betöltése a 832-es tárcímtől kezdve.
A betöltést ciklusutasítással végezzük el:


```
FOR I=0 TO 62 :  REM A SPRITE 63 BYTE-JA
READ X        :  REM A BYTE BEOLVASASA
POKE 832+I,X  :  REM A BYTE BEIRASA A TARBA
NEXT I        :  REM A CIKLUS LEZARASA
```

A továbbiakban minden a sprite-ra vonatkozó információ módosításához elegendő egyetlen POKE utasítás.

A sprite pozicionálása

A sprite képernyőn elfoglalt helyzetét két regiszterrel vezérelhetjük. A 0-s és 1-es regiszterek tartalma határozza meg a 0. sprite X és Y koordinátáját, a következő két regiszter (2-es és 3-as) tartalma az 1-es sprite koordinátáit és így tovább. A következő példákban a V változó értéke 53248. A grafikus programokat célszerű mindig a következő utasítással kezdeni:

```
V=53248   :  REM A VIC KEZDOCIME
```

A sprite pozícionálásához a következő két POKE utsításra van szükség:

```
POKE V+0, OSZL :  REM 0.SPRITE -X
POKE V+1, SOR  :  REM 0.SPRITE -Y
```

A DATA sorokkal meghatározott helikoptert az alábbi POKE utasítás a képernyő közepére helyezi

```
POKE V+0,160   :  POKE V+1,120
```

A sprite mozgatása

Ha a pozíciót folyamatosan változtatjuk, a sprite mozogni látszik a képernyőn. Az X koordinátát rendre 1-gyel változtatva a helikopter átrepül a képmezőn.

```
FOR  I=159 TO 100 STEP -1
     POKE V+0,I
NEXT I
```

Mivel a ciklusban az X koordinátát folyamatosan csökkentjük, a helikopter jobbról balra repül. A mozgás olyan gyorsan lezajlik, hogy szinte nem is érzékeljük.
Egy üres ciklus lelassítja a végrehajtást és a helikopter mozgását láthatóvá, méltóságteljesebbé teszi.
Az X irányú pozícionálás – ez már biztosan eszébe jutott az Olvasónak is – egy regiszterrel nem oldható meg tökéletesen. A vízszintes irányú koordináta maximális értéke ugyanis 320, az egy regiszterben tárolható legnagyobb számérték pedig 255.
A megoldást egy további regiszter igénybevétele jelenti.
A 16-os regiszter nyolc bitjét hozzárendeljük a nyolc sprite-hoz.
A bitek 1 értéke jelzi, hogy az adott sprite X koordinátája nagyobb, mint 255.
A következő POKE utasítással pl. a VIC tudomására hozzuk, hogy a helikopter pozíciójának X koordinátája nagyobb, mint 255:

```
POKE V+16,1
```

A POKE utasítás hatására a VIC az X koordinátát automatikusan megnöveli 255-tel. A visszaállítás is igen egyszerű:

```
POKE V+16,0
```

Mindössze ennyit kell tudni ahhoz, hogy BASIC programmal a sprite-okat tetszőleges irányba, tetszőleges sebességgel mozgathassuk a képernyőn.

Színezés

A sprite-ok színét pozíciójukhoz hasonlóan egyszerű POKE utasításokkal szabályozhatjuk. A VIC minden sprite-hoz hozzárendel egy színregisztert;

Regiszter:	39	40	41	42	43	44	45	46
Sprite:	0	1	2	3	4	5	6	7

Színek:

0	fekete
1	fehér
2	vörös
3	türkiz
4	lila
5	zöld
6	kék
7	sárga
8	narancsszín
9	barna
10	rózsaszín
11	1. szürke
12	2. szürke
13	világoszöld
14	világoskék
15	3. szürke

A helikopter a POKE V + 39,14 utasítás végrehajtása után világoskék színben pompázik.

Nagyítás

Amint azt már korábban említettük, a sprite-okat vízszintes és/vagy függőleges irányban kétszeresére nagyíthatjuk. A nagyítást ismét két regiszter tartalma jelzi, az egyik X- a másik pedig Y irányú nagyítást vezérel. A regiszterek bitjeihez ismét hozzárendeltük az egyes sprite-okat. Ha egy sprite-hoz tartozó bit értéke 1, a sprite felnagyítva látható a képernyőn (a regisztertől függően vízszintes vagy függőleges irányban).
A két regiszter felépítése a következő

Bit:	b7	b6	b5	b4	b3	b2	b1	b0
Sprite:	7	6	5	4	3	2	1	0

Az alábbi utasításokkal az első sprite-ot X- és Y irányban egyaránt felnagyítjuk (az első sprite alatt a 0. sprite értendő):

```
POKE V+23,1:    REM 0.SPRITE NAGYITASA Y-IRANYBAN.
POKE V+29,1:    REM 0.SPRITE NAGYITASA X-IRANYBAN.
```


A 0. és az 1. sprite-okat egyidejűleg a POKE V + 23,3 utasítással nagyíthatjuk fel.
A nagyítás mindkét irányban csak kétszeres lehet.
Ha például sprite-ok felhasználásával villanyújságot készítünk, a betűk mérete maximum négyszeresére növelhető.

A háttér

A képet térhatásúvá teszi az alakzatok megjelenési elsőbbségének szabályozása. Meghatározhatjuk pl., hogy a sprite a háttér előtt vagy mögött helyezkedjék-e el. Nézzük meg mi történik a helikopterrel a következő utasítás hatására:

```
POKE V+27,1
```

Vigyük a kurzort a sprite-ra és írjunk be néhány karaktert. Látszólag a sprite eltűnik a képernyőről, valójában azonban csak átkerül egy másik képsíkba.
Erről a következő utasítás végrehajtása után győződhetünk meg.

```
POKE V+27,0
```

A sprite most ismét a szöveg elé kerül. A prioritást vezérlő regiszter felépítése a következő:

Bit:	b7	b6	b5	b4	b3	b2	b0
Prioritás:	7	6	5	4	3	2	0

A nagyításhoz hasonlóan, most is az egyes bitek értéke határozza meg a megfelelő sprite prioritását. A bit nulla értéke azt jelenti, hogy a sprite a háttér előtt áll. A bitek különböző beállításával és törlésével több sprite-ot egymás fölé helyezhetünk úgy, hogy közben a háttér és előtér — jó színválasztás esetén — élesen megkülönböztethető.

A sprite-ok ütköztetése

A térbeli ábrázolásmódnak nem egyedüli eszköze a sprite és a háttér prioritásának programozhatósága. A sprite-okat egymással ütköztethetjük is. A VIC egy regiszterének értéke mindig jelzi, hogy az adott pillanatban mely sprite-ok ütköznek egymással. A regiszter szerkezete hasonló az előzőek szerkezetéhez. Tartalmát leolvashatjuk és felülírhatjuk, ezzel egyrészt értesülünk arról, hogy mely sprite-ok ütköznek, másrészt előidézhetjük az ütközést. Pl. ha a

```
PRINT PEEK (V+30) vagy KO=PEEK (V+30)
```

utasítással leolvassuk a regiszter tartalmát és azt találjuk, hogy értéke 3, akkor a képernyőn a 0. és 1. sprite ütközik. Leolvasás után a regiszterbe 0-t kell tölteni:

```
POKE V+30,0
```

Egyébként ugyanis a következőkben az ütközésről nem kapnánk valódi információt.

A sprite és a háttér ütközése

A különböző sprite-ok ütközésének regisztrálásához hasonlóan, a rendszer a sprite-ok és a háttér közötti ütközésre is reagálhat. Erre szolgál a 31. regiszter, amelynek kezelése hasonló a 30.

regiszteréhez. Az egyetlen eltérés a PEEK-lekérdezés eredményében jelentkezik. Az eredmény csak arra nézve ad felvilágosítást, hogy mely sprite(-ok) ütközött (ütköztek) egy karakterrel, az azonban nem derül ki, hogy az ütközés tárgya melyik karakter volt és az ütközés melyik képernyőpozíción következett be. Ha ezekre az információkra is szükség van, le kell kérdeznünk a többi regisztert és a tárpozíciót.
Az utasítás hasonló az előzőhöz:

PRINT PEEK (V+31) vagy KO=PEEK (V+31)

Leolvasás után a 31-es regiszter tartalmát is vissza kell állítani.

A képernyő eltolása

Két további hasznos regiszter a 17-es és a 22-es regiszter. Tartalmuk változtatásával az egész képernyőt lépésenként eltolhatjuk. Az eltolás mértéke minden lehetséges irányban (felfelé, lefelé, jobbra, balra) maximum 8—8 lépésre terjedhet. A 17. regiszter az Y irányú eltolásért a 22. regiszter pedig az X irányú eltolásért felelős. A regiszterek módosítása közben az alábbi két dologra kell ügyelnünk:
1) A regiszterek 3. bitjének értéke mindig 1 kell, hogy legyen, mert csak így kerülhet sor szabályos eltolásra.
2) A módosítás során a regiszterekbe nem szabad egyszerűen új értéket beírni, mert ezzel a többi bit is megváltozhat.
A 3. bit beállítása a 22. regiszterben:

POKE V+22, PEEK (V+22) OR 8

Akárt formával a képernyő egész tartalma a megadott értékkel eltolható.

Többszínű sprite-ok

Az a lehetőség, hogy többszínű sprite-okkal is dolgozhatunk, a sprite grafika legnagyszerűbb adottsága. A sprite három különböző színnel festhető ki, ugyanakkor természetesen kisebb a felbontása, mivel pontok helyett pontpárokkal dolgozunk. A 8 X 8-as mátrix helyett egy 4 X 8 as mátrix az ábrázolás alapegysége. A pontpárhoz ugyan most is két bitet rendelünk, de a két bit tartalma a választott színen kívül arról is kell, hogy informáljon, hogy a pont látható vagy sem.
Tudjuk, hogy 2 bittel összesen 4 információt közölhetünk: 00, 01, 10 és 11. Esetünkben az értékek információtartalma a következő:

00 — A pont színe azonos a háttérszínnel (a pont nem látható)
01 — A szín kódját a 37. regiszter szolgáltatja (a pont a megfelelő színben látható)
10 — A szín a sprite színregiszterből származik (a színt a 39-től 46-ig terjedő regiszterek valamelyike szolgáltatja)
11 — A színt a 38. regiszter szolgáltatja (a pont ekkor a megfelelő színben látható).

A fentiek alapján bizonyosan érthető, hogy a sprite egy saját színből, és az összes sprite-ra nézve közös két színből tevődhet össze. A nem látható pont színe azonos a háttér színével. Az Olvasó eddig talán kicsit elégedetlen volt a helikopter külső megjelenésével. Ennek igen egyszerű magyarázata van. Eredetileg többszínű helikoptert terveztünk, azonban a többszínű sprite kevesebb pontból áll mint az egyszínű, így az egyszínű megjelenítés eredményeként természetesen nem kaphattunk értelmes képet. A fejezet végén közöljük azt a programot, amely

felrajzolja a képernyőre a többszínű helikoptert. Javasoljuk, hogy a példa alapján az Olvasó készítsen önálló programokat, így lehet ugyanis a leggyorsabban elsajátítani a sprite-programozás technikáját.

```
F12

10 REM SPRITE BEMUTATO - HELIKOPTER
20 V=53248 :REM VIDEO KEZDOCIM
30 POKE V+32,15 :POKE V+33,14:REM HATTERSZINEK
40 PRINT"(CTRL-7)":REM NYOMJA 'CTRL'-T ES '7'-T ROVID IDEIG
50 POKE V+21, 3: REM SPRITE 0 ES 1 BEKAPCSOLAS
60 POKE V+28, 3: REM SPRITE 0 ES 1 MULTICOLOR
70 POKE V+39, 6: REM SPRITE 0 SZINE KEK
80 POKE V+40, 2: REM SPRITE 1 SZINE PIROS
90 POKE V+37,14: REM MULTICOLOR-SZIN 1 - VILAGOS KEK
100 POKE V+38, 0: REM MULTICOLOR-SZIN 2 - FEKETE
110 POKE 2040,13  :REM SPRITE 0 TARCIM   832-TOL 895-IG
120 POKE 2041,13  :REM SPRITE 1 TARCIM   832-TOL 895-IG
130 FOR I=0 TO 62: REM ADATOK BEOLVASASA
140 : READ X: REM SPRITE-PONTOK BEOLVASASA
150 : POKE 832+I,X: REM SPRITE-PONTOK TAROLASA
160 NEXT I: REM A CIKLUS VEGE
170 POKE V+0,24:POKE V+1,50 :REM SPRITE 0 KOORDINATAI
180 POKE V+2,60:POKE V+3,50 :REM SPRITE 1 KOORDINATAI
190 END
1000 DATA 000,000,000
1010 DATA 000,000,000
1020 DATA 000,000,000
1030 DATA 000,000,000
1040 DATA 000,000,000
1050 DATA 000,000,000
1060 DATA 000,000,000
1070 DATA 003,255,255
1080 DATA 000,002,000
1090 DATA 192,170,128
1100 DATA 194,150,080
1110 DATA 234,150,080
1120 DATA 194,170,168
1130 DATA 192,170,168
1140 DATA 000,032,128
1150 DATA 000,170,160
1160 DATA 000,000,000
1170 DATA 000,000,000
1180 DATA 000,000,000
1190 DATA 000,000,000
1200 DATA 000,000,000
```

Ha valóban sikerült az Olvasót önálló programok készítésére ösztönözni, szeretnénk a munkáját még néhány javaslattal támogatni. Az elkészült program minőségét mindig az előzetes tervezés alapossága szabja meg. A sprite-ok programozására ez különösen igaz, hiszen a grafikus alakzatokat lehetetlenség tervezés nélkül helyesen programozni.
A könyv végén a mellékletek között találunk egy tervezőlapot, amellyel a szerzők a sprite-programozás előkészítő munkáit igyekeznek megkönnyíteni. Célszerű a lapot kimásolni és sokszorosítani.
Ha valaki komolyan elmélyed a sprite-ok programozásában, be fogja látni, hogy enélkül a munka szinte lehetetlen.

A tervezés

Hogyan használjuk a tervezőlapot?
Először készítsük el a figura durva vázlatát egy normál papírlapon. Döntsük el, hogy a figura

egy- vagy többszínű legyen. Ezt követően a sprite típusának megfelelően, a vázlat alapján töltsük ki a tervezőlapot.

Többszínű sprite

Ha színes sprite mellett döntöttünk, ne feledkezzünk meg arról, hogy a lap két-két pontja a képernyőn egy pontként jelenik majd meg.
A tervezéssel párhuzamosan azt is el kell döntenünk, hogy melyik regisztert használjuk a színek tárolására.
Legcélszerűbb, ha a lapon nem töltünk ki minden négyzetet, hanem egy-egy négyzet-párba beírjuk azt a bitkombinációt ami a pont színét megszabja..
A lap kitöltése után ki kell számítani a bitkombinációk decimális értékét. A lap felső során feltüntettük a bináris helyiértékeket, így nem kell mást tenni, mint ezek közül azokat összeadni, amelyek alatt a sorban 1-es áll.
Eredményként megkapjuk minden sorhoz azt a három decimális számot, amelyeket a DATA sorokba el kell helyezni.

Az egyszínű sprite-ok

Az egyszínű sprite-ok tervezése sokkal egyszerűbb. Itt ugyanis a lap négyzetei és a képernyő pontjai között egyértelmű a megfeleltetés. Ebben az esetben, ha egy ceruzával minden négyzetet kitöltünk, a papíron kapott figurának hasonlítania kell a jövendőbeli sprite-ra. Az eljárás a továbbiakban azonos az előzővel; kiszámítjuk a byte-ok decimális értékét és elhelyezzük a DATA sorokban.

4. FEJEZET
INPUT-OUTPUT VEZÉRLÉS – A CIA 6526-OS CHIP

4.1. Általános tudnivalók

A 6526-os CIA (Complex Interface Adapter) chip a 65xx-család egy új periféria-modulja. Legfontosabb adottságai:
- 16 külön-külön programozható I/O vonal
- 8, illetve 16 bites handshake bevitelnél és kihozatalnál egyaránt
- 2 egymástól független, kaszkádolható 16 bites timer
- 24 órás (AM/PM) óra, programozható riasztóidővel
- 8 bites léptetőregiszter soros bevitelhez/kihozatalhoz

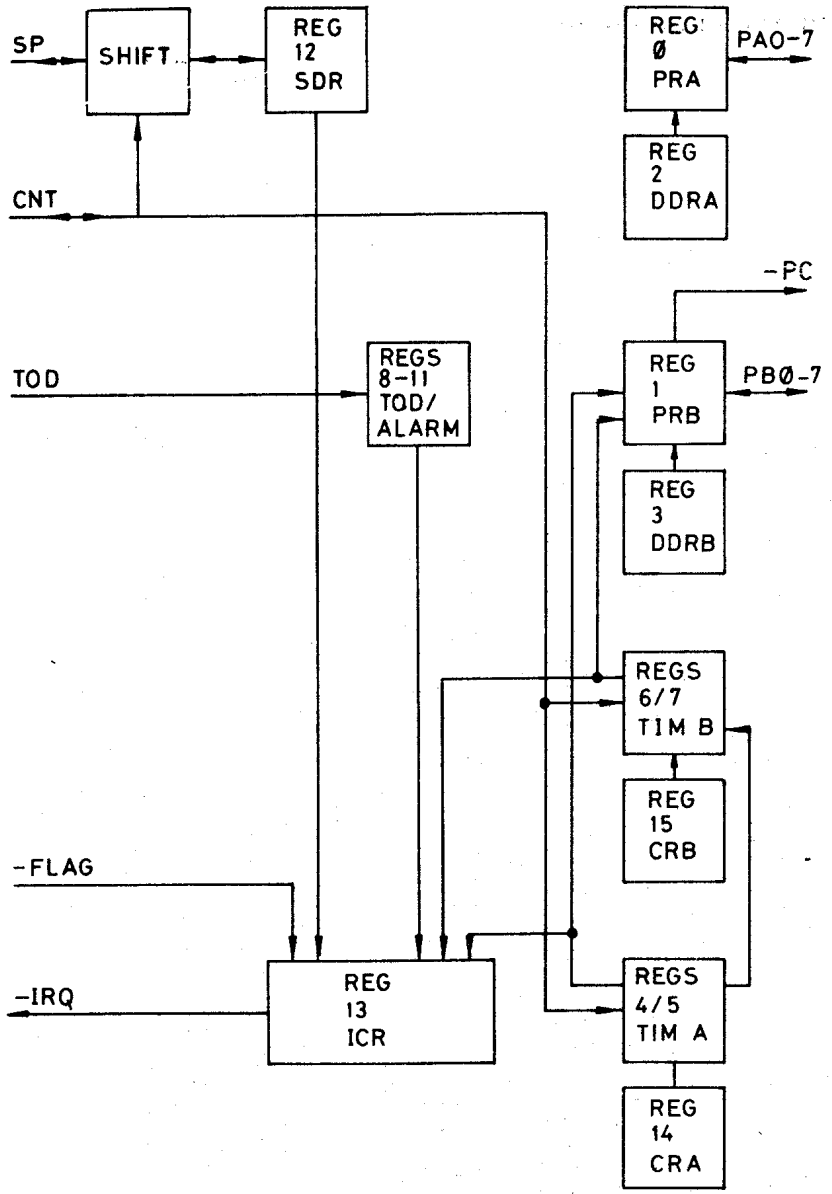
A CIA 6526-os chip blokksémáját a fejezet végén találja meg az Olvasó.
A CBM 64-es tárkiosztásában a CIA-t sajátos módon helyezték el, ezért érdemes figyelmesen elolvasni az idevonatkozó 4.6 alfejezetet.

A 40 polusú tok lábkiosztása:

1	Föld
2–9	A I/O port; 8 kétirányú bit
10–17	B I/O port; 8 kétirányú bit. A 6. és 7. bit a két timer alulcsordulásának kijelzésére programozható.
18	– PC (Port Control); csak kimenet; az adatok felhasználhatóságát jelzi a B kapun, vagy mindkét kapun.
19	TOD (Time Of Day); csak 50/60 Hz bemenet; a valós idejű órát kapcsolja.
20	+ 5 V; üzemi feszültség.
21	– IRQ (Interrupt ReQuest); csak kimenet; értéke 0, ha az ICR-ben beállított bithez tartozó esemény bekövetkezett.
22	R/W (Read/Write); csak bemenet; 0 = olvasás az adatbuszról, 1 = írás az adatbuszra.
23	– CS (Chip Select); csak bemenet; 0 = az adatbusz érvényes, 1 = az adatbusz nagy ellenállású (Tri-State).
24	– FLAG; csak bemenet; jelentése hasonló a PC jelentéséhez
25	02 (2. rendszerütem); csak bemenet; minden adatbusz akcióra csak akkor kerülhet sor, ha 02 = 1
26–33	DB7-DB0 (adatbusz); kétirányú; processzor-interface.
34	– RES (RESet); csak bemenet; 0 = CIA visszaállítása alapállapotba.
35–38	RS3-RS0 (Register Select); csak bemenet; a 16 CIA regiszter valamelyikének kiválasztására szolgál; csak akkor érvényes, ha CS = 0.
39	SP (Serial Port); kétirányú; a léptetőregiszter ütem bemenete/kimenete, vagy a timer kapcsoló (trigger) bemenete. (az ábrát l. a következő oldalon.)

4.2 A CIA regisztereinek leírása

0. REG PRA (az A port regisztere)
Elérés: READ/WRITE
0–7. bit: a regiszter tartalma megfelel a PA0-7 lábak állapotának.



1. REG PRB (a B port regisztere)
   Elérés: READ/WRITE
   0–7. bit: a regiszter tartalma megfelel a BP0–7 lábak állapotának.

2. REG DDRA (az A port adatirány-regisztere)
   Elérés: READ/WRITE
   0–7. bit: ezek a bitek határozzák meg az A port megfelelő adatbitjének irányát.
   0 = bemenet, 1 = kimenet.

3. REG DDRB (a B port adatirány-regisztere)
   Elérés: READ/WRITE
   0–7. bit: ezek a bitek határozzák meg a B port megfelelő adatbitjeinek irányát.
   0 = bemenet, 1 = kimenet

4. REG TA LO (az A timer alsó byte-ja)
   Elérés: READ
   0–7. bit: a regiszter visszaadja az A timer alsó byte-jának pillanatnyi állapotát.
   Elérés: WRITE
   0–7. bit: a regiszterbe kerül annak az értéknek az alsó byte-ja, amelytől a timernek nulláig kell visszaszámlálnia.

5. REG TA HI (az A timer felső byte-ja)
   Elérés: READ
   0–7. bit: a regiszter visszaadja az A timer felső byte-jának pillanatnyi állapotát.
   Elérés: WRITE
   0–7. bit: a regiszterbe kerül annak az értéknek a felső byte-ja, amelyről a timer indul.

6. REG TB LO (a B timer alsó byte-ja)
   Az elérés és a jelentés ugyanaz, mint a 4-es regiszternél.

7. REG TB HI (a B timer felső byte-ja)
   Az elérés és a jelentés ugyanaz, mint az 5-ös regiszternél.

8. REG TOD 10THS (az idő 1/10 másodpercekben)
   Elérés: READ
   0–3. bit: a valós idejű óra tizedmásodpercei BCD kódban.
   4–7. bit: mindig 0.
   Elérés WRITE, 7. CRB bit = 0.
   0–3 bit: a tizedmásodpercek BCD-kódban.
   4–7. bit: kötelezően 0 értékűek a bitek.
   Elérés WRITE, 7. CRB bit = 1.
   0–3. bit: a riasztási idő tizedmásodperceinek értéke BCD kódban.
   4–7 bit: kötelezően 0 értékűek a bitek.

9. REG TOD SEC (az idő másodpercben)
   Elérés: READ
   0–3. bit: az idő „egyes" másodpercei BCD kódban.
   4–6. bit: az idő „tizes" másodpercei BCD kódban.
   7. bit: mindig nulla.
   A többi elérési mód ugyanolyan, mint a 8-as regiszternél.

10. REG TOD MIN (az idő percekben)
Elérés: READ
0–3 bit: az idő „egyes" percei BCD kódban.
4–6. bit: az idő „tizes" percei BCD kódban.
7. bit mindig nulla.
A többi elérési mód ugyanolyan, mint a 8-as regiszternél

11. REG TOD HR (az idő órában)
Elérés: READ
0–3. bit: az idő „egyes" órái BCD kódban.
4. bit: az idő „tizes" órái BCD kódban.
5–6. bit: mindig nulla.
7. bit: 0 = délelőtt (AM), 1 = délután (PM)
A további elérési mód ugyanolyan, mint a 8-as regiszternél.

12. REG SDR (Serial Data Register)
Elérés: READ/WRITE
0–7. bit: ebből a regiszterből érkeznek az adatok az SP lábra, vagy megfordítva: az SP lábról a regiszterbe.

13. REG ICR (Interrupt Control Register)
Elérés: READ (INT DATA)
0 bit: 1 = az A timer alulcsordulása
1 bit: 1 = a B timer alulcsordulása
2 bit: 1 = az óra szerinti idő azonos a megválasztott riasztási idővel.
3 bit: 1 = az SDR megtelt vagy üres, a mindenkori üzemmódtól függően.
4 bit: 1 = A FLAG lábról jel érkezett.
5–6. bit: mindig nulla.
7 bit: az INT MASK és az INT DATA legalább egy bitje megegyezik.
Figyelem!: a regiszter olvasásakor az összes bit törlődik.
Elérés: WRITE (INT MASK)
A bitek értelmezése ugyanolyan, mint fent, a 7. bit kivételével.
7 bit: 1 = minden 1 értékű bit átveszi a megfelelő maszk-bitet. A többi bit nem változik. 0 = minden 1 értékű bit törli a megfelelő maszk-bitet. A többi bit nem változik.

14. REG CRA (az A Control Register)
Elérés: READ/WRITE
0 bit: 1 = az A timer startja, 0 = stopja.
1 bit: 1 = az A timer alulcsordulását a PB6 láb jelzi.
2 bit: 1 = az A timer minden alulcsordulása ellenkezőjére billenti át a PB6 lábat. 0 = az A timer minden egyes alulcsordulása a PB6 lábon egy a rendszerütem hosszával azonos HI-impulzust állít elő.
3 bit: 1 = az A timer a kimenő értékről csak egyszer vált nullára, majd leáll; 0 = az A timer a kimeneti értékről folyamatosan számol nulláig.
4 bit: 1 = új kezdőértéket kell betölteni az A timerbe.
Ez a bit strobe-ként működik. Értékét minden feltétlen betöltésnél újra be kell állítani.
5 bit: ez a bit határozza meg a timer léptetéseinek forrását. 1 = a timer a felfutó CNT-éleket számolja; 0 = a timer a rendszerütem-impulzusokat számolja.
6 bit: 1 = SP a bemenet, 0 = SP a kimenet.
7 bit: 1 = a valós idejű óra léptéke – 50 Hz; 0 = a lépték 50 Hz.

15. REG CRB (a B Control Register)
Elérés: READ/WRITE
0–4. bit: a bitek jelentése ugyanaz, mint a 14. regiszternél, de a B timerre és a PB7 lábra vonatkoztatva.
5–6. bit: ezek a bitek határozzák meg a B timer léptetésének forrását. 00 = a timer a rendszerütemeket számlálja; 10 = a timer a felfutó CNT-éleket számlálja; 01 = a B timer az A timer alulcsordulásait számlálja; 11 = a B timer az A timer alulcsordulásait számlálja, ha CNT = 1.
7. bit: 1 = a riasztás beállítása, 0 = az óra szerinti idő beállítása.

4.3 Az INPUT-OUTPUT portok

Az A és a B port egyenként egy nyolcbites adatregiszterből (PR) és egy nyolcbites adatirány-regiszterből (DDR) áll. Ha a DDR egy bitje 1 értékű, a megfelelő bit a PR-ben kimenet, és megfordítva, ha egy bit értéke a DDR-ben 0, a PR megfelelő bitje bemenet lesz. Mivel az olvasási ciklusban az adatregiszter a bemeneti és a kimeneti bitekhez rendelt lábak (PA0-7, PB0-7) állapotát visszaadja, a PB6 és PB7 bitek a timer kimeneti funkcióját is el tudják látni. A CIA és a PA/PB-re csatlakoztatott „külvilág" közötti átvitelt ún. „nyugtázási" üzemmóddal valósították meg. Erre a célra szolgálnak a PC és a FLAG vezetékek.
Miután a PRB olvasási vagy írási ciklusa végetért, egy ütemnyi időben a PB értéke 0 lesz. Ez a jel vagy arra utal, hogy a PB-n az adatok rendelkezésre állnak, vagy arra, hogy a PB fogadja az adatokat. A FLAG egy negatív impulzusél-triggerezett bemenet, amelyet például összekapcsolhatunk egy másik CIA PC-vonalával. Egy lefutó él a FLAG interrupt bitet is beállítja.
Az SDR soros adatport egy nyolcbites szinkron léptető regiszter. A CRA 6. bitje határozza meg az üzemmódot (bemenet vagy kimenet). Bemeneti üzemmódban az SP-ről érkező adatokat egy a CNT-n érzékelt felfutó jel mellett átveszi egy léptetőregiszter. Nyolc CNT impulzus után a léptetőregiszter tartalma átkerül az SDR-be és az ICR regiszter SP bitje 1-re vált.
Kimeneti üzemmódban az A timer baud-generátorként működik. Az SDR-ből érkező adatokat az A timer fél lefutási frekvenciája „kitolja" az SP-be. Így az elméletileg elérhető legnagyobb átviteli sebesség a rendszerütem 1/4-e.
Az átvitel az SDR-be irányuló adatbeírás befejeztével kezdődik, feltéve, ha az A timer jár és folyamatos üzemmódban van (CRA 0. bit = 1 és 3. bit = 0).
Az A timer által kialakított ütem megjelenik a CNT vonalon. Az adatok az SDR-ből betöltődnek a léptetőregiszterbe, majd a CNT minden lefutó éle mellett átkerülnek az SP-be.
Nyolc CNT impulzus után kialakul az SP megszakítás. Ha azonban közben az SDR-be új adatok kerültek, ezek automatikusan betöltődnek a léptetőregiszterbe és továbbítódnak. Ekkor megszakítás nem lép fel.
Az adatok átvitele az SDR-ből mindig a legmagasabb értékű bittel kezdődik, tehát az érkező adatokat is ilyen sorrendben kell továbbítani.

4.4 A timerek

A két intervallum-óra mindegyike egy 16 bites számlálóból (csak olvasható – read only) és egy 16 bites közbenső tárból (csak írható – write only) áll.
Az órába beírt adatok a közbenső tárba érkeznek, a leolvasott adatok viszont mindig a számláló pillanatnyi állását mutatják. A két óra egymástól függetlenül és egyidejűleg is használható. A különböző üzemmódok lehetővé teszik a hosszú késleltetések, változó impulzushosszak és impulzusláncok előállítását. A CNT bemenetre csatlakozva az órákkal külső impulzusokat számlálhatunk, vagy frekvenciákat is mérhetünk.
A timerekhez rendelt vezérlőregiszterekkel (CRA, CRB) az alábbi műveleteket végezhetjük el.

START/STOP(0.) bit:
Ezzel a bittel az órát elindíthatjuk vagy leállíthatjuk.

PB ON/OFF(1.) bit:
A bit bekapcsolásával az órajelet a PB vonalra vezethetjük (PB6 – A timer; BP7 – B timer). A bit értéke felülbírálja a DDRB-ben rögzített adatirányt.

TOGGLE/PUISE(2.) bit:
Ez a bit határozza meg a PB vonalon jelentkező jel impulzusának típusát. A PB vagy átbillen az ellenkező állapotba minden aláfutásnál, vagy pedig egy ütem időtartamával azonos hosszúságú pozitív impulzust állít elő.

ONE-SHOT/CONTINUOUS(3.) bit:
One-Shot üzemmódban az óra a közbenső tár értékétől indulva visszaszámol nullára, beállítja az IRC bitet, a számlálóba újra betölti a közbenső tár értékét, majd leáll. Folyamatos (Continuous) üzemmódban az előzőkben ismertetett folyamat ciklikusan ismétlődik.

FORCE LOAD(4.) bit:
Ez a bit lehetővé teszi, hogy az óra értékét bármikor újratöltsük függetlenül attól, hogy az óra éppen jár-e vagy nem.

INPUT MODE – a CRA 5. ill. a CRB 5–6. bitje:
Ezekkel a bitekkel megválaszthatjuk azt az ütemet, amellyel az óra visszafelé számlál. Az A óra ütemét vagy a rendszerütem, vagy pedig egy, a CNT-re továbbított ütem adja. A B órát nem táplálhatják az A óra lefutási impulzusai, sem a CNT = 1 értéke mellett, sem egyébként.

4.5 A valós idejű óra

A valós idejű óra (TOD) egy 1/10 másodperces felbontású, 24 órás (AM/PM) időmérő.

Regiszterei:
- 1/10 másodperc
- másodperc
- perc
- óra

Az AM/PM bit az óraregiszter legnagyobb értékű bitje.
A tárolás BCD kódban történik, így a leolvasott értékeket közvetlenül átszámítás nélkül felhasználhatjuk.
Az ütemezést egy 50/60 Hz-es jel látja el a TOD lábon (programozható a CRA 7. bitjével).
A valós idejű órához tartozik egy riasztó regiszter is, amellyel bármikor megszakítást idézhetünk elő. Ez a regiszter ugyanazokat a címeket foglalja le, mint a TOD regiszter, így az elérését a CRB 7. bitje vezérli. A riasztás regiszter csak írásra érhető el (write only). Függetlenül a CRB 7. bit értékétől, minden egyes olvasás megadja a TOD regiszter állását.
Az idő pontos beállításához és olvasásához tudnunk kell a következőket:
Az óraregiszter írásakor az óra automatikusan leáll, csak akkor indul újra, ha az 1/10 másodperc-regiszter írása véget ért.
Az óra leolvasása közben előfordulhat egy átvitel a már olvasott regiszterbe, ami pontatlansághoz vezetne, ha az óraregiszter olvasásának pillanatában a leolvasott értéket nem rögzítenénk. Emiatt a leolvasott érték mindig a közbenső tárba kerül. A közbenső tár csak az 1/10 másodpercek olvasása után szabadul fel ismét.

4.5.1 ÖTLETEK A VALÓS IDEJŰ ÓRA ALKALMAZÁSÁHOZ

Az operációs rendszer gondoskodik arról, hogy a TIS rendszerváltozó mindig az idő aktuális értékét tartalmazza.
Az idő maximális pontatlansága napi 1/2 óra.
A CIA beépített valós idejű órájának ütemét a hálózati frekvencia szabja meg.
A következő BASIC programokkal a valós idejű óra alkalmazási lehetőségei közül mutatunk be néhányat.
Az első program az óra értékének beállítását, a második pedig a leolvasását szemlélteti.
Az óra indításakor az 1/10 másodperceket mindig nullára kell beállítani.

P13

```
10 C=56328:REM CIA1 ORA KEZDOCIME
20 REM C=56584 CIA2 ORA ESETEN
30 POKE C+7,PEEK(C+7)AND127
35 POKE C+6,PEEK(C+6)OR 128
40 INPUT"ADJA MEG AZ IDOT HHPPSS FORMABAN ";A$
50 IF LEN(A$)<>6 THEN 40
60 H=VAL(LEFT$(A$,2))
70 M=VAL(MID$(A$,3,2))
80 S=VAL(RIGHT$(A$, 2))
90 IF H>23 THEN40
100 IF H>11 THEN H=H+68
110 POKE C+3,16*INT(H/10)+H-INT(H/10)*10
120 IF M>59 THEN40
130 POKE C+2,16*INT(M/10)+M-INT(M/10)*10
140 IF S>59 THEN40
150 POKE C+1,16*INT(S/10)+S-INT(S/10)*10
160 POKE C,0
```

A következő program leolvassa az óra aktuális tartalmát:

P14

```
10 C=56328:REM CIA1 ORA KEZDOCIME
20 PRINT"^S":REM C=56584 CIA2 ORA ESETEN
30 H=PEEK(C+3):M=PEEK(C+2):S=PEEK(C+1):T=PEEK(C)
40 FL=1
50 IF H>32 THEN H=H-128:FL=0
60 H=INT(H/16)*10+H-INT(H/16)*16:ON FL GOTO 80
65 IF H=12 THEN 85
70 H=H+12
80 IF H=12 THEN H=0
85 M=INT(M/16)*10+M-INT(M/16)*16
90 S=INT(S/16)*10+S-INT(S/16)*16
100 T$=STR$(T)
110 H$=STR$(H):IF LEN(H$)=2 THEN H$=" 0"+RIGHT$(H$,1)
120 M$=STR$(M):IF LEN(M$)=2 THEN M$=" 0"+RIGHT$(M$,1)
130 S$=STR$(S):IF LEN(S$)=2 THEN S$=" 0"+RIGHT$(S$,1)
140 PRINT"^S";
150 PRINT RIGHT$(H$,2)":"RIGHT$(M$,2)":"RIGHT$(S$,2)":0";
160 PRINTRIGHT$(T$,1)
170 GOTO 30
```

A STOP/RESTORE billentyűk egyidejű leütése után az óra értékét újra be kell állítani, hiszen ekkor az operációs rendszer minden regiszter tartalmát visszaállítja a bekapcsolási állapotra. A visszaállítás sajnos az 50/60 Hz-et kiválasztó bitet is érinti. Következésképpen az óra igen sokat késne.

4.6 A CIA chipek a CBM 64-ben

Ha ezeket a chipeket saját céljainkra szeretnénk felhasználni, ne feledkezzünk meg arról, hogy az operációs rendszerben meghatározott feladataik vannak. Különösen fontos ez olyan megszakítások előidézése során, amelyek hatására az operációs rendszer bizonyos rutinokat futtat.
Lehetőleg ne változtassuk meg pl. az ICR maszkokat.
Az alábbiakban összefoglaljuk a CIA chipek rendszerfeladatait:
A CIA 1 báziscíme: $DC00(56320)

0. REG (PRA)
0–7. bit: normális üzemmódban ezek a bitek határozzák meg a billentyűzet-mátrix sorának kiválasztását. Nem szabad megfeledkezni arról, hogy egyes bitek az operációs rendszertől függetlenül kapcsolatot tartanak fenn az 1-es controlporttal, amely a botkormányok és a vezérlő potenciométerek csatlakoztatására szolgál.
0–4. bit: a 0. botkormány vezérlése. A bitek jelentése rendre: felfelé, lefelé, balra, jobbra, billentyű.
6–7. bit: az A/B potenciométer kiválasztása. A két bit közül csak az egyik értéke lehet 1.

1. REG (PRB)
0–7. bit: normális üzemmódban ezek a bitek adják meg ,a billentyűzet-mátrix oszlopának értékét egy billentyű leütésekor.
0–4. bit: feladata ugyanaz, mint a 0. regiszteré, de a 2-es controlportra (1. botkormány) vonatkoztatva.
13. REG (ICR)
4. bit: kazetta-port input adatai.

A CIA 2. báziscíme: $DD00(56776)
0. REG (PRA)
0–1. bit: a VA 14–15. bitek (a videoram legmagasabb helyiértékű címbitjei).
2. bit: TXD (csak egy RS-232-es cartridge csatlakoztatása esetén foglalt, egyébként szabad).
3. bit: ATH (a soros busz kimenete).
4. bit: CLOCK (a soros busz kimenete).
5. bit: DATA (a soros busz kimenete).
6. bit: CLOCK (a soros busz bemenete).
7. bit: DATA (a soros busz bemenete).
1. REG (PRB)
0–7. bit: általában szabad. Egy RS-232 cartridge csatlakoztatása esetén a bitek jelentése a következő:
0. bit: RXD (Receive Data)
1. bit: RTS (Request To Send)
2. bit: DTR (Data Terminal Ready)
3. bit: RI (Ring Indicator)
4. bit: DCD (Data Carrier Detect)
6. bit: CTS (Clear To Send)
7. bit: DSR (Data Set Ready)
13. REG (ICR)
4. bit: RXD (csak RS-232 üzem esetén, egyébként szabad).

4.7 A bottkormány kezelése

A programozás során feltétlenül figyelembe kell vennünk, hogy a botkormány ugyanazokat a CIA biteket használja, amelyeket az operációs rendszer a billentyűzet lekérdezése közben igénybe vesz. Miközben tehát a botkormányt használjuk, fel kell függeszteni a billentyűzet lekérdezését. Ez csak olyan programban lehetséges, amely nem használja a billentyűzetet, azaz nem hajt végre egyetlen INPUT vagy GET műveletet sem.
A következő kis programmal leolvashatjuk a botkormány pillanatnyi állását:

```
P15

10 POKE 56322,224
20 J=PEEK(56320)
30 IF J(AND1)=0 THENPRINT"FEL"
40 IF J(AND2)=0 THENPRINT"LE"
50 IF J(AND4)=0 THENPRINT"BALRA"
60 IF J(AND8)=0 THENPRINT"JOBBRA"
70 IF J(AND16)=0 THENPRINT"TUZ!!!"
80 GOTO20
```

A program feltételezi, hogy a botkormányt a 2-es portra csatlakoztattuk. Az 1-es portra csatlakoztatott botkormány ellenőrzéséhez a 20-as sorban megadott címet 56321-re kell módosítani.
A végtelen ciklus miatt a programot csak a STOP/RESTORE billentyűkkel állíthatjuk le.
A következő sor beillesztésével ismét elérhetővé tehetjük a billentyűzetet:

100 POKE 56322,255

Természetesen ezt a programsort több helyre beillesztve a billentyűzetet és a botkormányt felváltva is használhatjuk.

4.8 Az IEC busz

4.8.1 EGY KIS TÖRTÉNELEM

Hallatlan esemény zaklatta fel 1974-ben a technika világát. A különböző országok mérőberendezéseket gyártó szakemberei összeültek egy kerekasztal beszélgetésre azzal a céllal, hogy kidolgozzanak és szabványosítsanak egy rendszert, amely lehetővé teszi a különböző berendezések, gépek összekapcsolását.
A dolog rendkívüliségét az adja, hogy egy új rendszerkoncepció kialakítása helyett, szembeszállva mindenféle nemzeti öntudattal a résztvevők szabványként elfogadták és kötelezőnek nyilvánították szinte változtatás nélkül egy amerikai cég már meglevő termékét. A szaklapok egyhangú üdvrivalgással fogadták az egyezményt, és a rendszert (amely egyébként a Hewlett Packard cég terméke) a General Purpose Interface Bus, rövidítve GPIB, „az elképzelhető legáltalánosabb és legbiztosabb" jelzőkkel illették.
A mikrogépekről ekkor még nem esett szó.
A rendszer elsősorban mérőberendezések, kiértékelő készülékek és nyomtatók összekapcsolását szolgálta.
A busz, amelynek nemzetközi elnevezése: IEC-625, hamarosan IEC-625 1. megjelöléssel a német ipari szabványok közé is bekerült. Az USA-ban gyakran használják az IEEE-488-as megjelölést, de ez semmiféle lényegi változást nem takar.
A busz számítógépes alkalmazási lehetősége először a mérési eredmények számítógépes kiértékelése kapcsán merült fel. Ekkor gondoltak először arra, hogy a mérőberendezésekhez számítógépes háttértárolókat (pl. lemezegységet) is lehetne csatlakoztatni.

Nem kell különösebb fantázia ahhoz, hogy kitaláljuk, hogy a Hewlett Packard cég a kellő pillanatban most is résen volt. Az azonban már messze nem természetes, hogy egy a méréstechnikától teljesen távolálló cég, a Commodore cég, a mikroszámítógépes világban szokatlan széleslátókörűségről téve tanúbizonyságot, átvette termékei közé az IEC buszt, és a CBM tulajdonosok örömére ma is gyártja.
Ennyit az IEC busz történelmi hátteréről.
Ugy gondoljuk, hogy ezt a számítógépes „középkorból" származó történetet mindenképpen érdemes volt elmesélnünk, hiszen rengetegen használják az IEC buszt, úgy, hogy az eredetéről nem tudnak semmit.

4.8.2 A „NAGY" IEC BUSZ

Az Olvasó joggal kérdezheti, hogy egy olyan könyvben, amely a C 64-es gép felépítésével foglalkozik, mi keresnivalója van a nagy CBM gépekhez készített IEC busznak?
A válasz nagyon egyszerű: egyrészt a nagy IEC busz, mint a C 64-es soros busz jóval összetettebb elődje, mindenképpen sok tanulságot nyújthat, ha általános betekintést szeretnénk nyerni ebbe a témakörbe. Másrészt, mivel az IEC buszt a C 64-es busz bővítésének tekinthetjük, egy sor közös tulajdonságuk van, tehát az előzőt tanulmányozva feltétlenül közelebb kerülünk az utóbbihoz.
Lehet, hogy sokukban felmerült már a gondolat, hogy vajon van-e valamilyen köze a közlekedési eszköznek, az autóbusznak ahhoz a „busz"-hoz, amely számítógépes szakkifejezésként terjedt el?
Talán meglepő, de a két dolognak jónéhány közös tulajdonsága van.
Az Olvasó képzeljen maga elé egy hosszú utcát több autóbusz-megállóval. A megállóhelyek megfeleltethetőek a vezetékre csatlakoztatott készülékeknek.
Tegyük fel, hogy az utcán egy valódi autóbusz halad. Az autóbusz a megállóhoz érve valóban megáll, ha új utasok szándékoznak beszállni, vagy néhány utas szeretne az autóbuszról leszállni. Az autóbusz tehát egy olyan szállítóeszköz, amellyel az utasok eljuthatnak egy adott megállóhelyről egy kiválasztott megállóhelyre. Azt, hogy a fel- és leszállás hol történjen, azt maga az utas dönti el, és döntéséről tájékoztatja az autóbusz vezetőjét.
Az már szervezési kérdés, hogy az előzetesen rögzített szabályok mindenkor biztonságos közlekedést garantáljanak.
A fentieket szinte változatlan formában elmondhatjuk, számítógépes kifejezéseket használva az egyes egységek közötti adatátvitelre vonatkoztatva.
Az autóbuszvezető (amelyből természetesen csak egy van) megfeleltethető az IEC busz vezérlőjének (controller), a megállók, amelyeken az autóbuszba beszállnak az utasok, megfelelnek a TALK (beszélő vagy adatküldő) külső egységeknek, azok az állomások pedig, ahol az utasok kiszállnak, megfelelnek a LISTEN (hallgató vagy adatfogadó) külső egységeknek.
A mikrogépes technikában a busz semmi egyebet nem jelent, mint egy olyan vezetéket, amely egy vezérlő által adott utasítások szerint dolgozik, és amelyre több készenléti állapotú egység van egyidejűleg kötve.

4.8.2.1 A csatlakoztatás és a lábkiosztás

Az IEC buszon általában egy trapézalakú 24 kimenetű Cinch típusú csatlakozó van, előfordul, hogy ehelyett D-Submin típusú 25 pólusú csatlakozót használnak, ez utóbbira most nem térünk ki.

A lábkiosztás az alábbi:

1–4 DI01-4 adatvonalak. Ezeket a vonalakat a TALKER egység látja el adatokkal, a LISTEN egység pedig innen fogadja az adatokat.
A DI01 a legalacsonyabb helyiértékű bit.

5. EOI End Or Identify. Kettős feladatú vezeték. Egyrészt a TALKER egység ezt a jelet küldi, az utolsó byte átvitele után. Másrészt a vezérlő ATN állapotban ezt a vonalat használja arra, hogy az SRO-n szolgáltatást igénylő külső egység számát nyugtázza.

6. DAV Data Valid. Ezen a vonalon kapja a TALKER egység azt az üzenetet, hogy az általa küldött adatbyte érvényes.

7. NRFD Not Ready For Data. A Listen egység a TALK egység tudomására hozza, hogy nem áll készen az adatok fogadására (pl. azért mert el van foglalva a korábban érkező adatok feldolgozásával).

8. NDAC Not Data Accepted. A LISTEN egység jelzi, hogy a buszon érkező adatbyte-ot még nem vette át. A jel ellentéte nyugtázza az adat átvételét.

9. IFC Interface Clear. A vezérlő minden csatlakoztatott egységet alapállapotba hoz (RESET).

10. SRQ Service Request. A külső egység ezen a vonalon bejelentheti műveletigényét. A vezérlő innen értesül arról, hogy pl. az egység egy feladat elvégzését befejezte, új adatra vár stb. A jel észlelését követően a vezérlő egy „azonosító" ciklusban (EOI és ATN vonalakkal) megállapítja, hogy melyik egység jelentkezett be. Ez utóbbi a C 64-esen nem működik.

11. ATN ATteNtion. A vezérlő, valahányszor utasítást készül küldeni, aktivizálja ezt a vonalat. A figyelmeztetés hatására az utasítást minden egység átveszi, függetlenül attól, hogy az neki szól vagy sem. A következő lépésben érkezik az egységszám, amelynek felismerése után a nem érintett egységek „leválhatnak" a buszról.

12. SHIELD – a buszkábel árnyékolása.

13–16. DI05-DI08 Az adatbusz felső négy bitje.

17. REN Remote ENable. Ez a vonal mindig aktív. Jelzi a buszra csatlakoztatott egységeknek, hogy a busz üzemben van.

18–24. GND földvezeték, amelyen belül a 24-est külön logikai földként deklaráljuk, ami azt jelenti, hogy a buszvonalakon észlelhető jelszintet ehhez a szinthez viszonyítjuk.

4.8.2.2 Az egységszámok

Az IEC busz működésének alapvető követelménye, hogy a csatlakoztatott készenléti állapotú egységek közül mindig csak az az egy legyen aktív, amelyre a pillanatnyi utasítás vonatkozik.
A választás az egységek között csak úgy lehet egyértelmű, ha mindegyikük egyedi, megkülönböztetett jellel van ellátva. A megkülönböztető jelet egységszámnak nevezzük. Az egységszám ráadásul nem egyszerűen egy „házszám", amely tényleg csak megkülönböztetésül szolgál.
Az egység kiválasztásán túl azt a feladatot is meghatározza, amit az egységnek el kell végeznie, nevezetesen, hogy az egység adatokat fogad vagy küld.
A címbyte két részből áll: az alsó byte tartalmazza a tényleges egységszámot (0-tól 15-ig), a felső byte pedig meghatározza a műveletet (16-tól 240-ig).

A lehetséges müveletek és kódjaik:

32 Az egység LISTEN állapotban van, tehát adatokat fogad. Ezt a kódot eredményezi pl. a BASIC PRINT# utasítás.

64 Az egység TALK állapotban van, tehát adatokat küld. (pl. a BASIC GET#, illetve INPUT# utasítás végrehajtása közben).

48 A LISTEN üzemmód befejeződött. Ez esetben az alsó byte értéke mindig 15.
80 A TALK üzemmód befejeződött. Az alsó byte értéke most is 15.

A 4-es egységszámú nyomtató egy teljes címbyte-ja például:

32 + 4 = 36 ($24)

4.8.2.3 A másodlagos cím

Az Olvasó bizonyára tudja, hogy egy lemezegységen egyszerre több file-t is meg lehet nyitni. Az OPEN utasításokban ilyenkor a különböző file-okhoz különböző logikai számot és adatcsatornaszámot (másodlagos címet) kell rendelni. Pl.:

```
10 OPEN 1,8,6, "FILE 1"
20 OPEN 2,8,7, "FILE 2"
```

A másodlagos cím jelentése a külső egység típusától függ.
A lemezegység esetében például elsősorban az egyes file-ok adatainak elkülönítését szolgálja. Általános vezérlőszerepe az IEC buszon tehát nincs.
A lemezegység másodlagos címe két részből áll. Az alsó négy bit tartalmazza azt az értéket, amit adatcsatornaként az OPEN utasításban megadtunk, ennek értéke 0 és 15 közé kell, hogy essen. A felső négy bit arról informálja a rendszert, hogy milyen művelet tartozik egy adott pillanatban az adatcsatornához. Pl.:

```
 96 PRINT, INPUT vagy GET
224 CLOSE
240 OPEN
```

A 0 és 15 közé eső értékek közül a 0 és 1 különleges jelentésű. Az egység a 0-t LOAD, az 1-et SAVE utasításként értelmezi, így ezeket az értékeket az OPEN utasításban nem lehet használni. Ha a külső egység Commodore nyomtató, a másodlagos cím a karakterkészlet kiválasztására szolgál.

4.8.2.4 Az ST rendszerváltozó

Az IEC buszon lezajlott művelet kimeneteléről az ST változóban tárolt információ alapján tájékozódhatunk. A programozónak minden INPUT, PRINT művelet után célszerű lekérdeznie a rendszerváltozó értékét, egyébként ugyanis nem szerezne tudomást pl. az IEC buszon fellépő üzemzavarról, és ez adatvesztéshez vezetne.

Az ST változó lehetséges értékei és azok jelentése:

1 Az OPEN és PRINT utasítások végrehajtása közben keletkezett hibára utal. Az adatbyte átvétele után az egység nem nyugtázta az adatátvitelt 64 ms-on belül.
2 Az INPUT vagy GET utasítás végrehajtása közben a TALK egységtől nem érkezett adat 64 ms-on belül.
64 Az éppen beolvasott adatbyte-tal együtt az egységről EOI jel érkezett, ami a lemezegység esetében a file végére utal.
–128 A címzési kísérletre semmi válasz nem érkezett a buszról. A BASIC program ekkor a DEVICE NOT PRESENT hibaüzenettel leáll.

A fenti értékek kombinációi is előfordulhatnak az ST rendszerváltozó értékei között. Legbiztosabb, ha logikai összehasonlítással (maszkoltan) kérdezzük le az információt. A file végét pl. a következőképpen:

```
100 GET# 1, A$ : IF (ST AND 64) THEN . . .
```


A file végén az ST értéke elvileg 66 is lehet, ha ugyanakkor fellépett az a jelenség is, amelyhez a 2 ST értéket rendeltük.

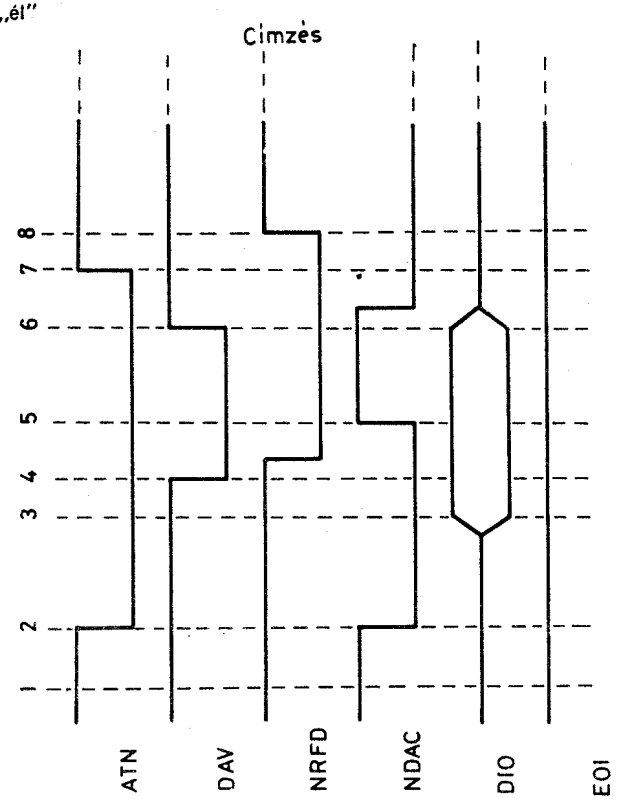
4.8.2.5 A címzés

A következő rövid kis program kapcsán elemezzük, hogy hogyan választja ki az IEC busz-vezérlő a megfelelő egységet, és hogyan történik az adatátvitel.

```
10 OPEN 1,4
20 PRINT#1, "A"
```

A program végrehajtásának eredménye egy A betű a nyomtatón. A művelet (nyomtatás) végrehajtását a 20-as sor váltja ki. Az OPEN hatására a nyomtató még nem értesül a feladatról, hiszen az utasítás nem tartalmaz az egység számára semmilyen információt, szemben a lemezegységre vonatkoztatott OPEN-nel, amelyben szerepel a file neve.
A lábkiosztást ismertető alfejezet alapján elemezzük a következő ábrákat.
A számoknak megfelelő jelszinteket idődiagramon ábrázoltuk.

1 A busz nyugalmi állapotban van, a vezérlővonalak nem aktívak, azaz magas szinten állnak. Az adatvonalak jelszintje indiferens; a C 64-esen magas.
2 Az ATN vonal jelzi a buszra kapcsolt egységeknek, hogy parancs érkezik. Válaszul az összes egység NDAC vonala alacsony szintre áll, tudtára adva a vezérlőnek, hogy a busz „él"



A számítógép megvizsgálja, hogy az NRFD és NADC vonalak ellentétes jelszintet mutatnak-e. Ha nem, a DEVICE NOT PRESENT hibaüzenetet küldi.

3 Az egységszám átkerül az adatvonalra, ez esetünkben 36 (32 + 4), ami arra utal, hogy a 4-es egységszámú nyomtatót LISTEN egységgé nyilvánítjuk.

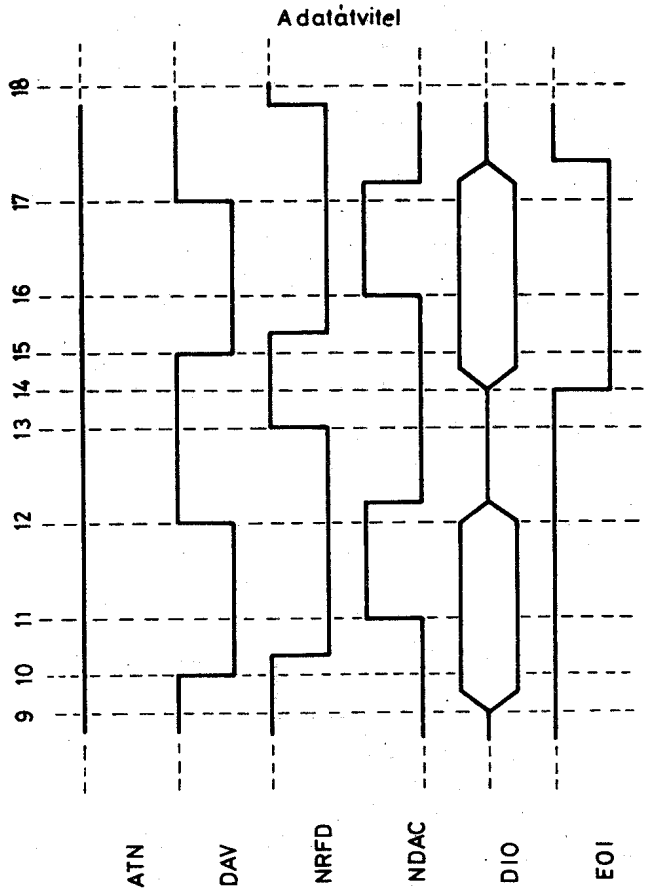
4 Az adatvonal érvényességét a DAV jelzi. Az NRFD vonal alacsony, mivel az egység az érkező adat feldolgozását végzi, és emiatt nincs készenléti állapotban.

5 A gép 64 ms-ot vár az egység pozitív válaszára (NDAC magas). Ha nem érkezik válasz, az ST változó értéke 1 lesz és a gép a művelet végrehajtását felfüggeszti.

6 A nyugtázást követően a gép a DAV jelet leveszi és az egységszámot is törli az adatvezetékről.

7 Az ATV vezeték magas szintjéből a gép arra következtet, hogy a címzési ciklus végetért. Ha másodlagos címet is kellene továbbítani, az ATN továbbra is alacsony szinten maradna, és egy DAV jellel a másodlagos cím is az adatvonalra kerülne.

8 A nyomtató az NRFD alacsonyra állításával jelzi, hogy kész az adat fogadására. A többi egység nem vesz részt a további feldolgozásban.



4.8.2.6 Az adatátvitel

9 Az "A" karakter az adatvezetékre kerül.

10 Az adat érvényességét a DAV jelzi.

11 A nyomtató az NDAC vonalon nyugtázza az adatot, majd az NRFD vónalon közli, hogy további adatot nem tud fogadni.

12 A DAV vonal ismét magas, és az adatot az egység átveszi.

13 A nyomtató feldolgozza az adatot és ismét készenléti állapotba kerül (az NRFD alacsony).

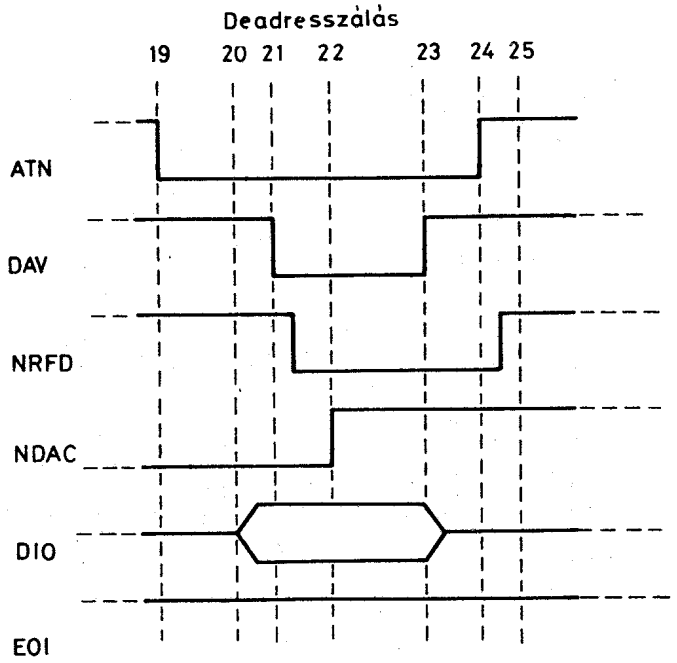
14–15 A 20-as sorban ugyan csak egy PRINT#1, 'A" utasítás áll, az operációs rendszer mégis automatikusan egy CHR$(13) karaktert is továbbít. Most ezt a karaktert tartalmazza az adatvonal. Minthogy a CHR$(13) egyben az utolsó karakter is, az adatciklus véget ér, és az EOI vonal aktív lesz.
A busz ismét nyugalmi állapotba kerül.

4.8.2.7 A deadresszálás

19 Az ATN riasztja az egységeket, ismét utasítás érkezik!

20–21 Az adatvonalra 63 (48 + 15) érték kerül. A 4.8.2.2 alfejezetben leírtak szerint 48 a LISTEN üzemmód végét jelzi, amely mellett az alsó byte értéke mindig 15.

22–25 Az események azonosak az 5–8 fázis eseményeivel, végül a busz ismét nyugalmi állapotba kerül.



4.8.3 SOROS BUSZ A C 64-ESEN

A C 64-es IEC busz csatlakozóját megvizsgálva, arra a következtetésre jutunk, hogy ez a legcsekélyebb mértékben sem hasonlíthat az előzőekben ismertetett rendszerre. A különbség szembeszökő.
A látszat azonban csal, és ha külsőre nem is hasonlítanak egymásra, mindkettő messzemenően az 1974-es konferencián született egyezmény szerint készült.
Hogy a C 64-es soros busz valójában miért tér el a megszokott IEC busztól, arra nehéz válaszolni. Valószínűleg azért, mert egy kisebb csatlakozót kellett készíteni.
Másrészt feltehető, hogy a gyártók igyekeztek viszonylag olcsó megoldást keresni, hogy ezzel a szélesebb igényt is ki tudják elégíteni, ami nem esik bele pl. a CBM 8050-es egység árkategóriájába.

4.8.3.1 Az alapelvek

A soros busz mindössze öt lábat tartalmaz:

1 SRQ — jelentése ua., mint amit a 4.8.2.1 alfejezetben leírtunk
2 GND
3 ATN — jelentése azonos az előzővel
4 CLK CLOCK — a soros busz az adatokat nem byte-onként, hanem bitenként továbbítja. A TALK egység egy impulzussal jelzi minden bit érvényességét a CLK vonalon keresztül.
5 DATA — Az egyetlen adatvonal, amelyen keresztül az adatbyte a legalsó bittel kezdve bitenként továbbítódik.

Az Olvasó persze most nem érti, hogy hová lettek a DAV, NRFD, NDAC stb. vonalak, hiszen azt állítottuk, hogy a soros busz is az IEC buszhoz hasonló elven működik.
Ez valóban így is van, de mivel csak 5 vonal áll rendelkezésre, a gyártóknak más módot kellett az alapelv megvalósítására keresniük.
Az előzőekben bemutatott adatáramlási folyamatot most a DATA és CLK vonalak vezérlik. A jelszintek ugrása között eltelt időben mindkét vonal értesül arról, hogy milyen jellegű átvitel történik (NDAC, EOI stb.).
Érthető, hogy a műveletek hibátlan végrehajtásában az időzítés rendkívül fontos szerepet játszik. Kezdetben pontosan ilyen jellegű nehézségek miatt jelentek meg a piacon hibás termékek, amelyek többnyire ugyan külső gyártóktól származtak, de mint ismeretes, a Commodore cégnek is gondot okozott például a VC 1526-os nyomtató illesztése.
A dolog bonyolultsága miatt az alábbiakban csak a leglényegesebb kérdésekre térünk ki, amelyek ismerete nem biztos, hogy elegendő lesz ahhoz, hogy az Olvasó saját külső egységeket építsen. Ha valaki minden figyelmeztetés ellenére sem riad vissza a feladattól, feltétlenül szerezzen be néhány nagyon érzékeny mérőműszert, amellyel a soros buszon lezajló folyamatokat láthatóvá tudja tenni.
Ez esetben némi segítséget biztosan nyújtanak az alábbiak:

4.8.3.2 A címzés

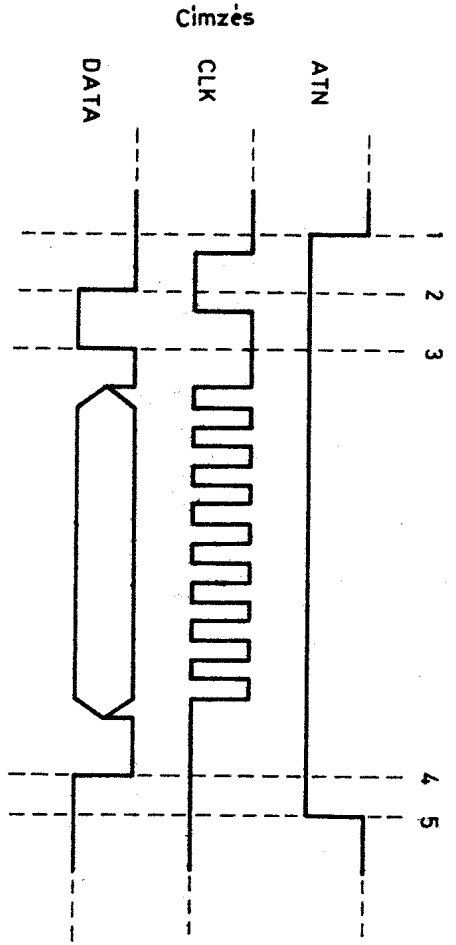
1 Az ATN alacsony, ami azt jelzi, hogy utasítás érkezik.
2 Válaszul a külső egység 1 ms-on belül az adatvonalat alacsonyra állítja. A gép erre azonnal reagál (a CLK magas), jelezve hogy egy byte elő van készítve az átvitelre.
3 A DATA vonal átveszi az NRFD szerepét, azaz mindaddig alacsony marad, amíg az egység nincs készen az adat fogadására (DATA = magas).

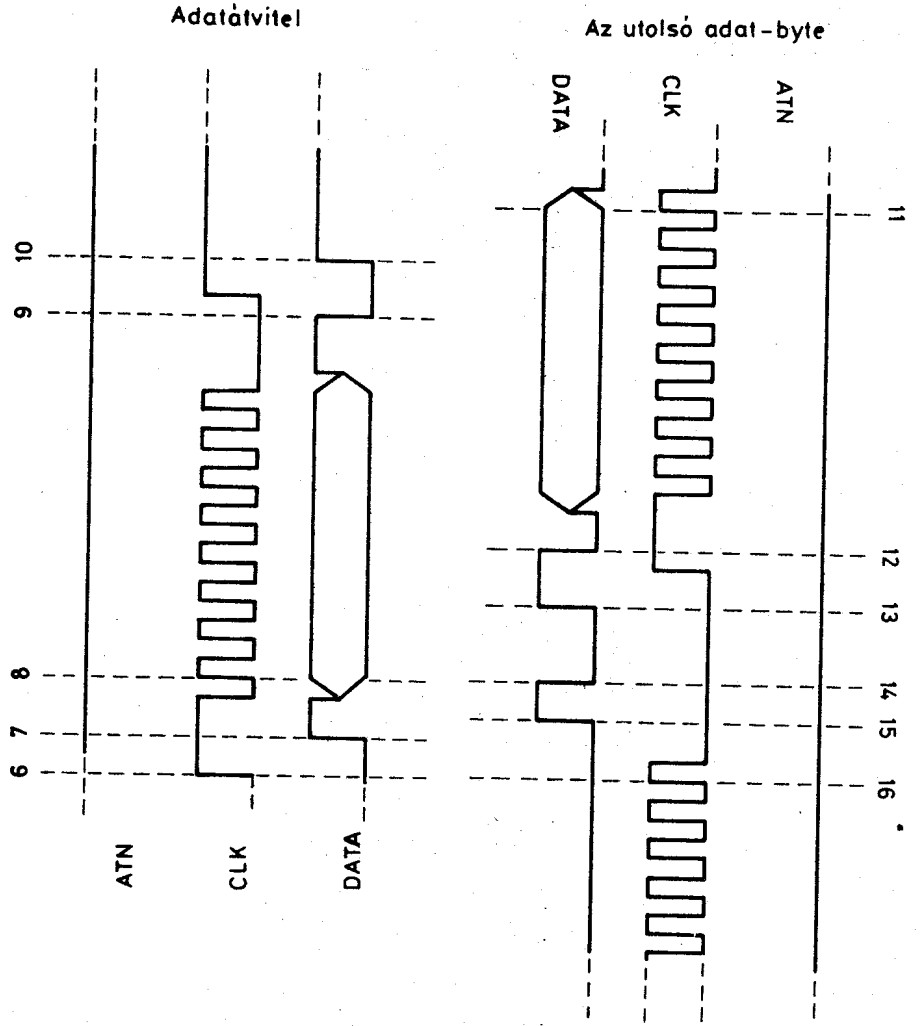
3–4 A DATA vonalon továbbítódik az egységszám; minden CLK magas-impulzussal egy bit. A CLK szerepe hasonló a DAV szerepéhez.
5 A nyomtatónak a DATA vonalat 1 ms-on belül alacsonyra kell állítania (NDAC).

A DATA vonalnak lényegében három feladata van: egyedül látja el a párhuzamos IEC busz DIO, NRFD és NDAC vonalainak feladatát.
Míg a párhuzamos busz esetében az adatvonalat mindig az aktuális TALK egység vezérelte, addig a soros busz adatvonalát a TALK és LISTEN egységek felváltva vezérlik.

4.8.3.3 Az adatátvitel

6 Az adatátvitel azzal kezdődik, hogy a TALKER egység a CLK vonalat magasra állítja, ezzel jelezve, hogy az adatbyte elő van készítve.
7 A LISTEN egység a DATA vonal magasra állításával jelzi készenlétét (az NRFD feladata).





8–9 Minden CLK impulzus egy bitet küld az adatvonalra.

10 A teljes byte átvételét a LISTEN egység a DATA vonal alacsonyra állításával nyugtázza (az NDAC és NRFD feladata) Az előző példa alapján ez pontosan az "A" karakter.

11–12 A leírt folyamat megismétlődik, az átvitt karakter a lezáró: CHR$(13).

13–14 Hol marad az EOI jel? Ez lesz a DATA vonal negyedik feladata.
Az eddigiek szerint miután a LISTEN egység a DATA vonalat alacsonyra állította (NRFD), a TALKER egység 0.2 ms-on belül az adatot a CLK alacsony szintje mellett továbbítja. Ha a CLK továbbra is magas marad, akkor már a TALKER adatbyte-ot is küldött, tehát az átvitelnek vége. Értesülését a LISTEN egység a DATA vonal alacsonyra állításával nyugtázza.

15 Amint a DATA vonal ismét magas (NRFD), a TALK egy üres byte-ot (CHR$(0)) küld a LISTEN egységnek (16). Végül a busz nyugalmi állapotba kerül.

Az egységszám törlése szintén az 1–5 pontokkal írható le, azzal a változtatással, hogy a továbbítandó byte tartalma 63.
Amint látható, az átvitel rendkívül precízen és biztonságosan szervezett folyamat. A soros busz párhuzamos busszal szembeni hátránya elsősorban az adatátvitel sebességében mutatkozik meg. A hátrányt azonban jelentősen csökkenti a lerövidített várakozási idő (1 ms a 64 ms-al szemben).
Reméljük, igyekezetünk nem hiábavaló és az IEC busz belső felépítésébe nyújtott rövid betekintés sokaknak hasznára válik.

5. FEJEZET
A BASIC – MÁS SZEMMEL

5.1 Így dolgozik a BASIC interpreter

A Commodore 64-es kényelmes BASIC értelmezővel (interpreter) dolgozik, amelynek elődje a Commodore PET 2001-eshez kifejlesztett MICROSOFT BASIC.
A BASIC nyelv változatszáma 2.0, ami a 3000-es Commodore-sorozat nyelve.
Először röviden összefoglaljuk az értelmező munka folyamatát.
Egy programsor gépelésekor az értelmező ellenőrzi, hogy van-e BASIC utasításszó a sorban Ha van, akkor azt átalakítja egy egy byte-os interpreter-kóddá, amely *token* néven is ismeretes. Ezután a sort a sorszámnak megfelelően beilleszti a programba. Futáskor a program az interpreter-kód alapján ismeri fel az utasítást. A kód legmagasabb bitjének (7. bit) értéke mindig 1. Az interpreter a kód alapján egy táblázatban megkeresi annak a rutinnak a címét, amely az utasítást végrehajtja. A következő ciklusiépésben betöltődik és hasonló módon végrehajtódik a következő utasítás. Az utasítások címeit a 7. fejezet, a rutinok részletes leírását pedig a 8. fejezet tartalmazza.
Ahhoz, hogy saját függvényeket és rutinokat tudjunk használni, ismernünk kell az adattárolás módját.
A BASIC három adattípust különböztet meg egymástól:

– valós vagy lebegőpontos számok (REAL)
– egész vagy fixpontos számok (INTEGER)
– füzérek (STRING)

A valós számok értéke a ± 1E–39-től ± 1E38-ig terjedhet.
Az egész számok értékkészlete – 32768-tól 32767-ig terjed.
A füzérek karakterláncok, amelyek hossza 0-tól 255-ig terjed.

Hogyan tárolja az interpreter a változókat?
A valós változók bejegyzései hét byte-ot foglalnak el;
az első két byte tartalmazza a változó nevét (ASCII kódban), a következő öt byte pedig a változó értéket.
A valós számok két részből állnak: az első byte-on található a kitevő, a következő négy byte-on pedig a mantissza (féllogaritmusos ábrázolás).
Az egész számok tárolásához csak két byte-ra van szükség; a 16 bites bináris ábrázolás alsó és felső byte-jára.
A füzéreknél az első byte a füzér hosszát tartalmazza (0–255), a következő két byte pedig a füzér címét.
A változótípusokat az interpreter a név alapján különbözteti meg egymástól. Az egész típusú változó nevének első és második betűjét tartalmazó byte 7. bitje 1.
A füzérek nevének második betűjét tartalmazó byte legmagasabb bitje 1.
A RAM-ban a változóterület közvetlenül a BASIC program után kezdődik. A nulláslapon egy mutató utal a táblázat kezdőcímére (45, 46 azaz $2D, $2E).
Hajtsuk végre a következő értékadó utasításokat:

```
A = 10.3
B% = – 23
C$ = "Commodore 64"
```


Végrehajtás után vizsgáljuk meg egy monitorprogrammal a tár tartalmát:

```
002D 03 08                                        a változótáblázat a $0803-nál kezdődik
0803 41 00 84 24 CC CC CD                         az A változó tartalma 10.3 lebegőpontos érték
080A C2 80 FF E9 00 00 00                         a B% változó tartalma bináris –23
0810 43 80 0C F4 9F 00 00                         a C$ változó hossza 12, tartalma a $9FF4 címen található
```

5.2 A program begépelésétől a feldolgozásig

Hogyan lehet változókat és kifejezéseket BASIC programból egy gépi kódú programnak átadni?
Az interpreter FRMEVL nevű és $AD9E kezdőcímű rutinja beolvas egy tetszőleges kifejezést a BASIC területről, és azt kiértékeli.
A rutin valós aritmetikával dolgozik, ha a kifejezés egész típusú változót vagy állandót tartalmaz; az interpreter ezeket a műveleteket elvégzése előtt valóssá konvertálja.
A rutin a numerikus és a szöveges kifejezések kiértékelését is el tudja végezni, a típust az interpreter egy kapcsoló alapján különbözteti meg. A kapcsoló a $0D(13)-as címen található; ha tartalma $00, a kifejezés numerikus, ha $FF, a kifejezés szöveges. A numerikus adatot az 1-es lebegőpontos akkumulátorban tároljuk, melynek rövidített neve FAC. A FAC a nulláslap $61-es címén található. Az aritmetikai műveletek elvégzéséhez rendelkezésre áll még egy lebegőpontos akkumulátor, az ARG, a $69-es (105) tárcímen. A függvények hívásakor a függvény argumentuma mindig a FAC-be kerül és az eredmény is innen töltődik vissza. Szövegkifejezések kiértékelésére az interpreter szintén két kitüntetett tárcímet használ, az ún. füzérleíró címeket, amelyben átmenetileg a füzér hosszát és címét tárolja. Ezek a tárcímek a $64 és a $65 (100/101) címek.
Híváskor a rutin a $B475-ös címen kezdődő programrésszel betölti a füzér hosszát az akkumulátorba, címét pedig az X (alsó byte) ill. az Y (felső byte) regiszterekbe.
Az aritmetikai műveleteket az interpreter alábbi rutinjai végzik:

Cím	Művelet	
$B853	Kivonás	FAC = ARG–FAC
$B86A	Összeadás	FAC = ARG+FAC
$BA28	Szorzás	FAC = ARG*FAC
$BB12	Osztás	FAC = ARG/FAC
$BF7B	Hatványozás	FAC = ARG felemelve a FAC kitevőre

Az interpreter és az operációs rendszer további rutinjainak címét és feladatuk leírását a 7. fejezetben találja meg az Olvasó.

5.3 Hogyan használhatiuk ki minél jobban a Commodore BASIC nyelv lehetőségeit?

5.3.1 HOGYAN BŐVITSÜK A BASIC-ET?

Eddigi ismereteink alapján a gépi kódú programokat a SYS és a USR utasítások segítségével építhettük be a BASIC programba.

Ebben a fejezetben egy elegánsabb módszert mutatunk be. Vegyük szemügyre a ROM listát. Keressünk benne egy olyan helyet, ahol az interpreter betölt és végrehajt egy BASIC utasítást:

```
A7E1   6C  08  03   JMP($0308);   az ugrási cím általában a $A7E4
A7E4   20  73  00   JSR $0073 ;   a következő karakter beolvasása a BASIC
                                 szövegből
A7E7   20  A7  ED   JSR $A7ED;   az utasítás végrehajtása
A7EA   4C  AE  A7   JMP $A7AE;   vissza az interpreter ciklushoz
```

Értelmezzük a fenti részletet. A $0308/$0309 címeken található egy mutató, amelyet saját gépi kódú rutinunk kezdőcímeként definiálhatunk. A gépi kódú rutin feladata a kódvizsgálat. Megszokott dolog, hogy a BASIC-bővítő utasításokat valamilyen különleges karakterrel, pl. felkiáltójellel kezdjük, megkülönböztetésül az eredeti BASIC utasításoktól.
Pl.:

```
100 !PRINT
```

A fenti utasítás jelölhet pl. egy saját nyomtató rutint.
A saját gépi kódú rutin beolvassa a BASIC területről az utasítást, és keresi a felkiáltójelet. Ha megtalálja, akkor elágazik egy további saját rutinra, egyébként visszatér a BASIC interpreterhez. A programrészlet lehet pl. a következő:

```
DECODE   JSR $0073       ;  CHRGET, a következő karakter beolvasása
         CMP #"!"        ;  összehasonlítás a különleges karakterrel
         BEQ FOUND       ;  elágazás a saját rutinra
         JSR $0079       ;  CHRGOT, a kapcsoló visszaállítása
FOUND    JSR $0073       ;  CHRGET, a következő karakter beolvasása
         JSR COMMAND     ;  a saját utasítás végrehajtása
         JMP $A7AE       ;  vissza az interpreter-ciklushoz
```

A $0308-as és a $0309-es címekre indításkor természetesen be kell írni a DECODE nevu szubrutin kezdőcímét.
Ha többféle bővítő utasítást használunk, meg kell írni azt a gépi kódú rutint, amely az egyes utasítások közül kiválasztja, azonosítja a megfelelőt.

5.3.2 HARDCOPY – RENEW – PRINT USING

a) A hardcopy

A program a képernyő aktuális tartalmát kinyomtatja. A nyomtató egységszáma 4. A rutint a SYS 9 * 4096 utasítással hívhatjuk.

P16.ASS

```
00BA          100 FA      = $BA
00B8          110 LF      = $B8
0071          120 TEMP    = $71
00B7          130 FNLEN   = $B7
00B9          140 SA      = $B9
FFC0          150 OPEN    = $FFC0
FFC9          160 CKOUT   = $FFC9
FFD2          170 BSOUT   = $FFD2
FFE1          180 STOP    = $FFE1
0067          190 STORE   = $67
FFCC          200 CLRCH   = $FFCC
FFC3          210 CLOSE   = $FFC3
```



```
1000            220         ;
9000            230         *=$9000     ; HARDCOPY
9000            240         ;
9000 A9 04      250         LDA #4
9002 85 BA      260         STA FA      ; NYOMTATO EGYSEGSZAMA
9004 A9 7E      270         LDA #126
9006 85 B8      280         STA LF      ; LOGIKAI FILE-SZAM
9008 A9 00      290         LDA #0      ; KEPERNYO ALSO BYTE
900A A0 04      300         LDY #4      ; KEPERNYO FELSO BYTE
900C 85 71      310         STA TEMP    ; CIM TAROLASA
900E 84 72      320         STY TEMP+1
9010 85 B7      330         STA FNLEN   ; NINCS FILENEV
9012 85 B9      340         STA SA      ; MASODLAGOS CIM 0
9014 20 C0 FF   350         JSR OPEN    ; NYOMTATO FILE NYITASA
9017 A6 B8      360         LDX LF      ; NYOMTATO LOGIKAI FILE-SZAMA
9019 20 C9 FF   370         JSR CKOUT   ; OUTPUT EGYSEG A NYOMTATO
901C A2 19      380         LDX #25     ; KEPERNYOSOROK SZAMA
901E A9 0D      390 LOOP    LDA #13     ; UJ SOR
9020 20 D2 FF   400         JSR BSOUT   ; NYOMTATORA
9023 20 E1 FF   410         JSR STOP    ; STOP BILLENTYU LEKERDEZES
9026 F0 2E      420         BEQ EXIT    ; AZ IRAS ISMET A KEPERNYORE TORTENIK
9028 A0 00      430         LDY #0
902A B1 71      440 LOOP2   LDA (TEMP),Y ; KEPERNYO OSZLOPOK SZAMA
902C 85 67      450         STA STORE
902E 29 3F      460         AND #$3F
9030 06 67      470         ASL STORE
9032 24 67      480         BIT STORE   ; KEPERNYOKOD
9034 10 02      490         BPL * + 4   ; ASCII-KODBA ATALAKITANI
9036 09 80      500         ORA #$80
9038 70 02      510         BVS * + 4
903A 09 40      520         ORA #$40
903C 20 D2 FF   530         JSR BSOUT   ; ES NYOMTATORA KULDENI
903F C8         540         INY
9040 C0 28      550         CPY #40     ; SOR VEG "?"
9042 D0 E6      560         BNE LOOP2
9044 98         570         TYA
9045 18         580         CLC         ; IGEN, MUTATO A KOVETKEZO SORRA
9046 65 71      590         ADC TEMP    ; SOR SZAMAT NOVELNI
9048 85 71      600         STA TEMP
904A 90 02      610         BCC * + 4
904C E6 72      620         INC TEMP + 1
904E CA         630         DEX         ; MAR MINDEN SOR KIIRVA "?"
904F D0 CD      640         BNE LOOP
9051 A9 0D      650         LDA #13
9053 20 D2 FF   660         JSR BSOUT   ; UJ SOR
9056 20 CC FF   670 EXIT    JSR CLRCH   ; NYOMTATO CSATORNA ZARASA
9059 A9 7E      680         LDA #126
905B 4C C3 FF   690         JMP CLOSE   ; VALAMENNYI CSATORNA ZARASA
905E          65535         .END

BSOUT =FFD2  CKOUT =FFC9  CLOSE =FFC3  CLRCH =FFCC  EXIT  =9056  FA    =00BA
FNLEN =00B7  LF    =00B8  LOOP  =901E  LOOP2 =902A  OPEN  =FFC0  SA    =00B9
STOP  =FFE1  STORE =0067  TEMP  =0071
```

A program BASIC betöltőprogramja:

```
P17
100 POKE 56,9*16:CLR: FOR I=36864 TO 36957
110 READ X:POKE I,X : S=S+X:NEXT
120 DATA 169,  4,133,186,169,126,133,184,169,  0,160,  4
130 DATA 133,113,132,114,133,183,133,185, 32,192,255,166
140 DATA 184, 32,201,255,162, 25,169, 13, 32,210,255, 32
150 DATA 225,255,240, 46,160,  0,177,113,133,103, 41, 63
160 DATA   6,103, 36,103, 16,  2,  9,128,112,  2,  9, 64
170 DATA  32,210,255,200,192, 40,208,230,152, 24,101,113
180 DATA 133,113,144,  2,230,114,202,208,205,169, 13, 32
190 DATA 210,255, 32,204,255,169,126, 76,195,255
200 IF S<>12023 THEN PRINT"HIBAS ADAT !!":END
210 PRINT"RENDBEN !"
```

b) BASIC program helyreállítása – a RE–NEW

Mindenkivel előfordulhat, hogy miután kiadott egy NEW parancsot, rájön, hogy a tárbeli programot mégsem kellett volna törölnie. Mivel a NEW parancs hatására az operációs rendszer valójában nem törli a programot, csak a BASIC terület mutatóit változtatja meg, a látszólag elveszett program visszaállítható, ha közben nem írtunk be új programsorokat. A következő gépi kódú program ezt a feladatot végzi el. Kezdőcíme 12*4096+15*256, azaz 52992.

```
P18.ASS

1000                100         ; RE-NEW FUNKCIO
1000                110         ; A TOROLT PROGRAMOT VISSZAHOZZA
002B                120 PRGSRT  = $2B
0022                130 TEMP    = $22
002D                140 PRGEND  = $2D
A663                150 CLR     = $A663
1000                160         ;
CF00                170         *=$CF00
CF00                180         ;
CF00 A5 2B          190         LDA PRGSRT      ; BASIC PROGRAMSTART
CF02 A4 2C          200         LDY PRGSRT+1
CF04 85 22          210         STA TEMP        ; MUTATOKENT TAROLNI
CF06 84 23          220         STY.TEMP+1
CF08 A0 03          230         LDY #3
CF0A C8             240 NULL    INY
CF0B B1 22          250         LDA (TEMP),Y ; ELSO SOR VEGET KERESI
CF0D D0 FB          260         BNE NULL        ; (NULLA BYTE)
CF0F C8             270         INY
CF10 98             280         TYA
CF11 18             290         CLC
CF12 65 22          300         ADC TEMP        ; OFFSET HOZZAADASA
CF14 A0 00          310         LDY #0
CF16 91 2B          320         STA (PRGSRT),Y  ; ES TAROLASA MUTATOKENT A KOVETKEZO
 SOR KEZDETERE
CF18 A5 23          330         LDA TEMP+1
CF1A 69 00          340         ADC #0          ; SORT MEGORIZNI
CF1C C8             350         INY
CF1D 91 2B          360         STA (PRGSRT),Y
CF1F 88             370         DEY             ; MOST NULLAT TARTALMAZ
CF20 A2 03          380 TO      LDX #3
CF22 E6 22          390 TDREIO  INC TEMP
CF24 D0 02          400         BNE *+4         ; A PROGRAM VEGET
CF26 E6 23          410         INC TEMP+1      ; HAROM NULLA BYTE JELOLI
CF28 B1 22          420         LDA (TEMP),Y
CF2A D0 F4          430         BNE TO
CF2C CA             440         DEX
CF2D D0 F3          450         BNE TDREIO
CF2F A5 22          460         LDA TEMP
CF31 69 02          470         ADC #2
CF33 85 2D          480         STA PRGEND
CF35 A5 23          490         LDA TEMP+1      ; MUTATO A PROGRAMVEGEN
CF37 69 00          500         ADC #0
CF39 85 2E          510         STA PRGEND+1
CF3B 4C 63 A6       520         JMP CLR         ; CLR ES KESZ
CF3E               65535        .END

CLR    =A663  NULL   =CF0A  PRGEND=002D   PRGSRT=002B  TDREIO=CF22  TEMP   =0022
TO     =CF20
```


A gépi kódú program BASIC betöltője:

P 19

```
100 FOR I=52992 TO 53053
110 READ X:POKE I,X :S=S+X :NEXT
120 DATA 165, 43,164, 44,133, 34,132, 35,160,  3,200,177
130 DATA  34,200,251,200,152, 24,101, 34,160,  0,145, 43
140 DATA 165, 35,105,  0,200,145, 43,136,162,  3,230, 34
150 DATA 208,  2,230, 35,177, 34,200,244,202,208,243,165
160 DATA  34,105,  2,133, 45,165,35 ,105, 0 ,133, 46, 76
170 DATA  99,166
180 IF S<> 7000 THEN PRINT"HIBA AZ ADATOKBAN!!!" : END
190 PRINT"RENDBEN !"
```

c) Formázott kiíratás – a PRINT USING

A C 64-es BASIC nyelvéből hiányzik az olyan utasítás, amellyel számokat, illetve számsorozatokat egymás alá, a tizedespontra igazítva kiírathatnánk a képernyőre vagy a nyomtatóra.
Most bemutatunk egy gépi kódú rutint, amelyet a BASIC programból SYS utasítással hívhatunk meg úgy, hogy a SYS utasítás paramétereként megadhatjuk a kiírandó számot.
A rutin ($C900 – $C90C) a számot betölti a lebegőpontos akkumulátorba, majd meghívja azt az interpreter rutint, amely a számot szöveggé konvertálja.
A kapott füzért (a $0100-as címen) a rutin a kívánt formára alakítja. A formátum paramétereit előre megadott tárcímekre, POKE utasításokkal kell beírni.

POKE 51612,X	0=egész szám, 1=a szám tizedespontot tartalmaz
POKE 51613,N	a tizedesjegyek száma
POKE 51615,ASC(" ")	a szám előtti karakter
POKE 51616,ASC(" ")	kitöltő karakterek a szám előtt

Tegyük fel, hogy egy valós számot maximum 10 helyre szeretnenk kiírni, amelyből egyet a tizedespont, kettőt pedig a tizedesjegyek foglalnak el.
A szám előtti helyeket üres karakterekkel töltjük ki. A rutin hívása ekkor:

```
SYS (51456) 100,
```

a végrehajtás eredménye pedig:

```
100.00
```

A POKE 51614,3 : POKE 51615,ASC("*"): POKE 51549,ASC("$") utasítások végrehajtása után a kiíratáskép:

```
**$100.000
```

Nyomtatáskor a következő parancsokra van szükség:

```
OPEN 1,4 : CMD 1
SYS (51456) X
PRINT#1 : CLOSE 1
```

A rutin listája:

P20.ASS

```
1000            100       ; PRINT USING
1000            110       ;
ADBA            120 FRMNUM = $ADBA
BDDD            130 ASCII  = $BDDD
AB1E            140 OUT    = $AB1E
1000            150       ;
C900            160       *=$C900
C900            170       ;
C900 20 8A AD   180       JSR FRMNUM      ; NUMERIKUS KIFEJEZES BEOLVASASA
C903 20 DD BD   190       JSR ASCII       ; ASCII-BE ATALAKITANI
C906 20 0D C9   200       JSR USING
C909 20 1E AB   210       JSR OUT         ; A FUZER KIIRASA
C90C 60         220       RTS
C90D A9 45      230 USING LDA #"E"
C90F 20 8E C9   240       JSR CHECK       ; EXPONENCIALIS ABRAZOLAS ELLENORZES
E
C912 B0 59      250       BCS SETPTR
C914 AD 9C C9   260       LDA DECINT      ; TIZEDES, VAGY EGESZ SZAM FLAG
C917 F0 59      270       BEQ INTEGR
C919 AD 02 01   280       LDA $102
C91C D0 0B      290       BNE L1
C91E AC 9D C9   300       LDY LENGHT      ; HUSSZUSAG - 1
C921 A9 30      310       LDA #"0"
C923 99 02 01   320 L2    STA $102,Y   ; PUFFER FELTOLTESE NULLAKKAL
C926 88         330       DEY
C927 D0 FA      340       BNE L2
C929 A9 2E      350 L1    LDA #"."
C92B 20 8E C9   360       JSR CHECK
C92E A8         370       TAY
C92F 90 02      380       BCC *+4
C931 A0 30      390       LDY #"0"
C933 A9 00      400       LDA #0
C935 20 8E C9   410       JSR CHECK
C938 98         420       TYA
C939 9D 00 01   430       STA $100,X
C93C A9 2E      440       LDA #"."
C93E 20 8E C9   450       JSR CHECK
C941 AC 9E C9   460       LDY DECLEN      ; A TIZEDESHELYEK SZAMA
C944 E8         470 L3    INX
C945 88         480       DEY
C946 D0 FC      490       BNE L3
C948 EC 9D C9   500 L8    CPX LENGHT
C94B B0 28      510       BCS SETPTR
C94D AC 9D C9   520       LDY LENGHT
C950 A9 00      530       LDA #0
C952 99 01 01   540       STA $101,Y
C955 BD 00 01   550 L6    LDA $100,X
C958 C9 20      560       CMP #" "        ; VEZETO KARAKTER
C95A D0 02      570       BNE L5
C95C A9 20      580       LDA #" "
C95E 99 00 01   590 L5    STA $100,Y
C961 CA         600       DEX
C962 10 06      610       BPL L4
C964 AD 9F C9   620       LDA FILLER
C967 88         630       DEY
C968 10 F4      640       BPL L5
C96A 88         650 L4    DEY
C96B 10 E8      660       BPL L6
C96D A9 00      670 SETPTR LDA #0  ; MUTATO A PUFFERRA
C96F A0 01      680       LDY #1
C971 60         690       RTS
C972 A9 00      700 INTEGR LDA #0
```



```
C974 20 8E C9   710       JSR CHECK
C977 90 F4      720       BCC SETPTR
C979 8A         730       TXA
C97A A8         740       TAY
C97B AD 02 01   750       LDA $102
C97E F0 09      760       BEQ L7
C980 A9 2E      770       LDA #"."
C982 20 8E C9   780       JSR CHECK
C985 90 02      790       BCC L7
C987 8A         800       TXA
C988 A8         810       TAY
C989 98         820 L7    TYA
C98A AA         830       TAX
C98B CA         840       DEX
C98C 10 BA      850       BPL L8
C98E A2 00      860 CHECK LDX #0
C990 DD 00 01   870 L9    CMP $100,X
C993 F0 06      880       BEQ L10
C995 E8         890       INX
C996 E0 0C      900       CPX #12
C998 D0 F6      910       BNE L9
C99A 18         920       CLC
C99B 60         930 L10   RTS
C99C 01         940 DECINT .BYTE 1     ; DECIMALIS
C99D 09         950 LENGHT .BYTE 9     ; HOSSZUSAG = 9
C99E 02         960 DECLEN .BYTE 2     ; TIZEDESHELYEK SZAMA = 2
C99F A0         970 FILLER .BYTE " "   ; KITOLTO KARAKTER
C95D            980 LEAD   =L5-1         ; VEZETO KARAKTER
C9A0          65535       .END
```

```
ASCII =BDDD  CHECK =C98E  DECINT=C99C  DECLEN=C99E  FILLER=C99F  FRMNUM=ADBA
INTEGR=C972  L1    =C929  L10   =C99B  L2    =C923  L3    =C944  L4    =C96A
L5    =C95E  L6    =C955  L7    =C989  L8    =C948  L9    =C990  LEAD  =C95D
LENGHT=C99D  OUT   =AB1E  SETPTR=C96D  USING =C90D
```

A BASIC betöltő:

P21

```
100 FOR I=51456 TO 51615
110 READ X:POKE I,X : S=S+X :NEXT
120 DATA  32,138,173, 32,221,189, 32, 13,201, 32, 30,171
130 DATA  96,169, 69, 32,142,201,176, 89,173,156,201,240
140 DATA  89,173,  2,  1,208, 11,172,157,201,169, 48,153
150 DATA   2,  1,136,208,250,169, 46, 32,142,201,168,144
160 DATA   2,160, 48,169,  0, 32,142,201,152,157,  0,  1
170 DATA 169, 46, 32,142,201,172,158,201,232,136,208,252
180 DATA 236,157,201,176, 32,172,157,201,169,  0,153,  1
190 DATA   1,189,  0 ,1  ,201, 32,208,2  ,169, 32,153,  0
200 DATA   1,202, 16,  6,173,159,201,136,16 ,244,136, 16
210 DATA 232,169,  0,160,1  , 96,169,  0, 32,142,201,144
220 DATA 244,138,168,173,  2,  1,240,  9,169, 46, 32,142
230 DATA 201,144,  2,138,168,152,170,202, 16,186,162,  0
240 DATA 221,  0,  1,240, 6 ,232,224, 12,208,246, 24, 96
250 DATA   0,  9,  2, 32
260 IF S<>18657 THEN PRINT"HIBA AZ ADATOKBAN!!!":END
270 PRINT"RENDBEN !!"
```

5.3.3 SAJÁT FEJLESZTÉSŰ MATEMATIKAI ELJÁRÁSOK

A C 64-es gép BASIC nyelve nagyon sok hasznos matematikai függvényt tartalmaz. Mégis megeshet, hogy valakinek olyan speciális eljárásra van szüksége, amely nem szerepel az utasításkészletben.
Abban az esetben, ha az eljárást nem egyetlen programban, hanem többször is szeretnénk használni, érdemes rutint készíteni belőle. A saját matematikai függvények készítésének programozástechnikai eszköze a USR utasítás.
A matematikai függvényekhez hasonlóan a USR függvényt a BASIC programban bárhol, pl. egy kifejezésben vagy egy PRINT utasításban is elhelyezhetjük.
Az interpretert POKE utasításokkal tájékoztathatjuk arról, hogy a tárban hol található a saját magunk által írt gépi kódú programrész. A USR függvény paramétere is lehet konstans, változó, vagy tetszőleges aritmetikai kifejezés. A paramétert és a számítás eredményét az interpreter egyaránt a FAC 1 lebegőpontos akkumulátorban tárolja. A rutin végrehajtása után a BASIC innen kapja vissza a függvény értékét.
Nézzünk néhány példát a USR függvény alkalmazására.

5.3.4 NÉGYZETGYÖKVONÁS, ÖSSZEGZÉS, SZORZÁS – SQR, SUM, PROD

A BASIC interpreter természetesen tartalmazza a négyzetgyökvonást, azonban a hatványozáshoz hasonlóan ezt a műveletet is logaritmussal végzi el, amely együtt jár a LOG és az EXP függvényhívásokkal.
A négyzetgyökvonást egyszerűbb és gyorsabb algoritmusokkal is elvégezhetjük, amelynek eredménye még pontosabb is, mintha logaritmusokkal számolnánk. Az algoritmus lépésenkénti iterációval közelíti a szám négyzetgyökét. Legyen a kezdőérték maga a szám, amelyből gyököt akarunk vonni, az iterációs képlet* pedig a következő:

$$X(n + 1) = (X(n) + a/X(n))/2$$

ahol "a" a szám, amelyből gyököt vonunk, X(n) a régi, X(n + 1) pedig az új közelítő érték. A gyakorlati tapasztalat azt mutatja, hogy négy lépésnél tovább nem érdemes elmenni, a negyedik lépés után a közelítő érték nem változik jelentős mértékben.
A négyzetgyökvonó függvény:

P22.ASS

```
BC2B                100 SIGN    = $BC2B
B248                110 ILL     = $B248
BBC7                120 FACA4   = $BBC7
0061                130 EXP     = $61
0067                140 COUNT   = $67
BBCA                150 FACA3   = $BBCA
BB0F                160 DIV     = $BB0F
B867                170 PLUS    = $B867
1000                180         ;
C800                190         *=$C800
C800                200         ;
C800 20 2B BC       210         JSR SIGN          ; ELOJEL TESZTELES
C803 F0 34          220         BEQ ENDE          ; HA AZ ERTEK=0, AKKOR KESZ
C805 10 03          230         BPL OK            ; POZITIV - A GYOKVONAS ELVEGEZHE
TO
C807 4C 48 B2       240         JMP ILL           ; NEGATIV, 'ILLEGAL QUANTITY
C80A 20 C7 BB       250 OK      JSR FACA4       ; FAC ATVITELE AKKU#4-BE
```

*Ez a négyzetgyökvonás jól ismert NEWTON-féle iterációs képlete. (a ford. megjegyzése)


```
C80D A5 61          260         LDA EXP
C80F 38             270         SEC
C810 E9 81          280         SBC #$81          ; KITEVO NORMALIZALAS
C812 08             290         PHP
C813 4A             300         LSR A             ; KITEVO FELEZES
C814 18             310         CLC
C815 69 01          320         ADC #1
C817 28             330         PLP
C818 90 02          340         BCC S1
C81A 69 7F          350         ADC #$7F          ; KITEVO UJRAELOALLITAS
C81C 85 61          360 S1      STA EXP
C81E A9 04          370         LDA #4            ; 4 ITERACIO
C820 85 67          380         STA COUNT
C822 20 CA BB       390 ITER    JSR FACA3     ; FAC AZ AKKU#3-BA
C825 A9 5C          400         LDA #$5C
C827 A0 00          410         LDY #$00          ; MUTATO AZ AKKU#4-RE
C829 20 0F BB       420         JSR DIV
C82C A9 57          430         LDA #$57
C82E A0 00          440         LDY #$00          ; MUTATO AZ AKKU#3-RA
C830 20 67 B8       450         JSR PLUS          ; HOZZAADAS FAC-HOZ
C833 C6 61          460         DEC EXP           ; FAC/2/(KITEVO-1)
C835 C6 67          470         DEC COUNT         ; SZAMLALO CSOKKENTESE
C837 D0 E9          480         BNE ITER          ; MEG EGY ITERACIO
C839 60             490 ENDE    RTS             ; KESZ
C83A              65535         .END

COUNT =0067  DIV   =BB0F  ENDE  =C839  EXP   =0061  FACA3 =BBCA  FACA4 =BBC7
ILL   =B248  ITER  =C822  OK    =C80A  PLUS  =B867  S1    =C81C  SIGN  =BC2B
```

Hívás előtt tudatnunk kell az interpreterrel a rutin tárbeli kezdőcímét, alsó és felső byte-ra bontva.
Az interpreter a két értéket a $311(785), illetve a $312(786) tárcímeken keresi. A megfelelő POKE utasítás tehát:

POKE 785,0 : POKE 785,12 * 16 + 8

A két POKE utasítást beépítettük a betöltőprogramba:

P23

```
100 FOR I=51200 TO,51257
110 READ X :POKEI,X:S=S+X:NEXT
120 DATA  32, 43,188,240, 52, 16,  3, 76,72 ,178, 32,199
130 DATA 187,165, 97, 56,233,129,  8, 74, 24,105,  1, 40
140 DATA 144,  2,105,127,133, 97,169,  4,133,103, 32,202
150 DATA 187,169, 92,160,  0, 32, 15,187,169, 87,160, 0
160 DATA  32,103,184,198,97 ,198,103,208,233, 96
170 IF S<>6211 THENPRINT" HIBA AZ ADATOKBAN!!!":END
180 POKE 785,0:POKE786,200:PRINT"RENDBEN!"
```

A betöltőprogramot lefuttatva a USR (A) utasítással az A számból az új eljárással vonhatunk négyzetgyököt.
Ha az interpreter négyzetgyökvonó rutinjának és az új eljárásnak a végrehajtási idejét összehasonlítjuk, látni fogjuk, hogy az új eljárás kb. 4-szer olyan gyorsan végzi el a műveletet, mint az interpreter.
Térjünk át egy másik feladatra, amely egy kicsit bonyolultabb az előzőnél.
Majdnem minden program tartalmaz összegző ciklust. Az adatokat többnyire egy tömbben tároljuk. Az összegző BASIC programrész néhány utasításból áll:

P24

```
10 DIM A(1000)
20 REM      .         .
30 REM      .         .
40 REM      .         .
50 REM      .         .
60 REM      .         .
70 REM     ADATOK KISZAMITASA
71 REM
80 REM      VAGY BEOLVASASA
81 REM      .         .
83 REM      .         .
84 REM      .         .
90 REM      .         .
100 S=0
110 FOR I=0 TO 1000 : S=S+A(I) : NEXT
120 PRINT S
```

A USR függvény most sem haszontalan. Egyrészt a fentieket a következő sorral helyettesíthetjük:

100 S= USR(A),

másrészt az összeadás végrehajtása is gyorsabb.
A USR függvény paramétere az összeadandókat tartalmazó tömb neve.
Ráadásul az alábbi szubrutinban mindössze két utasítást kell megváltoztatnunk ahhoz, hogy az eredmény a számoknak ne az összegét, hanem a szorzatát adja.

P25.ASS

```
ADBD                100 NUMTST = $ADBD
002F                110 ARRTAB = $2F
005F                120 TEMP   = $5F
0032                130 AREND  = $32
0045                140 VARNAM = $45
A445                150 ERROUT = $A445
0024                160 STORE  = $24
B196                170 SETARR = $B196
000E                180 INTFLG = $0E
BBA2                190 MEMAC1 = $BBA2
B867                200 MEMPLS = $B867
BC0C                210 A1TOA2 = $BC0C
B86F                220 AKPLUS = $B86F
B391                230 INTFLT = $B391
0031                240 ARREND = $31
1000                250        ;
033C                260        *=$033C
033C                270        ;
033C 20 BD AD       280        JSR NUMTST     ; NUMERIKUS VALTOZO "?"
033F A6 2F          290        LDX ARRTAB
0341 A5 30          300        LDA ARRTAB+1   ; MUTATO A TOMB KEZDETERE
0343 86 5F          310 S3     STX TEMP
0345 85 60          320        STA TEMP+1     ; CIKLUSVALTOZO
0347 C5 32          330        CMP ARREND+1
0349 D0 04          340        BNE S1
034B E4 31          350        CPX ARREND     ; TOMBOK VEGE MUTATO "?"
034D F0 1D          360        BEQ NOTFND
034F A0 00          370 S1     LDY #0
0351 B1 5F          380        LDA (TEMP),Y ; A NEV ELSO BETUJE
0353 C8             390        INY
0354 C5 45          400        CMP VARNAM     ; KERESETT NEVVEL OSSZEHASONLITJA
0356 D0 06          410        BNE S2         ; NEM - KOVETKEZO TOMB VIZSGALATA
```



```
0358 A5 46          420        LDA VARNAM+1 ; VALTOZO MASODIK KARAKTERE
035A D1 5F          430        CMP (TEMP),Y ; OSSZEHASONLITANI
035C F0 17          440        BEQ FOUND    ; RENDBEN
035E C8             450 S2     INY
035F B1 5F          460        LDA (TEMP),Y
0361 18             470        CLC
0362 65 5F          480        ADC TEMP       ; OFFSET HOZZAADASA
0364 AA             490        TAX
0365 C8             500        INY
0366 B1 5F          510        LDA (TEMP),Y
0368 65 60          520        ADC TEMP+1
036A 90 D7          530        BCC S3
036C A2 E2          540 NOTFND LDX #TAB<
036E 86 22          550        STX $22        ; HIBAJELZES MUTATO
0370 A9 03          560        LDA #TAB>
0372 4C 45 A4       570        JMP ERROUT     ; HIBAJELZES KIIRASA
0375 C8             580 FOUND  INY
0376 B1 5F          590        LDA (TEMP),Y
0378 18             600        CLC
0379 65 5F          610        ADC TEMP
037B 85 24          620        STA STORE
037D C8             630        INY
037E B1 5F          640        LDA (TEMP),Y
0380 65 60          650        ADC TEMP+1
0382 85 25          660        STA STORE+1
0384 C8             670        INY
0385 B1 5F          680        LDA (TEMP),Y ; INDEXEK SZAMA
0387 20 96 B1       690        JSR SETARR   ; MUTATO AZ ELSO TOMBELEMRE
038A 85 5F          700        STA TEMP
038C 84 60          710        STY TEMP+1
038E 24 0E          720        BIT INTFLG   ; INTEGERFLAG VIZSGALATA
0390 30 1F          730        BMI INTEGR
0392 20 A2 BB       740        JSR MEMAC1   ; ELEM A FAC-BA
0395 18             750        CLC
0396 90 04          760        BCC LOOP     ; UGRAS A CIKLUSBA
0398 20 67 B8       770 S5     JSR MEMPLS; VALTOZO + FAC
039B 18             780        CLC
039C A5 5F          790 LOOP   LDA TEMP
039E 69 05          800        ADC #5         ; MUTATO A KOVETKEZO ELEMRE
03A0 85 5F          810        STA TEMP
03A2 90 02          820        BCC S4
03A4 E6 60          830        INC TEMP+1
03A6 A4 60          840 S4     LDY TEMP+1
03A8 C5 24          850        CMP STORE      ; ARRAY VEGE "?"
03AA 90 EC          860        BCC S5
03AC C4 25          870        CPY STORE+1
03AE 90 E8          880        BCC S5
03B0 60             890 READY  RTS     ; IGEN, KESZ
03B1 20 D5 03       900 INTEGR JSR INTAKK ; INTEGERVALTOZO A FAC-BA
03B4 20 0C BC       910 S6     JSR A1TOA2; FAC + ARG
03B7 18             920        CLC
03B8 A5 5F          930        LDA TEMP
03BA 69 02          940        ADC #2         ; MUTATO A KOVETKEZO TOMBELEMRE
03BC 85 5F          950        STA TEMP
03BE 90 02          960        BCC S7
03C0 E6 60          970        INC TEMP+1
03C2 C5 24          980 S7     CMP STORE ; TOMB-TERULET VEGE "?"
03C4 90 06          990        BCC S8
03C6 A5 60         1000        LDA TEMP+1
03C8 C5 25         1010        CMP STORE+1
03CA B0 E4         1020        BCS READY
03CC 20 D5 03      1030 S8     JSR INTAKK; INTEGERVALTOZO A FAC-BOL
03CF 20 6F B8      1040        JSR AKPLUS   ; FAC+ARG
03D2 4C B4 03      1050        JMP S6
03D5 A0 00         1060 INTAKK LDY #0
03D7 B1 5F         1070        LDA (TEMP),Y
```

```
03D9 AA          1080        TAX
03DA C8          1090        INY
03DB B1 5F       1100        LDA (TEMP),Y
03DD A8          1110        TAY
03DE 8A          1120        TXA
03DF 4C 91 B3    1130        JMP INTFLT
03E2 41 52 52    1140 TAB    .TEXT"ARRAY NOT FOUN"
03F0 C4          1150        .BYTE $C4     ; SHIFT+D
03F1            65535        .END
```

```
A1TOA2=BC0C  AKPLUS=B86F  AREND =0032  ARREND=0031  ARRTAB=002F  ERROUT=A445
FOUND =0375  INTAKK=03D5  INTEGR=03B1  INTFLG=000E  INTFLT=B391  LOOP  =039C
MEMAC1=BBA2  MEMPLS=B867  NOTFND=036C  NUMTST=AD8D  READY =03B0  S1    =034F
S2    =035E  S3    =0343  S4    =03A6  S5    =0398  S6    =03B4  S7    =03C2
S8    =03CC  SETARR=B196  STORE =0024  TAB   =03E2  TEMP  =005F  VARNAM=0045
```

A betöltőprogram:

```
P2b

100 FOR I=828 TO 1008
110 READ X:POKE I,X :S=S+X:NEXT
120 DATA  32,141,173,166, 47,165, 48,134, 95,133, 96,197
130 DATA  50,208,  4,228, 49,240, 29,160,  0,177, 95,200
140 DATA 197, 69,208,  6,165, 70,209, 95,240, 23,200,177
150 DATA  95, 24,101, 95,170,200,177, 95,101, 96,144,215
160 DATA 162,226,134, 34,169,  3, 76, 69,164,200,177, 95
170 DATA  24,101, 95,133,36 ,200,177, 95,101, 96,133, 37
180 DATA 200,177, 95, 32,150,177,133, 95,132, 96, 36, 14
190 DATA  48, 31, 32,162,187, 24,144,  4, 32,103,184, 24
200 DATA 165, 95,105,  5,133, 95,144,  2,230, 96,164, 96
210 DATA 197, 36,144,236,196, 37,144,232, 96, 32,213,  3
220 DATA  32, 12,188,24 ,165, 95,105,  2,133, 95,144,  2
230 DATA 230, 96,197, 36,144,  6,165, 96,197, 37,176,228
240 DATA  32,213,  3, 32,111,184, 76,180,  3,160,  0,177
250 DATA  95,170,200,177, 95,168,138, 76,145,179, 65, 82
260 DATA  82, 65, 89, 32, 78, 79, 84, 32, 70, 79, 85, 78
270 DATA 196
280 IF S<>20399 THEN PRINT" HIBA AZ ADATOKBAN!":END
290 POKE 785,3*16+12:POKE 786,3 : PRINT"RENDBEN!"
```

A függvény argumentuma valós és egész típusú tömb egyaránt lehet.
Ha a rutin nem találja meg a változóterületen az adott nevű tömböt, akkor kiírja az „array not found error" hibaüzenetet. A szorzást pontosan ugyanolyan logikával végezhetjük el, mint az összeadást.
A szükséges változtatás mindössze annyi, hogy a $0398-as címtől kezdve a 20 28 BA, a $03CF címtől kezdve pedig 20 2B BA értékeket kell beírni.
A tárcímek tartalmát az alábbi POKE utasításokkal is megváltoztathatjuk:

POKE 921,40: POKE 922,186: POKE 976,43: POKE 977,186

A rutint a kazettapufferba helyeztük, így a kezdőcímet ismét át kell adni az interpreternek:

POKE 785,3 * 16 + 12: POKE 786,3

Meglepő eredményt kapunk, ha az összeadó BASIC ciklus és a fenti USR rutin végrehajtási sebességét összehasonlítjuk!

5.3.5 TÖBBPARAMÉTERES FELADATOK

A USR függvény óriási hátránya, hogy csak egy paraméter szerepelhet benne, holott nagyon sok esetben ennél többre lenne szükség. Ahhoz például, hogy a kurzort a képernyő adott helyére mozgassuk, meg kellene adnunk a sor- és oszlopkoordinátákat. A SYS utasításban egyáltalán nem adhatunk meg paramétert, felismerésekor az interpreter csak egy tárcímet fogad el, amely megadja a rutin tárbeli helyét. Az interpreter az 5.2 fejezetben említett FRMEVL rutinja több paraméterrel dolgozik. A rutinnak egymástól vesszővel, illetve zárójellel elválasztva tetszőleges számú paramétert átadhatunk. A kifejezéseket kiértékelő rutinnak nagyon sok beugrási pontja van, egyik pl. a zárójelek közötti paramétereket elemző programrész, amely megkeresi a paramétereket elválasztó vesszőt.
Egy másik programrész megvizsgálja, hogy egy paraméter a megengedett tartományba esik-e, stb. Az alábbiakban felsoroljuk a legfontosabb rutinokat, de az érdeklődők teljes képet kaphatnak a 8. fejezetben közölt ROM listából.

Cím	*Leírás*
AD8A	az argumentum kiértékelése és numerikus ellenőrzés
AD8D	numerikus ellenőrzés
AD8F	füzér-ellenőrzés
AD9E	az argumentum kiértékelése, tetszőleges kifejezés
AEF1	a zárójelen belüli argumentum kiértékelése
AEF7	bezárójel vizsgálata
AEFA	nyitójel vizsgálata
AEFD	vessző vizsgálata
B79E	egy byte betöltése (0–255 közötti érték) az X regiszterbe
0073	a következő karakter beolvasása a BASIC szövegből.

Ha az argumentumként megadott kifejezés vagy változó értéke nem esik a megengedett tartományba, az interpreter az „illegal quantity", rossz adattípus esetén a „type mismatch" hibaüzenetet írja ki a képernyőre.
Az adattípusok közötti átalakítás rendszerrutinjai a következők:

Cím	*Leírás*
B1BF	a FAC átalakítása egész típusúvá
B395	az A/X-ben tárolt 16 bites egész átalakítása valóssá
B3A2	az Y-ban tárolt szám átalakítása valóssá
BC9B	A FAC átalakítása 16 bites számmá
BCF3	számjegyfüzér átalakítása valós számmá
BDDD	a FAC átalakítása számjegyfüzérré

A több paraméteres feladatokat az általánosított SYS utasítással oldhatjuk meg. Példaként bemutatunk egy olyan rutint, amely a kurzort a képernyő adott helyére mozgatja. Paraméterként meg kell adni a HOME (bal felső sarok) pozícióhoz képest a vízszintes, illetve függőleges irányú elmozdulást. Az általánosított SYS utasítás:

SYS PR, oszlop, sor, változólista

ahol a PR a gépi kódú rutin decimális kezdőcíme, a következő két paraméter a kurzorpozíció vízszintes és függőleges irányú koordinátája, a változólista pedig a kírandó változók sora egymástól vesszővel elválasztva, ugyanúgy, mint ahogyan azok a PRINT utasításban szerepelnének.

P27.ASS

```
AEFD              100 CKCOM  = $AEFD
B79E              110 GETBYT = $B79E
FFF0              120 CURSOR = $FFF0
AAA4              130 PRINT  = $AAA4
1000              140        ;
C000              150        *=$C000
C000              160        ;
C000 20 FD AE     170        JSR CKCOM    ; VESSZO KERESESE
C003 20 9E B7     180        JSR GETBYT   ; OSZLOP ERTEKE AZ X-BE
C006 8A           190        TXA
C007 48           200        PHA          ; OSZLOPSZAM TAROLASA
C008 20 FD AE     210        JSR CKCOM    ; VESSZO KERESESE
C00B 20 9E B7     220        JSR GETBYT   ; SOR ERTEK BETOLTES
C00E 68           230        PLA
C00F A8           240        TAY          ; OSZLOPSZAM AZ Y-BA
C010 18           250        CLC
C011 20 F0 FF     260        JSR CURSOR   ; A KURZOR BEALLITASA
C014 20 FD AE     270        JSR CKCOM    ; VESSZO ELLENORZESE
C017 4C A4 AA     280        JMP PRINT    ; PRINT PARANCSRA
C01A            65535        .END

CKCOM =AEFD  CURSOR=FFF0  GETBYT=B79E  PRINT =AAA4
```

A betöltőprogram:

P28

```
100 FOR I=49152 TO 49177
110 READ X :POKE I,X : S=S+X: NEXT
120 DATA  32,253,174, 32,158,183,138, 72, 32,253,174, 32
130 DATA 158,183,104,168, 24, 32,240,255, 32,253,174, 76
140 DATA 164,170
150 IF S<>3566 THEN PRINT"HIBA AZ ADATOKBAN!":END
160 PRINT"RENDBEN !!"
```

Az alábbi két BASIC sor meghívja a $C000 címen elhelyezett rutint, amely a képernyő 20. sorának 24. oszlopára kiírja a PELDA szöveget.

P30

```
10 PR = 12*4096
100 SYS PR, 24, 20, "PELDA"
```


6. FEJEZET
GÉPI KÓDÚ PROGRAMOZÁS A C 64-ESEN

6.1 Amit a monitorprogramokról tudni kell

A gépi kódú programozáshoz elengedhetetlenül szükség van egy monitorprogramra. A megnevezésbeli „monitor" kifejezés ebben az esetben nem a tv készülékre utal. A számítógépes gyakorlatban monitoron, vagy monitorprogramon egy olyan segédprogramot értünk, amellyel közvetlenül hozzáférhetünk a tár tartalmához. A monitorral a regiszterek és a tárcímek tartalmát megjeleníthetjük a képernyőn, tetszőlegesen megváltoztathatjuk őket, gépi kódú programot futtathatunk, lemezen tárolhatunk, disassemblálhatunk, illetve javíthatunk.
Ebben a fejezetben ismertetjük azt az utasításkészletet, amellyel minden monitorprogram rendelkezik.
A programot, miután szalagról vagy lemezről betöltöttük, SYS utasítással indíthatjuk el.
Az általunk használt monitor kezdőcíme 12*4096, indítása tehát a következő:

SYS 12*4096

ahol a decimális cím megfelel a $C000 hexadecimális értéknek.
Hívás után a program a PROMT üzenettel jelentkezik be, ami azt jelenti, hogy a gép mostantól csak a monitor speciális utasításait tudja értelmezni, a BASIC utasításokat nem.
A képernyőn a következő látható:

```
C*
     PC   IRQ   SR   AC   XR   YR   XP
>;   E145 EA13  31   32   AC   34   F8
```

Az első sor felirata a 6510-es processzor fontos regisztereinek megnevezésére utal:

PC – a programszámláló
Ez a regiszter tartalmazza a soronkövetkező végrehajtandó utasítás címét. Jelen esetben a következő utasítás címe: E145

IRQ – a megszakításregiszter (interrupt)

Az IRQ vezérli a programbeli elágazásokat, megszakításokat.

SR – az állapotregiszter (status)
Az SR regiszter tartalmazza a kapcsolók (flagek) aktuális értékét.

A kapcsolók jelentése:
b7 – a negatív kapcsoló (értéke 1, ha az eredmény negatív)
b6 – a túlcsordulás kapcsoló (az aritmetikai számítás eredménye nagy)
b5 – értéke mindig 1
b4 – a programmegszakítást jelző kapcsoló (értéke 1, ha a program futása megszakad)
b3 – a decimális kapcsoló (jelzi az aktuális aritmetikát: BCD/HEX)
b2 – a megszakítási kapcsoló (a belső megszakításokat vezérli)
b1 – a nulla kapcsoló (értéke 1, ha az eredmény 0)
b0 – átviteli kapcsoló (értéke 1, ha a műveletvégzés közben átvitel keletkezik)

A legtöbb kézikönyv a kapcsolókra az alábbi jelölést használja:

b7	b6	b5	b4	b3	b2	b1	b0
N	V	1	B	D	I	Z	C

AC – az akkumulátor
Az akkumulátor a logikai és az aritmetikai műveletek legfontosabb regisztere. A művelet egyik operandusa és eredménye ebben a regiszterben található.

XR – az X regiszter
Az X regiszter a processzor munkaregisztere, a folyamatosan érkező adatokat fogadja.

YR – az Y regiszter
Feladata azonos az X regiszter feladatával.

SP – a veremregiszter
Az SP regiszter tartalmazza annak a veremnek a kezdőcímét, ahol a szubrutinok visszatérési címei találhatóak.

A monitorral a processzor regisztereinek tartalmát folyamatosan ellenőrizhetjük és módosíthatjuk. Módosítás közben a számokat hexadecimális alakban kell megadni (a 255-öt például FF-ként).
A módosítás menete:
A kurzort a kurzormozgató billentyűkkel a kívánt helyre visszük, beírjuk az új értéket, majd megnyomjuk a RETURN billentyűt.
A tárcímek módosításához először az M XXXX YYYY utasítással ki kell iratnunk a képernyőre a kívánt tárterület tartalmát, ahol az XXXX és az YYYY a tárterület kezdő- és végcíme, egy-egy négyjegyű hexadecimális szám.
Ha akkora területet jelölünk ki, amely egyszerre nem fér el a képernyőn, a monitor soronként tolja felfelé a képet, amíg el nem érkezik a végcímhez.
Megjelenítés után a módosítás menete azonos a fentivel.
Nézzünk erre egy példát:

```
>M  C000  C010
>:  C000  A9 10 8D 16 03 A9 C0 8D
>:  C008  17 03 A9 43 85 97 D0 16
>:  C010  A9 42 85 97 D8 4A 68 8D
```

A megjelenítés másik formája a disassemblálás (visszafordítás).
A disassemblálás előnye, hogy az áttekinthetetlen hexadecimális kódok könnyen érthető utasítássorozatokká alakulnak át. Az assembler utasításszavak számítógépes megnevezéseként gyakran találkozhatunk a *mnemonik* kifejezéssel.
Ha a valósághoz hűek akarunk maradni, meg kell említenünk a disassemblálás hátrányát is. Előfordulhat, hogy a programban olyan szöveg is található, amelynek semmi köze nincs az assembler utasításszavakhoz. A visszafordító (disassembler) megpróbálja a szöveget utasításkódként értelmezni és visszaalakítani, és ha ez nem sikerül, az utasításszó helyére kérdőjelet ír. Sokkal félrevezetőbb azonban, ha véletlen egyezés folytán ott is utasításszó jelenik meg, ahol valójában kérdőjelnek kellene állnia. A gyakorlottabb programozók általában különösebb nehézség nélkül megkülönböztetik az igazi és hamis utasításszavakat.
Az elkészült gépi kódú programot a

G XXXX

utasítással futtathatjuk. A G betű az angol GO TO kifejezés rövidítése, az XXXX pedig vagy egy gépi kódú program kezdőcíme, vagy egy beugrási cím.
Ha a gépi kódú program BRK utasítást tartalmaz, ami megfelel a BASIC STOP utasításnak, a program futása megszakad és a monitor a B* üzenettel visszajelentkezik.
A BASIC programok tesztelésénél alkalmazott módszer gépi kódú programoknál is hasznos. Időlegesen a program lényeges csomópontjaiba érdemes BRK utasításokat írni.
A gépi kódú programok tárolására szolgál a monitor

S: "NÉV", XX, YYYY, ZZZZ

utasítása. A program nevén kívül az utasításbeli XX az egységszám (kazettás egységre: 01, lemezegységre: 08), az YYYY és a ZZZZ pedig a program kezdő és végcíme hexadecimális alakban. A RETURN billentyű hatására a monitor a gépi kódú programot a megfelelő egységen tárolja.
A visszatöltő utasítás emlékeztet a BASIC-beli megfelelőjére:

L "NÉV", XX

Az L betű éppen a LOAD utasításszó kezdőbetűje.
Vannak olyan monitorok, amelyek betöltésnél paraméterként tetszőleges tárcímet is elfogadnak, és a programot az adott tárterületre töltik be. Ezzel a lehetőséggel azonban óvatosan kell bánni, mert a program csak akkor lesz az eredetitől eltérő tárterületen futtatható, ha az összes abszolút cím-hivatkozásokat kijavítjuk.
A munka végeztével a monitorprogramból az X paranccsal térhetünk vissza a BASIC-be.
Ha ügyelünk arra, hogy a gépi kódú program és a BASIC program által elfoglalt területek között ne legyen átfedés, akkor a monitorprogram és a BASIC felváltott használata semmilyen problémát nem okoz, egyik sem zavarja a másik munkáját.
Egy hasznos tanács! A monitor betöltése után, indítás előtt adjuk ki a NEW parancsot. Ellenkező esetben előfordulhat, hogy az első BASIC sor begépelésekor furcsa hibaüzenetet kapunk.
Természetesen monitor nélkül is dolgozhat bárki gépi kódú programokkal és írhat gépi kódú programot. Gondoljunk azonban arra, hogy BASIC-ben az egyedüli út a tárcímek tartalmának felülírására a POKE utasítás. Ez azt jelenti, hogy a gépi kódú program megírása azonos egy POKE utasításokból álló BASIC program megírásával, amelyek argumentumai a tárcímek, illetve az utasításkódok.
Mivel a POKE utasítás argumentuma csak decimális szám lehet, előzetesen minden hexadecimális értéket át kellene alakítanunk decimálissá, jobb esetben egy átalakító rutin segítségével. Ugyanez a bonyodalom vonatkozik a gépi kódú program visszaolvasására, amihez BASIC-ben csak a PEEK utasítást használhatjuk.
A körülményes, aprólékos munkát elkerülhetjük, ha felszereljük magunkat egy viszonylag olcsó monitorprogrammal.

6.2 A Commodore 64-es operációs rendszerének fontos tárcímei

Az operációs rendszer ROM rutinjai sok könnyítést jelentenek a programozásban azok számára, akik élni tudnak a beépített lehetőségekkel. Különösen jó szolgálatot tehetnek a külső egységek kezelését szervező rendszerrutinok.
A ROM végén található egy táblázat, amely az összes rendszerrutin ugrási címét tartalmazza. Az új típusú Commodore gépek tervezői ezt a táblázatot minden esetben csak bővítik és sohasem változtatják meg. Ez a magyarázata annak, hogy minden olyan gépi kódú rutin, amely valamelyik CBM gépen készült és elsősorban a KERNAL rutinokból építkezik, kisebb változtatásokkal futtatható a Commodore 64-esen is.

A Commodore 64-es a VC 20-as ugrási táblázathoz képest mindössze három címmel bővült, így a VC 20-ason készült programokat pillanatok alatt át lehet írni a C 64-esre.
Célszerű megismerni a külső egységekkel kapcsolatot teremtő rendszerrutinokat:

Cím	*Feladat*
$FF90	beállítja a rendszerüzenetek kiírását vezérlő kapcsolót
$FF93	másodlagos címet küld az IEC buszra a LISTEN utasítás után
$FF96	másodlagos címet küld az IEC buszra a TALK utasítás után
$FF99	ha az átviteli (carry) kapcsoló 1, a legmagasabb RAM címet betölti az X és az Y regiszterekbe; ha a kapcsoló 0, visszaolvassa a címet a regiszterekből
$FF9C	ugyanaz mint fent, de a RAM kezdőcímére vonatkoztatva
$FF9F	lekérdezi a billentyűzetet
$FFA2	beállítja az IEC buszra vonatkozó time-out kapcsolót
$FFA5	betölt egy byte-ot az IEC buszról az akkuba
$FFA8	az akkumulátor tartalmát továbbítja az IEC buszhoz.
$FFAB	UNTALK utasítást küld az IEC buszra
$FFAE	UNLISTEN utasítást küld az IEC buszra
$FFB1	LISTEN utasítást küld az IEC buszra
$FFB4	TALK utasítást küld az IEC buszra
$FFB7	betölti a státuszt az akkumulátorba.
$FFBA	beállítja a file-paramétereket. Hívása előtt a logikai file-számot az akkuba, az egységszámot az X, a másodlagos címet pedig az Y regiszterbe kell beírni.
$FFBD	beállítja a file-név paramétereit. Hívása előtt a név hosszát az akkuba, a címét pedig az X és az Y regiszterekbe kell beírni.
$FFC0	OPEN utasítás, megnyitja a logikai file-t
$FFC3	CLOSE utasítás, lezárja a logikai file-t. Hívása előtt a logikai file-számot az akkuba be kell írni.
$FFC6	CHKIN; kijelöli inputként az X regiszterben megadott logikai file-t. A file-t előzőleg meg kell nyitni az OPEN rutinnal.
$FFC9	CKOUT; kijelöli outputként az X regiszterben megadott logikai file-t. A file-t előzőleg meg kell nyitni az OPEN rutinnal.
$FFCC	CLRHC; visszaállítja az alaphelyzetet: elsődleges input egység a billentyűzet, output egység a képernyő.
$FFCF	BASIN; egy karaktert betölt az akkumulátorba.
$FFD2	BSOUT; egy karaktert kiír az akkumulátorból
$FFD5	LOAD; betölti a programot a tárba.
$FFD8	SAVE; tárolja a programot
$FFDB	újraállítja az időt
$FFDE	beolvassa az időt
$FFE1	lekérdezi a STOP billentyűt
$FFE4	GET; egy karaktert betölt az akkumulátorba
$FFE7	CLALL; visszaállítja az összes input/output csatornát, de a file-okat nem zárja le
$FFEA	hatvanad másodperccel növeli az időt
$FFED	SCREEN; beolvassa a képernyő sorainak és oszlopainak számát
$FFF0	törölt átvitel esetén beállítja a kurzort az X/Y pozícióra, beállított átvitelnél pedig beolvassa a kurzorpozíciót
$FFF3	beolvassa az I/O modul kezdőcímét

A gyakorlott programozók tisztában vannak azzal, hogy gyakran kevesebb fáradtsággal jár egy-egy program lényegi részének megírása, mint az adatforgalomhoz (input/output) szükséges esztétikus képernyő maszkjának programozása.
A képernyőszerkesztés hasznos segédeszközei a következő rutinok:

Cím	*Feladat*
$E518	alapállapotba hozza a képernyőt és a billentyűzetet (RESET)
$E544	CLR; törli a képernyőt
$E566	HOME; a kurzort a képernyő bal felső sarkába mozgatja
$E56C	meghatározza az aktuális kurzorpozíciót
$E5A0	alapállapotba hozza a videovezérlőt
$E5CA	bevitelre vár a billentyűzetről
SE5B4	beolvas egy karaktert a billentyűzet-pufferből
$E8EA	a képernyőt egy sorral felfelé tolja (görgetés)
$E9FF	töröl egy képernyősort
$EA1C	megjelenít a képernyőn egy színes karaktert (a képernyőkód az akkumulátorban, a szín az X regiszterben).

6.3 Az adatforgalom gepi kódú programokkal

Az adatok beolvasásához és kiíratásához számtalan gépi kódú programot állíthatunk össze a ROM rendszerrutinjaiból.
Az alábbi példákban felhasznált rendszerrutinok jobb megértéséhez az Olvasó rendelkezésére áll a részletesen dokumentált ROM lista.

6.3.1 EGY BYTE KIÍRÁSA ÉS BEOLVASÁSA

Az elemi rendszerrutinok:

BSOUT	$FFD2	—	egy byte kiírása
BASIN	$FFCF	—	egy byte beolvasása

Mindkét művelet alapregisztere az akkumulátor. A kiírandó byte mindig az akkumulátorban van, és a beolvasott byte is az akkumulátorba kerül.
A rendszerrutint beépítettük egy olyan programrészbe, amely egy szöveget kiír a képernyőre:

```
P31.ASS

FFD2              100 BSOUT  = $FFD2
1000              110        ;
C000              120        * = $C000
C000              130        ;
C000 A2 00        140 IRAS   LDX #0
C002 BD 0E C0     150 L1     LDA TEXT,X    ; SZOVEG OLVASASA BYTE-KENT
C005 20 D2 FF     160        JSR BSOUT       ; ES KIADAS
C008 E8           170        INX
C009 E0 0C        180        CPX #12         ; NINCS TOBB KARAKTER "?"
C00B D0 F5        190        BNE L1
C00D 60           200        RTS
C00E 4D 49 4E     210 TEXT   .TEXT"MINTASZOVEG"
C019 00           220        .BYTE $00
C01A            65535        .END

BSOUT =FFD2   IRAS  =C000  L1    =C002   TEXT  =C00E
```

A beolvasást a kiíráshoz hasonlóan programozhatjuk.

A szöveget a kurzor megjelenése után a klaviatúráról karakterenként beolvassuk mindaddig, míg a gépkezelő le nem üti a RETURN billentyűt.

```
P32.ASS

FFCF            100 BASIN  = $FFCF
1000            110        ;
C100            120        * = $C100
C100            130        ;
C100 A2 00      140 OLVAS  LDX #0
C102 20 CF FF   150 L1     JSR BASIN   ; EGY KARAKTER OLVASASA
C105 9D 0E C1   160        STA TEXT,X  ; ES TAROLASA
C108 E8         170        INX
C109 C9 0D      180        CMP #13     ; RETURN "?"
C10B D0 F5      190        BNE L1      ; NEM, TOVABBI KARAKTERT BEOLVASNI
C10D 60         200        RTS
C10E 20 20 20   210 TEXT   .TEXT"
SZOVEG RESZERE
C138          65535        .END

BASIN =FFCF  L1    =C102  OLVAS =C100   TEXT  =C10E
```

A legbonyolultabb input/output művelet is lebontható arra a két elemi lépésre, amit ezek a rutinok elvégeznek:
az egyik beolvas egy karaktert a billentyűzetről, a másik kiír egy karaktert a képernyőre.
Az persze egyáltalán nem mindegy, hogy a kiírás során a kurzor a képernyő melyik pozícióján van, és arról is gondoskodnunk kell, hogy pl. egy új programrész üres képernyővel induljon. A képernyőt két utasítással törölhetjük:

```
P33.ASS

FFD2            100 BSOUT  = $FFD2
1000            110        ;
C200            120        * = $C200
C200            130        ;
C200 A9 93      140        LDA #147    ; A KEPERNYO TORLESHEZ SZUKSEGES KOD
C202 20 D2 FF   150        JSR BSOUT   ; KIIRASA
C205 60         160        RTS
C206          65535        .END

BSOUT =FFD2
```

Ennél egyszerűbb megoldás egy közvetlen ugrás a képernyőtörlő rendszerrutinra:

```
JSR CLRSCR
```

A kurzort alaphelyzetbe hozhatjuk a

```
JSR HOME
```

ugróutasítással.
A következő rutin a kurzort a képernyő adott pozíciójára mozgatja:

```
P34.ASS

000A            100 ZEILE  = 10
0005            110 SPALTE = 5
FFF0            120 CURSOR = $FFF0
FFD2            130 BSOUT  = $FFD2
1000            140        ;
C300            150        * = $C300
```



```
C300            160        ;
C300 A6 0A      170        LDX ZEILE   ; A KURZOR SORA  0-24
C302 A4 05      180        LDY SPALTE  ; OSZLOPA        0-39
C304 18         190        CLC         ; CARRY BIT TORLESE
C305 20 F0 FF   200        JSR CURSOR  ; KURZOR BEALLITASA
C308 A9 41      210        LDA #"A"    ; A KIIRANDO KARAKTER
C30A 20 D2 FF   220        JSR BSOUT   ; KIIRAS A KEPERNYORE
C30D 60         230        RTS
C30E          65535        .END

ZEILEN:15   SYMBOLE:4   FEHLER:0

BSOUT =FFD2  CURSOR=FFF0  SPALTE=0005   ZEILE =000A
```

A CURSOR rendszerrutin kettős feladatot lát el. Ha hívásakor az átviteli kapcsoló értéke 0, akkor a kurzort az X és az Y regiszterekben meghatározott sorba, illetve oszlopba mozgatja. Ellenkező esetben (C = 1) az aktuális kurzorpozíció sor- és oszlopkoordinátáját az X, illetve Y regiszterbe tölti.
A rutinok kezdőcímei:

CLRSCR	$E544
HOME	$E566
CURSOR	$FFF0

6.3.2 ADATFORGALOM A KÜLSŐ EGYSÉGEK ÉS A GÉP KÖZÖTT

A külső egységek és a gép közötti adatforgalmat lebonyolító rendszerrutinokat csak akkor tudjuk hatékonyan használni, ha egy kicsit megismerkedünk a folyamat részleteivel.
Minden külső egységhez hozzárendeltek egy 0 és 15 közé eső számot, az egységszámot, amellyel az operációs rendszer az egyes egységeket azonosítja.

Egységszám	*Egység*
0	billentyűzet
1	kazettás egység
2	RS232
3	képernyő
4–15	a soros buszra csatlakoztatott egységek (lemezegység, nyomtató)

Az egységszámhoz vagy elsődleges címhez tartozik egy file-név, és nem kötelezően egy másodlagos cím, amely meghatározza a külső egység üzemmódját.
Amikor először fordulunk a külső egységhez, meg kell adnunk egy logikai file-számot, ezzel küszöböljük ki, hogy minden műveletnél külön-külön meg kelljen adni a fenti paramétereket.
Az OPEN utasításban a logikai file-számhoz hozzárendeljük a file nevét, egy elsődleges és egy másodlagos címet.
A továbbiakban elegendő, ha minden hivatkozást a logikai file-számra vonatkoztatunk. Az OPEN utasítást a BASIC nyelvből már jól ismerjük.
Mielőtt bármilyen input/output műveletet végeznénk, meg kell nyitnunk a file-t OPEN utasítással. Gépi kódban a file paramétereit OPEN előtt közölnünk kell az operációs rendszerrel.
A rendszer a logikai file-számot mindig a $B8 (184), az egységszámot a $BA(186) a másodlagos címet a $B9 (185), a file hosszát a $B7 (183), a file-név címeit pedig a $BB/$BC (187/188) tárcímeken keresi. Ha nem adunk meg file-nevet, a $B7 címre nullát kell beírni. A paraméterek tárolása után meghívhatjuk az OPEN rendszerrutint, amelynek kezdőcíme: $FFC0.
Az operációs rendszer minden input/output műveletet az elsődleges egységekre, a billentyűzetre/képernyőre vonatkoztat mindaddig, amíg ezt az alaphelyzetet nem módosítjuk. A

következő két utasítással megszüntethetjük az alapállapotot azáltal, hogy kijelöljük elsődleges output egységként az LF logikai file-számmal ellátott file-t:

```
LDX LF      ; logikai file-szám
JSR CKOUT ; $FFC9, kiviteli egység a file
```

Az utasítások végrehajtása után minden kiíró műveletet arra a külső egységre vonatkoztatunk, amelyhez a file-számot hozzárendeltük.
Ha LF a nyomtató logikai file-száma, akkor a KIIRÁS rutin a szöveget most a nyomtatóra írná.
A kiíró műveletek mindaddig az LF logikai számhoz rendelt egységre vonatkoznak, amíg az alapállapotot a következő utasítással vissza nem állítjuk:

JSR CLRCH; $FFCC, kiviteli egység a képernyő

Az input műveletek vezérlése ugyanígy programozható:

```
LDX LF      ; a beviteli egység logikai száma
JSR CHKIN ; $FFC6
```

A BASIN utasítás a továbbiakban a megnyitott file-ból olvas mindaddig, amíg CLRCH utasítással vissza nem állítjuk az alaphelyzetet.
A CLRCH utasítással az elsődleges beviteli egységnek a billentyűzetet jelöljük meg. Végrehajtása után a megnyitott file-t CLOSE utasítással le kell zárni.

```
LDA LF      ; logikai file-szám
JSR CLOSE ; $FFC3, a file lezárása.
```

6.3.3 AZ ADATTÁROLÁS TECHNIKÁJA – A LOAD ÉS A SAVE

A Commodore 64-esnél az adatokat és a programokat szalagon vagy lemezen tárolhatjuk. Mivel a kétféle külső egység műszaki felépítése jelentős mértékben különbözik, külön kell foglalkoznunk mindkét tárolási mód programozástechnikai megoldásával.

Adattárolás szalagon

A kazettás egység technikája elvileg csak a soros adatrögzítést és az adatok ugyanilyen sorrendben történő visszaolvasását teszi lehetővé. Meghatározott adatok közvetlen elérése nem lehetséges. Egészen addig folyamatosan olvasnunk kell, amíg el nem érjük a kívánt adatokat. Az adatok módosítására sincs lehetőség. A módosításhoz a teljes file-t be kell olvasni a tárba, majd a módosítás után visszaírni a szalagra.
A programtároláshoz elegendő a soros írás és olvasás, érdemes erre a célra ezt a viszonylag olcsó háttértárolót beszerezni. A kazettás egység egyetlen hátránya a lassúság. A viszonylag lassú olvasás és írás azzal magyarázható, hogy az egység minden adatot kétszer fogad, illetve küld. Erre műszaki okokból feltétlenül szükség van, így kiküszöbölhetjük ugyanis a sérült szalagrészek okozta tárolási hibákat.
Nézzük meg most közelebbről az adattárolás technikáját.
Írás előtt a gép automatikusan beindítja a kazettás egységet. Az operációs rendszer először felír a szalagra egy szinkronjelet, majd ezt kétszer egymás után az adatok követik. Ahhoz, hogy ezt az időigényes procedurát ne kelljen a kelleténél gyakrabban megismételni, az adatokat a szalagra írás előtt egy puffer gyűjti össze. A kazettapuffer 192 karakter hosszúságú és a Commodore 64-esben a 828-tól az 1019-ig terjedő címeken található ($33C–$3FB). A beolvasás is a pufferon át

zajlik le. Miután a szalagon különböző adatokat kell tárolni, nagyon fontos a megkülönböztetési lehetőség. Ennek érdekében a rendszer minden adat elé egy fejlécet *(header)* ír, amely az adat jellegét leíró információkat tartalmazza. A fejléc is először a pufferba kerül, majd innen a szalagra.
A fejléc felépítése a következő:

```
1.      byte – a fejtípus jelölése
2.      byte – kezdőcím, alsó byte
3.      byte – kezdőcím, felső byte
4.      byte – végcím, alsó byte
5.      byte – végcím, felső byte
6–21.   byte – file-név
```

A fejtípus (első byte) értelmezése a következő:

```
1 –  a BASIC programot a BASIC starttól kezdve kell betölteni.
2 –  adatblokk, a 2–192. byte-ok tartalmazzák az adatokat
3 –  a gépi kódú programot a tár rögzített kezdőcímétől kell betölteni
4 –  egy adatállomány adatfejléce
5 –  a szalag végét jelölő „end of tape" blokk.
```

Ha a fejléc 1-es vagy 3-as típusú, a következő 4 byte a program kezdő- és végcímét tartalmazza, majd ezeket követi a program neve. Ezután a szalagon egy blokkban található a program (kétszer egymás után).
A 3-as fejtípus esetén a program attól a címtől kezdődően töltődik be a tárba, amely a fejlécben található. Az 1-es típusnál a rendszer az olvasott kezdőcímet figyelmen kívül hagyja és a programot a BASIC starttól kezdődően tölti be (alaphelyzetben a startcím $800). Ha betöltésnél az 1-es másodlagos címet adjuk meg, a rendszer a programot a megadott címre tölti.
A gépi programoknál erre feltétlenül szükség van, mert ezek a programok csak azon a tárterületen futtathatók, amelyre írtuk őket.
A 2-es fejtípus a PRINT# utasítással szalagra írt adatokra utal.
Az INPUT# illetve a GET# utasítások mindaddig a pufferból olvasnak, míg a puffer ki nem ürül. Ha a puffer kiürült, a programfutás megszakad és a rendszer beolvassa a következő adatblokkot a szalagról a pufferba.
A program tárolásakor a másodlagos címmel határozzuk meg, hogy az adott programot BASIC programként vagy gépi kódú programként kell tárolni. Ha nem adunk meg másodlagos címet, a 0 BASIC programra (1-es fejtípus), az 1-es másodlagos cím (páratlan szám) pedig gépi kódú programra utal. A 2-es vagy 3-as másodlagos címek hatására a rendszer a program után még egy szalagvég jelet (end of tape) tartalmazó blokkot is felír a szalagra.
Nézzük meg az összefüggéseket táblázatos formában:

Betöltés

```
LOAD "NÉV",1     – BASIC program relatív betöltése, ill. a 3-as típusú gépi kódú program
                   abszolút betöltése
LOAD "NÉV",1,1   – minden program abszolút betöltése
```

Tárolás

```
SAVE "NÉV",1     – BASIC programként
SAVE "NÉV",1,1   – tárolás gépi kódú programként
SAVE "NÉV",1,2   – tárolás BASIC programként, kiegészítő EOT blokkal együtt
SAVE "NÉV",1,3   – tárolás gépi kódú programként, kiegészítő EOT blokkal együtt.
```

A szalagfile megnyitásakor a másodlagos címek jelentése a következő:

OPEN 1,1,0 "NÉV" — egy file megnyitása olvasásra
OPEN 1,1,1 "NÉV" — egy file megnyitása írásra
OPEN 1,1,2 "NÉV" — egy file megnyitása írásra, kiegészítő EOT blokkal.

Ha a szalagról olvasás, vagy egy file megnyitása közben a rendszer egy EOT blokkot talál, kiírja a „file not found error" hibaüzenetet.
A CLOSE utasítás lezárja a file-t. Ha a file írásra volt megnyitva, a rendszer a CLOSE hatására egy file-vég jelet (nulla byte-ot) ír a szalagpufferba, majd innen a szalagra. Az elmondottak alapján világos, hogy ha egy szalagfile-t megnyitottunk írásra, azt mindig le is kell zárni, ellenkező esetben ugyanis a rendszer az utolsó adatokat már nem írná fel a szalagra, így a puffer utolsó adattartalma elveszne.
Ha az OPEN utasításban a 2-es másodlagos címet adtuk meg, a rendszer egy EOT blokkal zárja az adatokat.
Az adatkiírás módját néhány megkötés korlátozza.
A rendszer az adatokat ASCII formátumban tárolja a szalagon. Ha a kiírást olyan PRINT# utasításokkal végezzük, amelyben CHR$ függvény is szerepel, hibát okozhat a CHR$(0) kiírása, azt ugyanis a rendszer végjelként értelmezi. Felírás előtt ezt a lehetőséget ki kell zárnunk.

Adattárolás lemezen

A fenti megkötések egyike sem vonatkozik a lemezen adatátvitelre. A lemez sávokra és szektorokra van felosztva, amelyek külön-külön elérhetők. Az adatokhoz anélkül hozzáférhetünk, hogy számos más adatot át kellene olvasnunk.
A programok tárolása:
A rendszer először a program címének alsó- és felső byte-ját tárolja, majd magát a programot. Tároláskor nincs különbség a BASIC program és a gépi kódú program között. Eltérés csak a betöltésnél mutatkozik.

LOAD "NÉV",8 — betölt egy BASIC programot a program kezdőcímét figyelmen kívül hagyva (a BASIC starttól)
LOAD "NÉV",8,1 — betölt egy gépi kódú programot, a betöltés helyét a tároit kezdőcím határozza meg.

A program kezdőcímét lemezről a következő programmal olvashatjuk be:

```
P35

10 OPEN 1,8,0,"NAME":REM PROGRAMFILE-T OLVASASRA MEGNYITNI
20 GET#1,A$,B$:REM KEZDOCIMET BEOLVASNI
30 IF A$="" THEN A$=CHR$(0)
40 IF B$="" THEN B$=CHR$(0)
50 PRINT ASC(A$)+256*ASC(B$):REM KEZDCIM
60 CLOSE1
```

A kezdőcímet decimális alakban kapjuk meg. A 30-as és 40-es sor üres füzéreinek lekérdezése elengedhetetlen, mivel a GET utasítás nem fogadja el a nulla byte-ot.
A következő program egy gépi kódú programot tárol a lemezen:

```
P36

10 OPEN 1,8,1,"NAME":REM PROGRAMFILE-T KIIRASRA NYITNI
20 PRINT#1,CHR$(AD-INT(AD/256)*256);:REM CIM ALSO BYTE
30 PRINT#1,CHR$(AD/256);:REM CIM FELSO BYTE
40 FOR I=0 TO N-1:REM N DB BYTE-T MENTENI
50 PRINT#1,CHR$(PEEK(AD+I));:NEXT
60 CLOSE 1:REM CSATORNA ZARASA
```


Az AD változó a kezdőcímet, az N pedig a program hosszát tartalmazza.
Hogyan tárolhatjuk egy tetszőleges tárterület tartalmát a lemezen és hogyan olvashatjuk azt vissza?
Az operációs rendszer alábbi két rutinja végzi el ezt a feladatot:

```
LOAD    $FFD5
SAVE    $FFD8
```

A LOAD rutin használata:
Az akkumulátorba nullát töltünk, ami a LOAD utasításra utal. Ha a LOAD hívása közben az akkumulátor tartalma 1, akkor a rendszer VERIFY utasítást hajt végre. A másodlagos cím dönti el, hogy hová irányul a betöltés. Ha a másodlagos cím nem egyenlő nullával, a betöltés a programban tárolt címtől kezdődik. Ha a másodlagos cím nulla, az X (alsó) és az Y (felső) regiszterben tárolt cím határozza meg a betöltési területet. A rendszernek tárolás előtt ismernie kell az egységszámot és a file nevét, ezeket szintén rendszerrutinokkal adhatjuk át.
Példa: egy gépi kódú program betöltése lemezről az eredeti tárterületre:

```
P37.ASS

FFD5                100 LOAD   = $FFD5
002D                110 ADR    = $2D
1000                120        ;
C400                130        * = $C400
C400                140        ;
C400 A2 08          150        LDX #8      ; FLOPPY EGYSEG SZAMA
C402 A0 01          160        LDY #1      ; MASODLAGOS, CIM
C404 20 BA FF       170        JSR $FFBA   ; FILE-ADATOK TAROLASA
C407 A9 07          180        LDA #7      ; FILE-NEV HOSSZA
C409 A2 1A          190        LDX #NAME<  ; FILE-NEV TARCIM ALSO BYTE
C40B A0 C4          200        LDY #NAME>  ; CIM - FELSO BYTE
C40D 20 BD FF       210        JSR $FFBD   ; FILE-NEV KIJELOLESE
C410 A9 00          220        LDA #0      ; LOAD-FLAG
C412 20 D5 FF       230        JSR LOAD    ; PROGRAM TOLTES
C415 86 2D          240        STX ADR     ; VEGCIM - ALSO BYTE
C417 84 2E          250        STY ADR+1   ; VEGCIM - FELSO BYTE
C419 60             260        RTS
C41A 46 49 4C       270 NAME   .TEXT"FILENEV"
C421              65535        .END

ADR   =002D  LOAD  =FFD5  NAME  =C41A
```

A LOAD rutin az X és az Y regiszterekbe visszaadja a betöltött program végcímét.
A következő program a szalagról betölt egy programot úgy, hogy a tárolt címet figyelmen kívül hagyja.

```
P38.ASS

FFD5                100 LOAD   = $FFD5
002D                110 ADR    = $2D
1000                120        ;
C400                130        * = $C400
C400                140        ;
C400 A2 01          150        LDX #1      ; MAGNETOFON EGYSEGSZAM
C402 A0 00          160        LDY #0      ; MASODLAGOS CIM
C404 20 BA FF       170        JSR $FFBA   ; FILE-ADATOK TAROLASA
C407 A9 05          180        LDA #5      ; FILE-NEV HOSSZA
C409 A2 1E          190        LDX #NAME<  ; FILE-NEV TARCIM ALSO BYTE
C40B A0 C4          200        LDY #NAME>  ; CIM FELSO BYTE
C40D 20 BD FF       210        JSR $FFBD   ; FILE-NEV ADATOK TAROLASA
```

```
C410 A9 00       220         LDA #0      ; LOAD KAPOCSOLO
C412 A2 00       230         LDX #$00    ; LOAD CIM ALSO BYTE
C414 A0 60       240         LDY #$60    ; LOAD CIM FELSO BYTE
C416 20 D5 FF    250         JSR LOAD    ; PROGRAM TOLTES
C419 86 2D       260         STX ADR     ; VEGCIM - ALSO BYTE
C41B 84 2E       270         STY ADR+1   ; VEGCIM - FELSO BYTE
C41D 60          280         RTS
C41E 4D 41 47    290 NAME    .TEXT"MAGNOFILE"
C427            65535        .END

ADR   =002D  LOAD  =FFD5  NAME  =C41E
```

A Commodore 64-eshez kifejlesztett monitorprogram a programokat mindig abszolút címekkel tárolja és tölti vissza.
BASIC programból közvetlenül LOAD utasítással nem lehet gépi kódú programot betölteni. A LOAD utasításban ugyan megadhatunk 1-es másodlagos címet, és ezzel a gépi kódú program az eredeti tárterületre kerül.
A programból végrehajtott LOAD utasítás betöltés után végcímként a BASIC program végcímét adja vissza, így a vezérlés ismét a BASIC program elejére kerül, ami azt jelenti, hogy a betöltés megismétlődik. Bár ez a BASIC programok utántöltésénél (overlay) célszerűnek látszik, a gépi kódú programok vagy más tárterületek betöltésénél problémát okozhat. Ilyen esetekben ajánlatos egy kis gépi kódú rutint, pl. az előzők valamelyikét használni, amelyet a BASIC programból SYS utasítással hívhatunk meg. Lehetőség van a paraméterek — például a file-név és az egységszám — átadására is. A megfelelő rutinok behívásával és alkalmazásával kapcsolatos tudnivalókat a 6. fejezet tartalmazza.
A programokat vagy tetszőleges tárterületek tartalmát a SAVE rutinnal tárolhatjuk. Előzetesen meg kell adnunk a betöltés paramétereit, a kezdő és a végcímet, az egységszámot és a file nevét. (Az utóbbi lemezes tárolás esetén nem maradhat el.)
A kezdőcím a nulláslap két egymást követő címe kell, hogy legyen (alsó és felső byte) s az akkumulátorba kell betölteni a cím mutatóját. A végcímet az X (alsó byte) és az Y (felső byte) regiszterek tartalmazzák. Az egységszámot és a file nevét ugyanúgy kell megadni, mint a LOAD rutinnál. Tegyük fel, hogy a $C000-tól terjedő tármezőt PROGRAMM néven lemezen akarjuk tárolni és a program kezdőcímét a $FB/$FC tárcímekre írtuk be.

P39.ASS

```
FFD8             100 SAVE    = $FFD8 ; KEZDOCIMET A $FB, $FC TARHELYRE (ALSO-FELS
O BYTE),
1000             110         ;              ; VEGCIMET 220,230 SORBAN BEIRNI
1000             120         ;
C500             130         * = $C500
C500             140         ;
C500 A2 08       150         LDX #8      ; FLOPPY-EGYSEG SZAMA
C502 20 BA FF    160         JSR $FFBA   ; FILE-ADATOK TAROLASA
C505 A9 08       170         LDA #8      ; FILE-NEV HOSSZA
C507 A2 18       180         LDX #NAME<  ; FILE-NEV TARCIM ALSO BYTE
C509 A0 C5       190         LDY #NAME>  ; CIM FELSO BYTE
C50B 20 BD FF    200         JSR $FFBD   ; FILE-NEV ADATOK TAROLASA
C50E A9 FB       210         LDA #$FB    ; MUTATO A KEZDOCIMRE
C510 A2 01       220         LDX #$01    ; VEG CIM +1  ALSO BYTE
C512 A0 09       230         LDY #$09    ; VEG CIM +1  FELSO BYTE
C514 20 D8 FF    240         JSR SAVE    ; PROGRAM TAROLASA
C517 60          250         RTS
C518 50 52 4F    260 NAME    .TEXT"PROGRAMM" ; FILE-NEV
C520            65535        .END

NAME  =C518  SAVE  =FFD8
```


A SAVE rutin a végcím tartalmát már nem tárolja, tehát a program által elfoglalt terület végcíménél mindig 1-gyel nagyobb értéket kell végcímként megjelölni, ha azt akarjuk, hogy a tárolásból a program utolsó byte-ja ne maradjon ki.
A következő példában egy programot név nélkül, szalagvég jellel lezárva tárolunk a szalagon.

P40.ASS

```
FFD8             100 SAVE    = $FFD8
1000             110         ;
C500             120         * = $C500
C500             130         ;
C500 A2 01       140         LDX #1      ; MAGNETOFON-EGYSEG SZAMA
C502 A0 02       150         LDY #2      ; MASODLAGOS CIM A SZALAGVEGJELHEZ (EOT)
C504 20 BA FF    160         JSR $FFBA   ; FILE-ADATOK TAROLASA
C507 A9 00       170         LDA #0      ; FILE-NEV NINCS
C509 20 BD FF    180         JSR $FFBD   ; FILE-NEV ADATOK TAROLASA
C50C A9 2B       190         LDA #$2B    ; MUTATO A BASIC PROGRAMKEZDETRE
C50E A6 2D       200         LDX $2D     ; PROGRAM  VEGCIM ALSO BYTE
C510 A4 2E       210         LDY $2E     ;          VEGCIM FELSO BYTE
C512 20 D8 FF    220         JSR SAVE    ; PROGRAM MENTESE
C515 60          230         RTS
C516            65535        .END

SAVE  =FFD8
```

6.3.4 AZ RS232-ES ILLESZTŐ

Az RS232-es egy illesztő egység a soros adatátvitelhez.
Ugyanezt az illesztőt Európában gyakran V 24 megjelöléssel látják el.
Az adatátvitel soros és párhuzamos módja közötti eltérés nagyon egyszerűen annyi, hogy míg az előzőnél a nyolc bit egyetlen vonalon egymás után, addig az utóbbinál nyolc egymástól független vonalon egyszerre továbbítódik. A kétféle megoldás mindegyikének van előnye, hátránya egyaránt.
A párhuzamos átvitel jóval gyorsabb, mint a soros, ugyanakkor a soros átvitelhez elegendő pl. egyetlen telefonvonal, amit nem mondhatunk el a párhuzamosról.
A Commodore 64-es operációs rendszere alapkiépítésben tartalmazza az RS232-es kezeléséhez szükséges szoftvert.
Maga az illesztő egy, a User-portra csatlakoztatható modul, amely +/— 12 V jelszintet képes megkülönböztetni.
A C 64-es operációs rendszerében az RS232-es egységszáma 2. Ha a programozó a 2-es egységhez OPEN utasítással egy logikai file-t rendel, a rendszer az adatátvitelhez két puffert, egy input és egy output puffert jelöl ki, melyek mindegyike 256 byte-os. Alaphelyzetben ezek a BASIC RAM végén találhatók.
Amikor a BASIC programban OPEN utasítással megnyitunk egy, az RS232-eshez rendelt file-t, a változók tartalma törlődik, ezért az OPEN utasítást mindig a program elején kell elhelyezni. A rendszer ilyenkor azt sem vizsgálja meg, hogy van-e még elegendő tárterület. A CLOSE utasítás hatására a pufferterületek a BASIC számára ismét szabaddá válnak, de a rendszer ekkor végrehajt egy CLR utasítást, így a változók tartalma elvész.
Vigyázni kell tehát arra is, hogy a CLOSE utasítást mindig a BASIC program legvégén helyezzük el.
A BASIC programban egyszerre csak egy RS232-es csatorna lehet nyitva.
Az adatcsatorna lezárása megszakítja az adatátvitelt, és visszaállítja a puffermutatót. A

SYS 61604

utasítás mindaddig vár, míg a puffer tartalma átvitelre nem került, ezért ezt az utasítást (gépi kódban JSR $F0A4) minden CLOSE utasítás előtt célszerű kiadni.
Az adatátvitelhez szükséges paramétereket a vezérlőregiszter és az utasításregiszter rögzíti. A két regiszter tartalma a file-név első két karaktereként kerül átvitelre.
A vezérlőregiszter tartalma határozza meg az átvitel sebességét *baud*ban (bit/s) mérve, illetve az adat- és stopbitek számát.
Minden adatszó (5–8 bit) átvitelét egy stopbit átvitele követi.
Az utasításregiszter tartalma meghatározza az átvitel – a handshake – illetve a paritásvizsgálat módját.

A vezérlőregiszter alsó négy bitjének jelentése:

Bitek 3210	decimális	átviteli sebesség (baud)	
0000	0	az átvitel sebességet a programozó a file nevében adja át	
0001	1	50	
0010	2	75	
0011	3	110	
0100	4	134.5	
0101	5	150	
0110	6	300	
0111	7	600	
1000	8	1200	
1001	9	1800	
1010	10	2400	
1011	11	3600	(nem használható)
1100	12	4800	(nem használható)
1101	13	7200	(nem használható)
1110	14	9600	(nem használható)
1111	15	19200	(nem használható)

A táblázat szerint az 50 és a 2400 közötti átviteli sebességek programozására van lehetőségünk.
Az adatbitek száma az 5. és 6. bit tartalma szerint az alábbi:

Bitek 6 5	decimális	adatbitek száma
0 0	0	8 bit
0 1	32	7 bit
1 0	64	6 bit
1 1	96	5 bit

A stopbitek számát a 7. bit tartalma határozza meg:

7. bit	decimális	stopbitek száma
0	0	1. stopbit
1	128	2. stopbit

Az utasításregiszter tartalmának jelentése:

0. bit	decimális	handshake
0	0	3 huzalos
1	1	X huzalos

4. bit	decimális	az átvitel módja
0	0	duplex
1	16	félduplex

Bitek 7 6 5	decimális	a paritás ellenőrzése
X X 0	0	nincs paritásellenőrzés, nincs 8. adatbit
0 0 1	32	páratlan paritás
0 1 1	96	páros paritás
1 0 1	160	nincs paritásellenőrzés, a 8. adatbit mindig 1
1 1 1	224	nincs paritásellenőrzés, a 8. adatbit mindig 0.

Tegyük fel, hogy egy RS232-es adatcsatornát az alábbi paraméterekkel akarunk megnyitni:

átviteli sebesség : 2400 baud
adatbitek száma : 7 ASCII bit
stopbitek száma : 2
paritásellenőrzés : nincs
a 8. bit mindig 0
az átvitel módja : duplex
handshake : 3 huzalos

A paramétereket közvetítő OPEN utasítás:

```
OPEN 1,2,0,CHR$(10 + 0 + 128) + CHR$(0 + 0 + 224)
```

Az átviteli sebesség programozása

A fenti táblázatban megadott baud értékektől eltérhetünk, ha az átviteli sebességet a CIA timeren keresztül programozzuk.
A 2400 baudos sebességet szoftveresen semmiképpen nem adhatjuk meg, ehhez ugyanis az RS232-es illesztő túlságosan lassú.
A timer a baud értéktől függő időközönként kivált egy nem maszkolható megszakítást (NMI), és ezalatt az idő alatt zajlik le az adatok fogadása. Az OPEN utasításban a file-név harmadik és negyedik karaktereként megadhatjuk az idő értékét. Az idő és a baud értéke közötti átszámítás képlete:

```
T = 492662/BAUD – 101
```

A kapott érték alsó byte-ja a file-név harmadik-, felső byte-ja pedig negyedik karaktere.
A vezérlőregiszternek ilyenkor nullát kell tartalmaznia, ez tájékoztatja az operációs rendszert arról, hogy az átviteli sebességet a file nevében kell keresnie.
Az alábbi példában az előzővel azonos paraméterek mellett az átviteli sebesség értékét 1000 baudra programoztuk:

```
100 BAUD = 1000
110 T = 492622/BAUD – 101
120 TH = INT(T/256):TL = T–TH * 256
130 OPEN 1,2,0,CHR$(128) + CHR$(224) + CHR$(TL) + CHR$(TH)
```

A programozott átviteli sebesség értéke 8 baudtól 2400 baudig terjedhet.
Az állapotregiszter jelentése az RS232-es esetében eltér a megszokottól.
BASIC programból most is az ST változót kell lekérdezni, ennek értéke azonban minden leolvasás után törlődik.
A rendszerváltozó értékét célszerű egy munkaváltozóban tárolni, az ST ugyanis csak akkor vonatkozik az RS232-esre, ha az utoljára végzett adatátviteli művelet az illesztőn át zajlott le.
Gépi kódú programmal anélkül is leolvashatjuk a státuszt, hogy annak értéke leolvasás után törlődne.
Az állapotregiszter egyes bitjeinek jelentése:

Bit	Leírás
0	paritáshiba
1	kerethiba
2	az input puffer megtelt
3	használaton kívül
4	nincs CTS jel (Clear To Send)
5	használaton kívül
6	nincs DSR jel (Data Set Ready)
7	a break jelet az illesztő átvette

A feltétel teljesülése esetén a megfelelő bit értéke 1.
Gépi kódú programmal az RS232-es puffereit (input és output) a tár bármely területén elhelyezhetjük. Az OPEN utasításban beállított puffermutatót később megváltoztathatjuk.
Az input puffer mutatója a $F7/$F8, az output puffer mutatója pedig a $F9/$FA tárcímeken található a nullláslapon.
Az RS232-es gépi kódú programozása pontosan azokat a szabályokat követi, amelyek bármely más külső egység programozására vonatkoznak, azzal az eltéréssel, hogy egységszámként 2-t kell megadni (lásd a 6.3 alfejezetet).

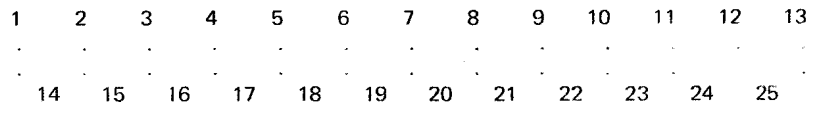
Az RS232-es néhány fontos input/output tárcíme:

$0293	659	vezérlőregiszter
$0294	660	utasításszó
$0295/$0296	661/662	az idő értéke
$0297	663	állapotregiszter
$0298	664	az adatbitek száma (az OPEN utasításban)
$0299/$029A	665/666	az idő értéke adatküldésnél
$029B		az input puffer mutatója írás közben
$029C		az input puffer mutatója olvasás közben
$029D		az output puffer mutatója olvasás közben
$029E		az output puffer mutatója írás közben

A nullláslapon van még néhány, az adatátvitelhez szükséges munkarekesz.

Az RS232-es illesztő csatlakoztatása

Az RS232-es egy Cannon típusú 25 pólusú csatlakozóval rendelkezik, amelynek kiosztása a következő:



Láb	Rövidítés	Irány	Angol megnevezés
1	GND		Protective Ground
2	TXD	OUT	Transmitted Data
3	RXD	IN	Received Data
4	RTS	OUT	Request To Send
5	CTS	IN	Clear To Send
6	DSR	IN	Data Set Ready
7	GND		Signal Ground
8	DCD	IN	Data Carrier Detected
20	DTR	OUT	Data Terminal Ready
22	RI	IN	Ring Indicator

Két számítógép összekapcsolását írja le a következő táblázat:

1-es gép	láb		láb	2-es gép
Transmitted Data	2	----→	3	Received Data
Received Data	3	←----	2	Transmitted Data
Ready To Send	4	----→	5	Clear To Send
Clear To Send	5	←----	4	Ready To Send
Data Set Ready	6	←----	20	Data Terminal Ready
Data Terminal Ready	20	----→	6	Data Set Ready
Ground	7	←---→	7	Ground

A táblázatban a nyilak az információ áramlásának irányát jelzik.
Az ábrázolt csatlakoztatáshoz X-es handshake-et használtunk. 3-as handshake mellett elegendő a 2-es, 3-as és 7-es lábakat csatlakoztatni. Ha egy olyan nyomtatót akarunk illeszteni, amelynek az átviteli paramétereit nem ismerjük, alkalmazzuk a következő trükköt:
Kössük össze a 2-es, 3-as és 7-es lábakat a fentiek szerint, a 4-es, 5-ös és 8-as, illetve a 6-os és 20-as lábakat pedig zárjuk rövidre.
Ezzel a módszerrel a legtöbb egységet „munkára lehet kényszeríteni". Ha átvitel közben az adatok elvesznének, változtassuk az átviteli sebességet, a paritást, esetleg mindkettőt mindaddig, míg elfogadható eredményt nem kapunk.
A vezérlővonalak rövidre zárásával gyakorlatilag 3-huzalos üzemmódban dolgozunk. Ha mégsem sikerül megfelelő sebességet elérni, célszerű BASIC helyett gépi kódban programozni.

6.3.5 A SOROS BUSZ

A soros buszon keresztül a C 64-eshez több, egyszerre üzemeltethető külső egységet csatlakoztathatunk. A soros busz a nagy CBM gépeken ismert IEC buszhoz (IEEE 488) hasonló elven működik, de az adatokat nem párhuzamosan (byte-onként), hanem sorosan (bitenként) továbbítja.

Az IEC busz működési elve

Az IEC buszra csatlakoztatott egységeket meg kell különböztetni egymástól. Erre szolgál az elsődleges cím, vagy egységszám.
A Commodore 64-es operációs rendszere az IEC busz egységeihez a 4-től 15-ig terjedő egységszámokat rendeli.
Amikor egy egységet meg kell „szólítani", a buszvezérlő — esetünkben az alapgép — egy figyelmeztető jelzést (attention, rövidítve ATN) ad le a soros buszra, ezt követően elküldi a

ivválasztott egység számát, majd az ATN jelet visszavonja. Következő lépésként tudatni kell a
ivválasztott egységgel, hogy az adatok küldésére vagy fogadására készüljön fel.
z angol „talk" (beszélni) és „listen" (hallgatni) szavaknak megfelelően az egység a rendszertől
ALK vagy LISTEN utasítást kap aszerint, hogy küldenie vagy fogadnia kell az adatokat.
z utasítást — nem kötelező jelleggel — követi egy másodlagos cím, amely meghatározza az
datfeldolgozás módját. Most kerülhet csak sor a tényleges adatátvitelre. A soronkövetkező
yte átvitelére csak akkor kerülhet sor, ha az előző átvitel rendben lezajlott. A handshake
onalon keresztül közli az adatot küldő egység, hogy az adatot továbbította a buszra, illetve itt
lzi a fogadó egység, hogy az adatot feldolgozta és készen áll a következő byte fogadására. Az
tvitel sebességét a két egység együttesen szabja meg. Az adatátvitel befejeztével a rendszer törli
buszról az egységszámot oly módon, hogy a fogadó egységnek UNLISTEN, a küldő egységnek
NTALK parancsot ad.
soros busz ismét szabad a következő ATN jel érkezéséig.
logyan írjunk gépi kódú programot az IEC busz kezelésére?
z operációs rendszer megfelelő rutinjai most is rendelkezésünkre állnak (ld. a ROM lista utolsó
ldalait).
éldaként nézzük meg, hogyan lehet a lemezegység hibaüzenetét gépi kódú programmal
eolvasni.
megértést biztosan elősegíti, ha először BASIC nyelven oldjuk meg a feladatot:

P41

```
10 OPEN 15,8,15 :REM PARANCS-CSATORNA NYITASA
20 INPUT#15,A$,B$,C$,D$: REM HIBAUZENET BEOLVASASA
30 PRINT A$","B$","C$","D$:REM ES KIIRASA
40 CLOSE 15 : REM PARANCS-CSATORNA LEZARASA
```

Az INPUT utasítást parancsként nem adhatjuk ki, a hibacsatornát tehát csak programból tudjuk
eolvasni.
feladat megoldása gépi kódban:

P42.ASS

```
00BA              100 FA      = $BA
FFB4              110 TALK    = $FFB4
00B9              120 SA      = $B9
FF96              130 SATALK  = $FF96
FFA5              140 IECIN   = $FFA5
FFD2              150 PRINT   = $FFD2
FFAB              160 UNTALK  = $FFAB
1000              170         ;
C000              180         * = $C000
C000              190         ;
C000 A9 08        200         LDA #8      ; FLOPPY-EGYSEG SZAMA
C002 85 BA        210         STA FA
C004 20 B4 FF     220         JSR TALK    ; TALK (ADASRA FELSZOLLITAS) PARANCS AZ
EGYSEGNEK
C007 A9 6F        230         LDA #15+$60 ; MASODLAGOS CIM 15+$60
C009 85 B9        240         STA SA
C00B 20 96 FF     250         JSR SATALK  ; TALK PRANCS A MASODLAGOS CIMNEK
C00E 20 A5 FF     260 L       JSR IECIN ; EGY BYTE BEOLVASASA
C011 20 D2 FF     270         JSR PRINT   ; KIIRAS A KEPERNYORE
C014 C9 0D        280         CMP #13     ; EZ CARRIGE RETURN "?"
C016 D0 F6        290         BNE L       ; NEM, TOVABBI JELEK OLVASASARA
C018 20 AB FF     300         JSR UNTALK ; UNTALK (TALK PARANCS TORLVE) PARANCS A
Z EGYSEGNEK
C01B 60           310         RTS         ; KESZ
C01C             65535        .END

FA    =00BA  IECIN =FFA5  L     =C00E   PRINT =FFD2  SA    =00B9  SATALK=FF96
TALK  =FFB4  UNTALK=FFAB
```


A BASIC betöltő:

P43

```
100 FOR I=49152 TO 49179
110 READ X:POKE I,X :S=S+X: NEXT
120 DATA 169,  8,133,186, 32,180,255,169,111,133,185, 32
130 DATA 150,255, 32,165,255, 32,210,255,201, 13,208,246
140 DATA  32,171,255, 96
150 IF S<>4169 THEN PRINT"HIBA AZ ADATOKBAN!":END
160 PRINT"RENDBEN!"
```

A gépi kódú programot a SYS 12 * 4096 utasítással kell indítani.
Az alábbi, valamivel hosszabb gépi kódú programmal beolvashatjuk a lemez tartalomjegyzékét
anélkül, hogy a tárbeli BASIC program elveszne.

P44.ASS

```
00BB              100 FNADR   = $BB
00B7              110 FNLEN   = $B7
00BA              120 FA      = $BA
00B9              130 SA      = $B9
F3D5              140 SNDNAM  = $F3D5
FFB4              150 TALK    = $FFB4
FF96              160 SATALK  = $FF96
0090              170 STATUS  = $90
FFA5              180 IECIN   = $FFA5
BDCD              190 LNPRT   = $BDCD
FFD2              200 PRINT   = $FFD2
F642              210 CLSFIL  = $F642
1000              220         ;
C000              230         *=$C000
C000              240         ;
C000 A9 24        250         LDA #"$"      ; FILENEV (DOLLARJEL)
C002 85 FB        260         STA $FB       ; TAROLASA
C004 A9 FB        270         LDA #$FB      ; FILE-NEV CIME
C006 85 BB        280         STA FNADR
C008 A9 00        290         LDA #0        ; FELSO BYTE
C00A 85 BC        300         STA FNADR+1
C00C A9 01        310         LDA #1        ; FILE-NEV HOSSZA
C00E 85 B7        320         STA FNLEN
C010 A9 08        330         LDA #8        ; LEMEZEGYSEG SZAMA
C012 85 BA        340         STA FA
C014 A9 60        350         LDA #$60      ; LOAD MASODLAGOS CIME
C016 85 B9        360         STA SA
C018 20 D5 F3     370         JSR SNDNAM    ; FILE NYITAS - NEVVEL
C01B A5 BA        380         LDA FA
C01D 20 B4 FF     390         JSR TALK      ; TALK (ADAS) PARANCS AZ EGYSEGNEK
C020 A5 B9        400         LDA SA
C022 20 96 FF     410         JSR SATALK    ; TALK PARANCS A MASODLAGOS CIMRE
C025 A9 00        420         LDA #0
C027 85 90        430         STA STATUS    ; STATUS TORLESE
C029 A0 03        440         LDY #3        ; ELSO 3 BYTE ATOLVASASA
C02B 84 FB        450 L1      STY $FB   ; TAROLASA SZAMLALOKENT
C02D 20 A5 FF     460         JSR IECIN     ; BYTE OLVASASA
C030 85 FC        470         STA $FC       ; ES TAROLASA
C032 A4 90        480         LDY STATUS    ; STATUS VIZSGALAT
C034 D0 2F        490         BNE L4
C036 20 A5 FF     500         JSR IECIN     ; EGY BYTE OLVASASA
C039 A4 90        510         LDY STATUS    ; STATUS VIZSGALAT
C03B D0 28        520         BNE L4
C03D A4 FB        530         LDY $FB       ; SZAMLALO OLVASASA
C03F 88           540         DEY           ; ES CSOKKENTESE
C040 D0 E9        550         BNE L1
```

```
C042 A6 FC       560        LDX $FC     ; BYTE VISSZATOLTESE
C044 20 CD BD    570        JSR LNPRT   ; 16-BITES SZAM KIADASA
C047 A9 20       580        LDA #" "    ; FOGLALT BLOKKOK SZAMA
C049 20 D2 FF    590        JSR PRINT   ; SZOKOZ KIIRAS
C04C 20 A5 FF    600 L3     JSR IECIN; KOVETKEZO BYTE OLVASASA
C04F A6 90       610        LDX STATUS  ; STATUS VIZSGALAT
C051 D0 12       620        BNE L4
C053 AA          630        TAX         ; BYTE VIZSGALAT
C054 F0 06       640        BEQ L2      ; NULL "?", HA IGEN, SORVEG
C056 20 D2 FF    650        JSR PRINT   ; EGYEBKENT A BYTE KIIRASA
C059 4C 4C C0    660        JMP L3      ; ES A KOVETKEZO KARAKTER OLVASASA
C05C A9 0D       670 L2     LDA #13  ; CARRIAGE RETURN
C05E 20 D2 FF    680        JSR PRINT   ; KIIRASA
C061 A0 02       690        LDY #2      ; KETTO BYTE A SZEKTOR LANCOLASHOZ
C063 D0 C6       700        BNE L1      ; FOLYTATAS
C065 20 42 F6    710 L4     JSR CLSFIL ; LOGIKAI FILE ZARASA
C068 60          720        RTS
C069          65535         .END

CLSFIL=F642  FA     =00BA  FNADR =00BB   FNLEN =00B7  IECIN =FFA5  L1     =C02E
L2     =C05C  L3     =C04C  L4     =C065   LNPRT =BDCD  PRINT =FFD2  SA     =00B9
SATALK=FF96  SNDNAM=F3D5  STATUS=0090   TALK  =FFB4
```

A BASIC betöltő:

```
P45

100 FOR I=49152 TO49256
110 READ X:POKEI,X:S=S+X:NEXT
120 DATA 169, 36,133,251,169,251,133,187,169,  0,133,188
130 DATA 169,  1,133,183,169,  8,133,186,169, 96,133,185
140 DATA  32,213,243,165,186, 32,180,255,165,185, 32,150
150 DATA 255,169,  0,133,144,160,  3,132,251, 32,165,255
160 DATA 133,252,164,144,208, 47, 32,165,255,164,144,208
170 DATA  40,164,251,136,208,233,166,252, 32,205,189,169
180 DATA  32, 32,210,255, 32,165,255,166,144,208, 18,170
190 DATA 240,  6, 32,210,255, 76, 76,192,169, 13, 32,21'
200 DATA 255,160,  2,208,198, 32, 66,246,96
210 IF S<>15343 THEN PRINT"HIBA AZ ADATOKBAN!":END
220 PRINT"RENDBEN!"
```

A gépi kódú programot ismét a SYS 12 * 4096 utasítással indíthatjuk el. A képernyőn megjelenik a lemez tartalomjegyzéke, és közben a tárbeli BASIC program sértetlen marad.
Az átviteli sebességet nagymértékben növelhetjük, ha a külső egységek illesztésére soros IEC busz helyett egy a tárbővítő portra csatlakoztatható IEC buszt használunk.
Ezzel a nagy Commodore gépek külső egységeinek — mint pl. a kettős lemezmeghajtó — csatlakoztatását is megoldhatjuk.
A párhuzamos IEC busz programozása csak annyiban tér el a soros busz programozásától, hogy meg kell változtatnunk az olyan alaprutinok kezdőcímeit, mint pl. egy byte beolvasása és kiírása, a TALK, illetve a LISTEN parancs továbbítása stb. Ha viszont a gépi kódú program logikai file-on keresztül, a BASIN és a BASOUT utasításokkal dolgozik (ld. 6.3 alfejezet), minden változtatás nélkül használható a párhuzamos IEC buszra is.

7. FEJEZET
A COMMODORE 64-ES, A VC 20-AS ÉS A CBM GÉPEK TÁRKIOSZTÁSÁNAK ÖSSZEHASONLÍTÁSA

7.1 A nulláslap és egyéb fontos tárterületek kiosztása

Cím		Foglaltság
hexadecimális	decimális	
00	0	a processzorport adatirány-regisztere
01	1	a processzorport
02	2	használaton kívül
03–04	3–4	a lebegőpontos/fixpontos átalakítás vektora
05–06	5–6	a fixpontos/lebegőpontos átalakítás vektora
07	7	a kereső karakter
08	8	az idézőjel kapcsoló
09	9	az oszlop, a TAB utasításnál
0A	10	LOAD = 0, VERIFY = 1, interpreterkapcsoló
0B	11	mutató az input pufferben, a dimenziók száma
0C	12	a DIM utasítás kapcsolója
0D	13	típus, $00 = numerikus, $FF = szöveges
0E	14	kapcsoló, egész= $80, valós = $00
0F	15	idézőjel kapcsoló LIST utasításnál
10	16	FN kapcsoló
11	17	INPUT($00), GET($40), READ($98) kapcsoló
12	18	az ATN előjele
13	19	aktív I/O egység
14–15	20–21	egészértékű cím, például a sorszám
16–	22	mutató a füzérveremben
17–18	23–24	mutató az utolsó füzérre
19–21	25–33	füzérverem
22–25	34–37	mutató különböző célokra
26–2A	38–42	függvénykiértékelő és aritmetikai regiszter
2B–2C	43–44	mutató a BASIC program kezdetére
2D–2E	45–46	mutató a változóterület kezdetére
2F–30	47–48	mutató a tömbkezdetre
31–32	49–50	mutató a tömbök végére
33–34	51–52	mutató a füzérek kezdetére
35–36	53–54	füzérek segédmutatója
37–38	55–56	mutató a BASIC RAM végére
39–3A	57–58	az aktuális BASIC sorszám
3D–3E	61–62	a következő utasítás mutatója CONT-nál
3F–40	63–64	DATA utasítás aktuális sorszáma
41–42	65–66	mutató a következő DATA elemre
43–44	67–68	a bevitel forrása, mutató
45–46	69–70	a változók neve
47–48	71–72	a változók címe
49–4A	73–74	a változók értéke, mutató
4B–4C	75–76	a programmutató átmeneti tára
4D	77	összehasonlító műveletek maszkja
4E–4F	78–79	mutató az FN-re
50–53	80–83	füzér-leíró

Cím		Foglaltság
hexadecimális	decimális	
54	84	konstans $4C, ugrás a függvényekre
55–56	85–86	függvények ugrási vektora
57–5B	87–91	aritmetikai regiszter, akku# 3
5C–60	92–96	aritmetikai regiszter, akku# 4
61–65	97–101	lebegőpontos akku# 1, FAC
66	102	a FAC előjele
67	103	számláló a polinom kiértékeléséhez
68	104	a FAC kerekítő byte-ja
69–6D	105–109	lebegőpontos akku# 2, ARG
6E	110	az ARG előjele
6F	111	a FAC és az ARG előjelének összehasonlító byte-ja
70	112	a FAC kerekítő byte-ja
71–72	113–114	a polinom kiértékelése, mutató
73–8A	115–138	CHRGET rutin, egy karakter beolvasása a BASIC szövegből
7A–7B	122–123	programmutató
8B–8F	139–143	az utolsó RND érték
90	144	az ST státuszszó
91	145	a STOP billentyű kapcsolója
92	146	időállandó a szalaghoz
93	147	LOAD ($00) vagy VERIFY ($01) Kapcsoló
94	148	soros (IEC) kivitel, kapcsoló
95	149	az IEC busz output puffere
96	150	EOT vétele szalagról, kapcsoló
97	151	a regiszterek közbenső tára
98	152	a megnyitott file-ok száma
99	153	az aktív input egység
9A	154	az aktív output egység
9B	155	szalagparitás
9C	156	a „byte vétele megtörtént" kapcsoló
9D	157	közvetlen üzemmód($80), ($00) programmód, kapcsoló
9E	158	szalag 1. menet,ellenőrzőösszeg
9F	159	szalag 2. menet, hibajavítás
A0–A2	160–162	idő
A3	163	a soros olvasás bitszámlálója
A4	164	szalagszámláló
A5	165	számláló szalagra írás közben
A6	166	mutató a szalagpufferban
A7–AB	167–171	munkaterület a szalagműveletekhez
AC–AD	172–173	a szalagpuffer és a görgetés (scroll) mutatója
AE–AF	174–175	a program vége, mutató a LOAD/SAVE utasításnál
B0–B1	176–177	a szalagműveletek állandói
B2–B3	178–179	szalagpuffer-mutató
B4	180	bitszámláló a szalagon
B5	181	az RS232 következő bitje
B6	182	a kihozandó byte puffere
B7	183	a file-név hossza
B8	2	a logikai file-szám
B9	185	a másodlagos cím

Cím		Foglaltság
hexadecimális	decimális	
BA	186	az egységszám
BB–BC	187–188	a file-név, mutató
BD	189	a soros bevitel/kihozatal munkaterülete
BE	190	szalagmenet számláló
BF	191	a soros kihozatal puffere
C0	192	szalaghajtó motor, kapcsoló
C1–C2	193–194	a képernyős input/output kezdőcíme
C3–C4	195–196	a képernyős input/output végcíme
C5	197	a lenyomott billentyű sorszáma, 64 = nincs lenyomva billentyű
C6	198	a lenyomott billentyűk száma
C7	199	RVS mód, kapcsoló
C8	200	a sor vége bevitelkor
C9	201	a kurzor sora bevitelnél
CA	202	a kurzor oszlopa bevitelnél
CB	203	lenyomott billentyű, 64 = nincs lenyomott billentyű
CC	204	kurzor kapcsoló; 0 = kurzor BE, 1 = kurzor KI
CD	205	a kurzorvillogás számlálója
CE	206	a kurzor alatti karakter
CF	207	kurzorkapcsoló; 1 = BE-fázis, 0 = KI-fázis
D0	208	bevitel a billentyűzetről, vagy a képernyőről, kapcsoló
D1–D2	209–210	az aktuális képernyősor kezdete, mutató
D3	211	a kurzor oszlopa
D4	212	idézőjel mód, kapcsoló
D5	213	a képernyősor hossza
D6	214	a kurzor sora
D7	215	különböző célokra
D8	216	a beszúrások száma
D9–F2	218–242	a képernyősor kezdetének MSB-je
F3–F4	243–246	a szín-RAM, mutató
F5–F6	245–246	a billentyűzet dekódolási táblázata, mutató
F7–F8	247–248	az RS232-es input puffer, mutató
F9–FA	249–250	az RS232-es output puffer, mutató
		* * * * *
00FF–010A	255–266	lebegőpontos érték/ASCII átalakítás puffere
0100–013E	256–318	javítás szalagolvasásnál
0100–01FF	256–511	processzor-verem
0200–0258	512–600	BASIC input puffer
0259–0262	601–6100	a logikai file-számok táblázata
0263–026C	611–620	az egységszámok táblázata
026D–0276	621–630	a másodlagos címek táblázata
0277–0280	631–640	a billentyűzet puffer
0281–0282	641–642	a BASIC RAM kezdete
0283–0284	643–644	a BASIC RAM vége
0285	645	a soros IEC busz time out kapcsolója
0286	646	az aktuális szín
0287	647	a kurzor alatti szín
0288	648	a videoram, felső byte
0289	649	a billentyűzet-puffer hossza

Cím		Foglaltság
hexadecimális	decimális	
028A	650	az összes billentyű ismétlési funkciója, kapcsoló
028B	651	az ismétlési sebesség számlálója
028C	652	az ismétlési késleltetés számlálója
028D	653	a SHIFT, a COMMODORE és a CTRL (0, 1, és 2. bit) kapcsolója
028E	654	a SHIFT kapcsolója
028F–0290	655–656	a billentyűzet-dekódolás, mutató
0291	657	a SHIFT/COMMODORE letiltva, kapcsoló
0292	658	képernyőgörgetés, kapcsoló
0293	659	az RS232-es vezérlőregiszter
0294	660	az RS232-es utasításregiszter
0295–0296	661–662	bit-timing
0297	663	az RS232-es státusz
0298	664	az adatbitek száma az RS232-nél
0299–029A	665–666	az RS232-es átviteli sebessége
029B	667	az RS232-n keresztül fogadott byte, mutató
029C	668	RS232-es input, mutató
029D	669	RS232-esen továbbítandó byte, mutató
029E	670	RS232-es output, mutató
029F–02A0	671–672	szalagüzemmód alatti IRQ tárolása
02A1	673	CIA 2 NMI, kapcsoló
02A2	674	CIA 1, A timer
02A3	675	CIA 1 megszakítás, kapcsoló
02A4	676	a CIA 1, A timer, kapcsoló
02A5	677	a képernyősor
02A6	678	PAL- (1) vagy az NTSC-változat (0), kapcsoló
02C0–02FE	704–766	11. sprite
0300–0301	768–769	$E38B-a BASIC melegindítás vektora
0302–0303	770–771	$A483- egy sor bevitelének vektora
0304–0305	772–773	$A57C- átalakítás interpreter kóddá, vektor
0306–0307	774–775	$A71A- átalakítás szöveggé, vektor (LIST)
0308–0309	776–777	$A7E4- a BASIC utasításcím beolvasása, vektor
030A–030B	778–779	$AE87- a kifejezés kiértékelésének vektora
030C	780	a SYS utasítás akkumulátora
030D	781	a SYS utasítás X regisztere
030E	782	a SYS utasítás Y regisztere
030F	783	a SYS utasítás állapotregisztere
0310	784	$4C – JMP a USR függvényre
0311–0312	785–786	$B248 USR vektor
0314–0315	788–789	$EA31 IRQ vektor
0316–0317	790–791	$FE66 BRK vektor
0318–0319	792–793	$FE47 NMI vektor
031A–031B	794–795	$F34A OPEN vektor
031C–031D	796–797	$F291 CLOSE vektor
031E–031F	798–799	$F20E CHKIN vektor
0320–0321	800–801	$F250 CKOUT vektor
0322–0323	802–803	$F333 CLRCH vektor
0324–0325	804–805	$F157 INPUT vektor
0326–0327	806–807	$F1CA OUTPUT vektor
0328–0329	808–809	$F6ED STOP vektor

Cím		Foglaltság
hexadecimális	decimális	
032A–032B	810–811	$F13E GET vektor
032C–032D	812–813	$F32F CLALL vektor
032E–032F	814–85	$FE66 a melegindítás vektora
0330–0331	816–817	$F4A5 LOAD vektor
0332–0333	818–819	$F5ED SAVE vektor
033C–03FB	828–1019	szalagpuffer
0340–037E	832–894	13. sprite
0380–03BE	896–958	14. sprite
03C0–03FE	960–1022	15. sprite

7.2 A BASIC rutinok kezdőcímei

A Commodore 64-es és a VC 20-as gépek BASIC interpreterje teljesen azonos. A C 64-es címeket könnyen átszámíthatjuk a VC 20-as megfelelő címeivé.
A $A000 és $BFFF tárcímek közötti C 64-es címhez $2000-t hozzáadva megkapjuk a tárcím VC 20-as megfelelőjét. Pl. a C 64-es $A860 tárcímnek a VC 20-as $C860 felel meg. A $E00-tól a $E37A-ig terjedő tartományban a C 64-es címekből le kell vonnunk 3-at. Pl. a C 64-es $E30E címéhez ily módon a $E30B VC 20-as tárcím tartozik.

Cím	Leírás
A000	start vektor
A002	NMI vektor
A004	„cbmbasic"
A00C	a BASIC utasítások címe, mínusz 1
A052	a BASIC függvények címei
A080	a BASIC műveletek hierarchia-kódjai és címei
A09E	a BASIC utasításszavak jegyzéke
A19E	a BASIC hibaüzenetek
A328	a hibaüzenetek címei
A364	a BASIC értelmező üzenetei
A38A	a FOR-NEXT és a GOSUB veremkereső rutinja
A3B8	blokk-eltoló rutin
A3FB	szabad hely keresése a veremben
A408	helyfoglalás a tárban
A435	az "out of memory" kiírása
A437	a hibaüzenet kiírása
A469	Break-beugrás
A474	Ready-beugrás
A480	az input várakozó ciklusa
A49C	a programsorok törlése és beszúrása
A533	a BASIC programsorok újraláncolása
A560	egy sor beolvasása az input pufferba
A571	a "string too long" kiírása
A579	egy sor átalakítása interpreter-kódokká
A613	a BASIC sor kezdőcímének keresése
A642	a NEW BASIC utasítás
A65E	a CLR BASIC utasítás
A68E	a programmutató beállítása a BASIC kezdetre
A69C	a LIST BASIC utasítás

Cím	Leírás
A717	az interpreter-kód átalakítása utasításszóvá
A742	a FOR BASIC utasítás
A7AE	az interpreter-ciklus, egy BASIC utasítás végrehajtása
A8ED	egy BASIC utasítás végrehajtása
A81D	a RESTORE BASIC utasítás
A82C	a STOP billentyű lenyomott állapotában megszakítja a programot
A82F	a STOP BASIC utasítás
A831	az END BASIC utasítás
A857	a CONT BASIC utasítás
A871	a RUN BASIC utasítás
A883	a GOSUB BASIC utasítás
A8A0	a GOTO BASIC utasítás
A8D2	a RETURN BASIC utasítás
A8A8	a DATA BASIC utasítás
A906	a következő utasítás keresése
A909	a következő sor keresése
A928	az IF BASIC utasítás
A93B	a REM BASIC utasítás
A94B	az ON BASIC utasítás
A96B	egy BASIC sor címének keresése
A9A5	a LET BASIC utasítás
AA80	a PRINT# BASIC utasítás
AA86	a CMD BASIC utasítás
AAA0	a PRINT BASIC utasítás
AB1E	egy füzér kiírása
AB3E	az üresjel vagy a "cursor right" karakter kiírása
AB4D	hibakezelés inputnál
AB7B	a GET BASIC utasítás
ABA5	az INPUT# BASIC utasítás
ABBF	az INPUT BASIC utasítás
AC06	a READ BASIC utasítás
ACFC	"? extra ignored" és "? redo from start"
AD1D	a NEXT BASIC utasítás
AD8A	az FRMNUM beolvassa a kifejezést és numerikusan ellenőrzi
AD8D	numerikus ellenőrzés
AD8F	a füzér ellenőrzése
AD99	"type mismatch" kiírása
AD9E	az FRMEVL beolvas egy tetszőleges kifejezést és ezt kiértékeli
AE83	az aritmetikai kifejezés beolvasása
AEA8	π lebegőpontos állandó
AED4	a NOT BASIC utasítás
AEF1	egy zárójelek közé helyezett kifejezés beolvasása
AEF7	"bezáró zárójel" ellenőrzése
AEFA	"kezdő zárójel" ellenőrzése
AEFD	"vessző" ellenőrzése
AEFF	egy akkuban levő karakter ellenőrzése
AF08	a "syntax error" kiírása
AF28	egy változó beolvasása
AFE6	az OR BASIC utasítás
AFE9	az AND BASIC utasítás
B016	összehasonlítási műveletek

Cím	Leírás
B081	a DIM BASIC utasítás
B113	egy betű vizsgálata
B194	az első tömbelem mutatójának kiszámítása
B1A5	lebegőpontos állandó — 32768
B1AA	FAC/integer átalakítás
B245	a "bad subscript" kiírása
B248	az "illegal quantity" kiírása
B34C	a tömb méretének kiszámítása
B37D	a FRE BASIC függvény
B39E	a POS BASIC függvény
B3A6	a közvetlen üzemmód vizsgálata
B3AB	az "illegal direct" kiírása
B3AE	az "undef'd function" kiírása
B3B3	a DEF BASIC utasítás
B3E1	az FN szintaktikus ellenőrzése
B3Fe	az FN BASIC függvény
B465	az STRS BASIC függvény
B475	szövegkezelés, a szöveg mutatójának kiszámítása
B487	a füzér felépítése
B526	Garbage Collection, a felhasználatlanul maradt füzérek eltávolítása
B63D	füzérek összekapcsolása
B6A3	a FRESTR, füzérkezelés
B6EC	a CHR$ BASIC függvény
B700	a LEFT$ BASIC függvény
B72C	a RIGHT$ BASIC függvény
B737	a MID$ BASIC függvény
B77C	a LEN BASIC függvény
B782	a füzérparaméterek beolvasása
B78B	az ASC BASIC függvény
B79B	beolvas egy egy byte-os kifejezést (0-tól 255-ig)
B7AD	a VAL BASIC függvény
B7EB	beolvassa a címet (0-tól 65535-ig) és a byte értéket (0-tól 255-ig)
B7F7	FAC átalakítása címformátumra (0-tól 65535-ig)
B80D	a PEEK BASIC utasítás
B824	a POKE BASIC utasítás
B82D	a WAIT BASIC utasítás
B849	FAC = FAC + 0.5
B850	mínusz FAC = konstans (A/Y) — FAC
B853	mínusz FAC = ARG — FAC
B867	plusz FAC = konstans (A/Y) — FAC
B86A	plusz FAC = ARG + FAC
B97E	az "overflow" kiírása
B9BC	a LOG lebegőpontos állandói
B9EA	a LOG BASIC függvény
BA28	szorzás FAC =konstans (A/Y) * FAC
BA2B	szorzás FAC = ARG * FAC
BA8C	ARG = konstans (A/Y)
BAE2	FAC = FAC * 10
BAF9	lebegőpontos 10
BAFE	FAC = FAC/10
BB0F	FAC =konstans (A/Y)/FAC

Cím	Leírás
BB12	FAC = ARG/FAC
BB8A	"division by zero" kiadása
BBA2	FAC = konstans (A/Y)
BBC4	akku#4 = FAC
BBCA	akku#3 = FAC
BBD0	változó = FAC
BBFC	FAC = ARG
BC0C	ARG = FAC
BC1B	FAC kerekítése
BC2B	FAC előjelének beolvasása
BC39	az SGN BASIC függvény
BC58	az ABS BASIC függvény
BC5B	a konstans (A/Y) összehasonlítása a FAC-kel
BC9B	a FAC átalakítása egész számmá
BCCC	az INT BASIC függvény
BCF3	ASCII/lebegőpontos átalakítás
BDB3	a lebegőpontos állandók átalakítása ASCII-re
BDC2	a sorszám kiírása hibaüzenetnél
BDCD	egy pozitív egész szám kiírása (0-tól 65535-ig)
BDDD	a FAC átalakítása ASCII-re
BF11	0.5, lebegőpontos állandó
BF16	bináris számok a FAC ASCII-re alakításához
BF71	az SQR BASIC függvény
BF78	hatványozás FAC = konstans (A/Y) a FAC hatványkitevőn
BF7B	hatványozás FAC = ARG a FAC hatványkitevőn
BFBF	lebegőpontos állandók az EXP függvényhez
BFED	az EXP BASIC függvény
E043	a polinom kiszámítása
E059	a polinom kiszámítása
E08D	lebegőpontos állandók az RND függvényhez
E097	az RND BASIC függvény
E107	a "break" kiírása
E10C	BSOUT – egy karakter kiírása
E112	BASIN – egy karakter beolvasása
E118	CKOUT – az output egység kijelölése
E11E	CHKIN – az input egység kijelölése
D124	GETIN – egy karakter beolvasása
E12A	a SYS BASIC utasítás
E156	a SAVE BASIC utasítás
E165	a VERIFY BASIC utasítás
E168	a LOAD BASIC utasítás
E1BE	az OPEN BASIC utasítás
E1C7	a CLOSE BASIC utasítás
E1D4	a LOAD és a SAVE paramétereinek beolvasása
E219	az OPEN paramétereinek beolvasása
E264	a COS BASIC függvény
E26B	a SIN BASIC függvény
E2B4	a TAN BASIC függvény
E2E0	lebegőpontos állandók a SIN és a COS függvényekhez
E30E	az ATN BASIC függvény
E33E	az ATN függvény lebegőpontos állandói

Cím	Leírás
E37B	BASIC NMI beugras
E394	BASIC hidegindítás
E3A2	a CHRGET rutin másolata
E3BA	az RND függvény kezdőértéke
E3BF	a RAM inicializálása a BASIC-hez
E447	a BASIC vektorok táblázata
E453	a BASIC vektorok betöltése

7.3 A VC 20-as és a Commodore 64-es összehasonlító táblázata

C 64	VC 20	Leírás
E45F	E429	az operációs rendszer üzenetei
E4E0	–	várakozás a Commodore billentyűre
E4EC	–	RS232 timing állandói
E500	E500	CIA vagy VIA BASIC címeinek beolvasása
E505	E505	az oszlop/sor képernyőformátum beolvasása
E50A	E50A	a kurzor beállítása, vagy a kurzor helyzetének beolvasása
E518	E518	képernyő-reset
E544	E55F	a képernyő törlése
E566	E581	cursor home
E5A0	E5BB	a videovezérlő inicializálása
E5B4	E5CF	egy karakter beolvasása a billentyűzet-pufferből
E5CA	E5E5	billentyűzet-input várakozó ciklusa
E632	E64F	egy karakter beolvasása a képernyőről
E684	E6B8	az idézőjel vizsgálata
E6B6	E6EA	a sorkezdet MSB-jének kiszámítása
E8DA	E921	a színkód táblázat
E8EA	E975	a képernyőgörgetés
E9C8	EA56	a sor eltolása felfelé
E9FF	EA8D	a képernyősor törlése
EA1C	EAA1	a karakter és a szín beállítása a képernyőn
EA24	EAB2	a szín-RAM mutatójának kiszámítása
EA31	EABF	az interrupt rutin
EA87	EB1E	a billentyűzet lekérdezése
EB48	EBDC	a SHIFT-, a CTRL- és a Commodore billentyűk vizsgálata
EB79	EC46	billentyűzet-dekódolási táblázat, mutató
EB81	EC5E	dekódolási táblázatok
EC44	ED21	a vezérlőkarakter ellenőrzése
EC78	ED69	dekódolási táblázatok
ECB9	EDE4	a videóvezérlő állandói
ECE7	EDF	"load (r) run (cr)"
ECF0	EDFE	a képernyő LSB táblázatának kezdete
ED09	EE14	TALK küldése
EDOC	EE17	LISTEN küldése
ED40	EEE4	egy byte továbbítása az IEC buszra
EDB9	EEC0	másodlagos cím a LISTEN-hez
EDC7	EECE	másodlagos cím a TALK-hoz
EDEF	EEF6	UNTALK küldése
EDFE	EF04	UNLISTEN küldése

C 64	VC 20	Leírás
EE13	EF19	egy byte beolvasása az IEC buszról
EEB3	EF96	egy millimásodperces késleltetés
EEBB	EFA3	RS232-es output
EF4A	F027	az RS232-es adatbitek számának meghatározása
F014	FOED	kiírás az RS232-es pufferébe
F086	F14F	GET az RS232-esből
F0A4	F160	az IEC time out beállítása
FODB	F174	az operációs rendszer hibaüzenetei
F12B	F1E0	az üzenetek kiírása
F157	F20E	BASIN – egy karakter beolvasása
F1CA	F27A	BSOUT – egy karakter kiírása
F20E	F2C7	CHKIN – az input egység kijelölése
F250	F309	CKOUT – az output egység kijelölése
F291	F34A	CLOSE
F30F	F3CF	a logikai file-szám keresése
F31F	F3DF	a file paramétereinek beállítása
F32F	F3EF	CLALL, lezárja az összes I/O csatornát
F333	F3F3	CLRCH, lezár egy I/O csatornát
F34A	f40A	OPEN
F49E	F542	LOAD
F5AF	F647	"searching for filename" kiírása
F5D2	F66A	"loading/verifying" kiírása
F5DD	F675	SAVE
F68F	F728	"saving filename" kiírása
F69B	F734	UDTIM futási idő növelése
F6DD	F760	az idő beolvasása
F6E4	F767	az idő beállítása
F6ED	F770	a STOP billentyű lekérdezése
F6FB	F77E	az operációs rendszer hibaüzeneteinek kiírása
F72C	F7AF	a programfej beolvasása a szalagról
F76A	F7E7	a fej kiírása a szalagra
F7D0	F84D	a szalagpuffer kezdőcímének beolvasása
F7D7	F854	a szalagpuffer kezdő és végcímének beállítása
F7EA	F867	a szalagfej név szerinti keresése
F80D	F88A	a szalagpuffer mutatójának növelése
F817	F894	várakozás a kazettás egység billentyűjére olvasásnál
F82E	F8AB	a kazetta-billentyű lekérdezése
F838	F8B7	várakozás a kazettás egység billentyűjére írásnál
F841	F8C0	egy blokk beolvasása szalagról
F84A	F8C9	a program betöltése szalagról
F864	F8EA	a szalagpuffer felírása a szalagra
F86B	F8EA	a blokk vagy a program felírása a szalagra
F8BE	F92F	várakozás az I/O művelet végére
F8E1	F94B	a STOP billentyű vizsgálata
F92C	F98E	megszakító rutin szalagról olvasásnál
FB97	F8DB	soros input bitszámlálójának beállítása
FBA6	FBEA	egy bit felírása a szalagra
FBCD	FC0B	megszakító rutin szalagra írás közben
FCB8	FCF6	az IRQ vektor beállítása
FCCA	FD08	a meghajtómotor kikapcsolása

C 64	VC 20	Leírás
FCD1	FD11	a végcím elérésének ellenőrzése
FCDB	FD B	a címmutató növelése
FCE2	FD22	RESET
FD02	FD3F	ROM ellenőrzése $8000-nél vagy $A000-nál
FD10	FD4D	a ROM modul azonosítása
FD15	FD52	a hardver és I/O vektor táblázat
FD30	FD6D	táblázat a haruver és az I/O vektorhoz
FD50	FD8D	az operatív tár inicializálása
FD8B	FD6D	az IRQ vektorok táblázata
FDF9	FE49	a file-név paramétereinek beállítása
FE00	FE50	az aktív állomány paramétereinek beállítása
FE07	FE57	a státusz beolvasása
FE18	FE66	az "operációs rendszer üzenetei" kapcsoló beállítása
FE1C	FE6A	a státusz beállítása
FE21	FEF6	az IEC busz timeout kapcsolójának beállítása
FE25	FE73	a RAM felsőhatár beállítása, ill. beolvasása
FE34	FE82	a RAM alsóhatár beállítása, ill. beolvasása
FE43	FEA9	NMI rutin
FEC2	FF5C	az RS232 átviteli sebességének állandói (baud-állandók)
FF48	FF72	megszakításkezelés
FF81	FF8A	az operációs rendszer rutinjainak ugrási táblázata

7.4 A CBM 8000-es és a Commodore 64-es összehasonlítási táblázata

Az alábbi táblázat a CBM 8000-es és a C 64-es gépek BASIC értelmezőit veti össze a megfelelő címeik összehasonlításával.
A táblázat alapján a 8000-es Commodore sorozat gépeire írt gépi kódu programokat könnyen átírhatjuk a Commodore 64-esre.

8000	64	Értelmezés
B000	A00C	a BASIC utasítások címe (mínusz 1)
B066	A052	a BASIC függvények címei
B094	A080	a BASIC operátorok hierarchia-kódjai és címei
B0B2	A09E	a BASIC utasításszavak jegyzéke
B20D	A19E	a BASIC hibaüzenetek
B322	A38A	a FOR NEXT és a GOSUB utasítás veremkereső rutinja
B350	A3B8	blokk-eltoló rutin
B393	A3FB	szabad hely keresése a veremben
B3CD	A435	"out of memory" kiírása
B3CF	A437	a hibaüzenet kiírása
B3FF	A474	ready-üzemmód
B406	B480	input várakozó ciklus
B4B6	A533	a BASIC programsorok újraláncolása
B4E2	A560	egy sor beolvasása az input pufferbe
B4FB	A579	egy sor átalakítása interpreter-kódokká
B5A3	A613	egy BASIC sor keresése
B5D2	A642	a NEW BASIC utasítás
B5EE	A65E	a CLR BASIC utasítás

8000	64	Értelmezés
B622	A68E	a programmutató beállítása a BASIC kezdetre
B630	A69C	a LIST BASIC utasítás
B6B5	A717	az interpreter-kódok átalakítása utasításszavakká
B6DE	A742	a FOR BASIC utasítás
B74A	A74E	az interpreter ciklus, egy BASIC utasítás végrehajtása
B785	A7ED	egy BASIC utasítás végrehajtása
B7B7	A81D	a RESTORE BASIC utasítás
B7C6	A82C	a STOP és az END
B7EE	A857	a CONT BASIC utasítás
B808	A871	a RUN BASIC utasítás
B813	A883	a GOSUB BASIC utasítás
B830	A8A0	a GOTO BASIC utasítás
B85D	A8D2	a RETURN BASIC utasítás
B883	A8F8	a DATA BASIC utasítás
B891	A906	a következő utasítás keresése
B894	A909	a következő sor keresése
B8B3	A92B	az IF BASIC utasítás
B8C6	A93B	a REM BASIC utasítás
B8D6	A94B	az ON BASIC utasítás
B8F6	A96B	egy BASIC sor címének keresése
B930	A9A5	a LET BASIC utasítás
BA88	AA80	a PRINT# BASIC utasítás
BA8E	AA86	a CMD BASIC utasítás
BAA8	AAA0	a PRINT BASIC utasítás
BB1D	AB1E	egy füzér kiírása
BB4C	AB4D	hibakezelés bevitelnél
BB7A	AB7B	a GET BASIC utasítás
BBA4	ABA5	az INPUT# BASIC utasítás
BBBE	ABBF	a READ BASIC utasítás
BCF7	ACFC	"? extra ignored" és "? redo from tart"
BD19	AD1D	a NEXT BASIC utasítás
BD84	AD8A	az FRMNUM, beolvas egy kifejezést és ezt numerikusan ellenőrzi
BD87	AD8D	numerikus ellenőrzés
BD89	AD8F	a füzér ellenőrzése
BD93	AD99	a "type mismatch" kiírása
BD98	AD9E	az FRMEVL, egy tetszőleges kifejezés beolvasása és kiértékelése
BDD1	AE83	egy kifejezés következő elemének beolvasása
BEA0	AEA8	π, lebegőpontos állandó
BEE9	AEF1	egy zárójelek közé helyezett aritmetikai kifejezés beolvasása
BEEF	AEF7	a bezáró zárójel ellenérzése
BEF2	AEFA	a nyitó zárójel ellenőrzése
BEF5	AEFD	a vessző ellenőrzése
BEF7	AEFF	az akkumulátorban levő karakter vizsgálata
BF00	AF08	a "syntax error" kiírása
BF0C	AF28	egy változó beolvasása
C086	AFE6	az OR BASIC művelet
C089	AFE9	az AND BASIC művelet
C0B6	B016	összehasonlítási műveletek
C121	B081	a DIM BASIC utasítás
C1B6	B113	egy betű vizsgálata

8000	64	Értelmezés
C2D9	B1A5	– 32768, lebegőpontos állandó
C2DD	B1AA	FAC/integer átalakítás
C370	B245	"bad subscript" kiírása
C373	B248	"illegal quantity" kiírása
C477	B34C	a tömb méretének kiszámítása
C4A8	B37D	a FRE BASIC függvény
C4C9	B39E	a POS BASIC függvény
C4CF	B3A6	a közvetlen üzemmód vizsgálata
C4DC	B3B3	a DEF BASIC utasítás
C50A	B3E1	az FN szintetikus ellenőrzése
C51D	B3F4	az FN BASIC függvény
C58E	B465	az STRS BASIC függvény
C59E	B475	füzérkezelés, a füzér mutatójának kiszámítása
C5B0	B487	a füzér értékének beállítása
C66A	B526	Garbage collection
C74F	B63D	füzérek összekapcsolása
C7B5	B6A3	füzérkezelés, FRESTR
C822	B6EC	a CHR$ BASIC függvény
B836	B700	a LEFT$ BASIC függvény
C862	C72C	a RIGHT$ BASIC függvény
C86D	B737	a MID$ BASIC függvény
C8B2	B77C	a LEN BASIC függvény
B8B8	B782	a füzérparaméterek beolvasása
C8C1	B78B	az ASC BASIC függvény
C8D1	B78B	egy egy byte-os kifejezés beolvasása (0–255-ig)
C8E3	B7AD	a VAL BASIC függvény
C921	B7EB	a cím és a byte értékének beolvasása
C92D	B7F7	a FAC átalakítása címformátumra (0–65535-ig)
C943	B80D	a PEEK BASIC függvény
C95A	B824	a POKE BASIC utasítás
C963	B82D	a WAIT BASIC utasítás
C97F	B849	FAC = FAC + 0.5
C986	F850	mínusz FAC = konstans (A/Y) – FAC
C989	B853	mínusz FAC = ARG – FAC
C99D	B867	plusz FAC = konstans (A/Y) + FAC
C9A0	B86A	plusz FAC = ARG + FAC
CAB4	B97E	"overflow" kiírása
CAF2	B9BC	a LOG függvény lebegőpontos állandói
CB20	B9EA	a LOG BASIC függvény
CB5E	BA28	szorzás FAC = konstans (A/Y) * FAC
CB61	BA2B	szorzás FAC = ARG * FAC
CBC2	BA8C	ARG = konstans (A/Y)
CC18	BAE2	FAC = FAC * 10
CC2F	BAF9	10, lebegőpontos állandó
CC34	BAFE	FAC = FAC/10
CC45	BB0F	osztás FAC = konstans (A/Y)/FAC
CC4A	BB12	osztás FAC = ARG/FAC
CCC0	BB8A	"division by zero" kiírása
CCD8	BBA2	FAC = konstans (A/Y)
CCFD	BBC4	akku# 4 = FAC

8000	64	Értelmezés
CD03	BBCA	akku#3 = FAC
CD0A	BBD0	változó = FAC
CD32	BBFC	FAC = ARG
CD42	BC0C	ARG = FAC
CD51	BC1B	a FAC kerekítése
CD61	BC2B	a FAC előjelének beolvasása
CD6F	BC39	az SGN BASIC függvény
CD8E	BC58	az ABS BASIC függvény
CD91	BC5B	konstans (A/Y) összehasonlítása a FAC-kel
CDD1	BC9B	FAC/integer konvertálás
CE02	BCCC	az INT BASIC függvény
CE29	BCF3	ASCII/FAC konvertálás
CEE9	BDB3	lebegőpontos állandók átalakítása ASCII-re
CF78	BDC2	a sorszám kiírása hibaüzenetnél
CF93	BDDD	FAC/ASCII átalakítása
D0C2	BF11	0.5, lebegőpontos állandó
D0C7	BF16	bináris számok a FAC/ASCII átalakításhoz
D108	BF71	a SQR BASIC függvény
D112	BF78	hatványozás = konstans (A/Y) az FAC-edik hatványon
D115	BF7B	hatványozás = ARG az FAC-edik hatványon
D156	BFBF	az EXP függvény lebegőpontos állandói
D184	BFED	az EXP BASIC függvény
D1D7	E043	polinomszámítás
D1ED	E059	polinomszámítás
D221	E08D	az RND függvény lebegőpontos állandói
D229	E097	az RND BASIC függvény
D282	E264	a COS BASIC függvény
D289	E26B	a SIN BASIC függvény
D2D2	E2B4	a TAN BASIC függvény
D2FE	E2E0	a SIN és a COS függvény lebegőpontos állandói
D32C	E30E	az ATN BASIC függvény
D35C	E33E	az ATN függvény lebegőpontos állandói
D399	E3A2	a CHRGET rutin másolata

7.5 A VC 20-ason készített programok atalakítasa Commodore 64-esre

Már a külső megjelenési forma alapján is felismerhető, hogy a Commodore 64-es a VC 20-as idősebb fivére. Emögött azonban sokkal több rejlik, mint a „fiatalabb" és az „idősebb" testvér egymás közöttí megkülönböztetése. A Commodore 64-es abban különbözik a VC 20-astól, hogy nagyfelbontású színes grafikát biztosít, sprite-ok előállítására és mozgatására képes, s egy szintetizátor segítségével hang előállítására is alkalmas. Ez azonban még korántsem jelenti azt, hogy végérvényesen le kellene mondanunk a VC 20-as szoftverekről.
A VC 20-as gépi moduljainak és kazettáinak egyszerű átvételére sajnos nincs lehetőség. Pedig biztosan sokan rendelkeznek olyan VC 20-as programokkal, amelyeket szívesen átvennének a C 64-re. A programok adaptálási módját elsősorban a program nyelve szabja meg.

1) A gépi kódú programok

Bár ezeknél a programoknál az adaptáció nem nehéz, mégis bonyolultabb, mint az egyszerű BASIC programok esetében A gépi kódú programok adaptálásához ajánlatos az összehasonlítási táblázatot áttanulmányozni (lásd a 7.1 fejezetet). Ha például egy olyan ugrási címet találunk, amely egy ROM-beli címre utal, ezt a címet meg kell keresnünk a Commodore 64-es összehasonlítási táblázatában, hogy a programban felcserélhessük a VC 20-as címével.
Az egyes címek közötti különbséget általában saját magunk is kiszámíthatjuk.
Igy például a VC 20-asnál a BASIC GET rutin kezdőcíme a hexadecimális CB7B. A Commodore 64-esnél ez a cím pontosan 2000 hexadecimális számmal alacsonyabb, vagyis $AB7B. Az E000-tól kezdődően a Commodore 64-esnél 3-at kell levonni, hogy a megfelelő VC 20-as címet megkaphassuk.

2) A BASIC programok

A BASIC programoknál a módosításokat sokkal egyszerűbben elvégezhetjük. A módosításnál főként az eltérő képernyőformátumra kell ügyelni. Itt arra figyeljünk, hogy a VC 20-as programok képernyőformátumát vagy teljesen meg kell változtatni, vagy minden 22 karakter után egy RETURN karaktert kell beszúrni, nehogy más karakterek csússzanak be ugyanabba a sorba.
Az adaptálás során az egyetlen nehézséget a POKE utasítások jelentik. A POKE utasításokat a hivatkozott tárcímek alapján meg kell különböztetnünk egymástól:

a) POKE utasítás a képernyőtárban
Itt teljes címet kell megváltoztatni. Hasonlítsuk össze a karaktergenerátor- és a képernyőoldal-táblázatokat.
b) A VC 20-as gép ROM-jára vonatkozó értékek
Itt pontosan ugyanúgy kell eljárnunk, mint a gépi kódú programok előzőekben ismertetett módosításánál.

7.6 CBM programok átalakítása a Commodore 64-esre

Bizonyára sokan vannak, akiknek még fel sem tűnt, hogy milyen nagy hasonlatosság van a Commodore 64-es és a nagyobb CBM gépek között. A hasonlóság alapot teremt arra, hogy az eltérő típusú gépek (pl. a PET) programjait átvegyük a C 64-esre. A legtöbb gép egy és ugyanazon képernyőformátummal dolgozik. Természetesen itt sem nélkülözhetünk bizonyos változtatásokat. Óriási előnyt jelent viszont az a tény, hogy a BASIC programokat teljes egészében átvehetjük, ha nem tartalmaznak POKE utasításokat.
A POKE utasításokban meg kell változtatnunk a hivatkozott címeket, ha azok eltérnek a Commodore 64-estől. A PET/CBM gépekről széleskörű szakirodalom áll rendelkezésre. A gyártó által publikált szakirodalmi jegyzékben megtalálhatók a megfelelő kiadványok.
Általában azt mondhatjuk, hogy ezeknél a gépeknél a BASIC és a gépi kódú nyelvben nincs eltérés. A 6510-es processzor teljesen kompatibilis a 6502-essel. Csak az operációs rendszerben levő rutinok beugrási címei, a képernyőtár címei és a nulláslap vektorai különböznek.
Mindezek figyelembevételével, ha már megfelelő gyakorlattal rendelkezünk, a PET/CBM/VC-20 gépek összes programját minden nehézség nélkül átírhatjuk a Commodore 64-es gépre. Ugyanakkor az adaptált programok bővítésével a Commodore 64-es színeit, hangját és grafikáját is kihasználhatjuk.
Lehet, hogy az átalakított program sokkal hasznosabb lesz, mint az eredeti volt.
Széljegyzetként még egy ötlet: a VC 20-as kazettákat csak akkor tudjuk beolvasni a C 64-esbe,

ha ezeket előzőleg betöltöttük egy CBM gépbe, majd ismét tároltuk szalagon. A Commodore-64-es képes egy CBM gép szimulálására is, a következő program segítségével:

```
P46

10 POKE 56576,6   : REM KEPERNYO ATHELYEZESE 32768 CIMRE
20 POKE 53272,4
30 POKE 648,128   : REM BASIC ERTESUL A KEPERNYO UJ HELYZETEROL
40 POKE 1024,0:POKE 44,4:POKE 56,128 :REM BASIC INDITAS/VEGE
50 PRINT CHR$(147)
60 NEW
```


8. FEJEZET
A COMMODORE 64-ES ROM LISTÁJA

8.1 A Commodore 64-es ROM lista hasznos tulajdonságai

A most következő több mint 100 oldalon található az egész operációs rendszer és a Commodore 64-es BASIC interpreterjének assembler listája. Ha valaki hajlandó beleélni magát a gép lelkébe, a lista tanulmányozásával határtalan ösztönzést kaphat az assembler szintű programozáshoz. A legnagyobb hasznot hajtó tevékenység azonban az, ha az operációs rendszer és a BASIC interpreter rutinjait alprogramként használjuk a saját programjainkban. Ezzel sok programozási munkát takaríthatunk meg.
Miként találhatjuk meg leggyorsabban a szükséges rutinokat?
Az összes BASIC utasítás címe megtalálható a BASIC interpreter elején az ugrási táblázatban.
Ha például a GOTO utasítást szeretnénk közelebbről megvizsgálni, meg kell keresnünk az ugrási táblázatban a $A8AO címet. Ez a tulajdonképpeni utasítás címe. Ha lebegőpontos aritmetikai műveleteket akarunk végezni (amelyhez ismernünk kell a változók tárolási módjait), a szükséges rutinokat a $B800 és az ezt követő címeken találjuk.
Hasznos, ha ehhez az 5. fejezet tartalmát is áttanulmányozzuk. Az operációs rendszer a $E400 címen kezdődik. Itt van a képernyőszerkesztő, majd ezt követi a billentyűzetet és a megszakítást kezelő rutin. A lista a soros IEC busz input és output rutinjaival, az RS232-es illesztő és a kazettás egység kezelésével folytatódik. A legfontosabb rutinokat egy ugrási táblázat foglalja össze, a $FF81 címtől kezdődően, a ROM végén. A 7. fejezet mégegyszer ismerteti a beugrási pontokat.

Az alkalmazás lehetőségét nézzük meg egy példán keresztül.
A gépi kódú programozás során gyakran találjuk magunkat szemben azzal a feladatattal, hogy egy tármezőt, ami lehet például egy grafikus kép vagy egy táblázat, el kell tolnunk.
Ha az operációs rendszer rutinjaira támaszkodunk, az egész feladat csak a paraméterek átadására és az alprogram behívására korlátozódik. A beugrási cím $A3BF. Tegyük fel, hogy a $24BA – $29CO tartományt (a $29CO-t is beleértve) úgy akarjuk eltolni, hogy $3000-nél végződjék. Ehhez a korábbi blokk-kezdet és blokkvég címét, valamint az új blokkvég címét át kell adnunk.
A megoldás tehát a következő:

```
P47.ASS

033C                100         * = $033C
033C                110         ;
033C A9 BA          120         LDA #$BA
033E A0 24          130         LDY #$24    ; MUTATO A REGI BLOKK-KEZDETRE
0340 85 5F          140         STA $5F
0342 84 60          150         STY $60
0344 A9 C1          160         LDA #$C1
0346 A0 29          170         LDY #$29    ; MUTATO A REGI BLOKK-VEGE+1-RE
0348 85 5A          180         STA $5A
034A 84 5B          190         STY $5B
034C A9 01          200         LDA #$01
034E A0 30          210         LDY #$30  ; MUTATO AZ UJ BLOKK-VEGE+1-RE
0350 85 58          220         STA $58
0352 84 59          230         STY $59
0354 20 BF A3       240         JSR $A3BF ; ELTOLORUTIN INDITASA
0357              65535         .END
```

Ezzel a rutinnal egy tetszőleges tármezőt magasabb címre csúsztathatunk. A BASIC interpreternek azért van szüksége erre a rutinra, hogy egy teljes tömbmezőt felfelé eltolhasson, ha a régiek közé egy új változót kell behelyeznie.
Az operációs rendszer rutinjainak felhasználásával nemcsak időt és fáradtságot takaríthatunk meg.
Az így felépített programok sokkal rövidebbek lesznek, tehát a felszabadult tárterületekkel szabadon gazdálkodhatunk.
A következő oldalakon táblázatba foglalva közöljük a rutinok megnevezését és címét, hogy az Olvasó számára megkönnyítsük az eligazodást a ROM listában.

8.2 A ROM rutinok jegyzéke

Lenyomott szalag-billentyű lekérdezése	$F82E
Egy tömbelem címének kiszámítása	$B30E
A BASIC utasítások címei (mínusz 1)	$A00C
A BASIC függvények címei	$A052
A hibaüzenetek címei	$A328
A címmutató növelése	$FCDB
Az RND függvény kezdőértéke	$E3BA
Az operatív tár inicializálása	$FD50
A tömbelem keresése	$B2E9
A tömbváltozó beszúrása	$B261
A sorszám kiírása hibaüzenetnél	$BDC2
Egy kérdőjel kiírása	$AB45
Egy üres jel kiírása	$AB3B
Output az RS232-es pufferbe	$F014
ARG = konstans (A/Y)	$BA8C
ARG átvitele a FAC-be	$BBFC
ASCII kód/képernyőkód konvertálás	$E691
A szalag előkészítése olvasásra	$F8E2
A szalagfej név szerinti keresése	$F7EA
A szalagpuffer felírása a szalagra	$F864
A szalagpuffer-mutató növelése	$F80D
CKOUT BASIC rutin	$E4AD
BASIC hidegindítás	$E394
NMI BASIC beugrás	$E37B
CLOSE BASIC utasítás	$E1C7
CLR BASIC utasítás	$A65E
CMD BASIC utasítás	$AA86
CONT BASIC utasítás	$A857
DATA BASIC utasítás	$A8F8
DEF BASIC utasítás	$B3B3
DIM BASIC utasítás	$B081
END BASIC utasítás	$A831
FOR BASIC utasítás	$A742
GET BASIC utasítás	$AB7B
GOSUB BASIC utasítás	$A883
GOTO BASIC utasítás	$A8A0
IF BASIC utasítás	$A928
INPUT BASIC utasítás	$ABBF
INPUT# BASIC utasítás	$ABA5

LET BASIC utasítás	$A9A5
LIST BASIC utasítás	$A69C
LOAD BASIC utasítás	$E168
NEW BASIC utasítás	$A642
NEXT BASIC utasítás	$AD1D
ON BASIC utasítás	$A94B
OPEN BASIC utasítás	$E1BE
POKE BASIC utasítás	$B824
PRINT BASIC utasítás	$AAA0
PRINT# BASIC utasítás	$AA80
READ BASIC utasítás	$AC06
REM BASIC utasítás	$A93B
RESTORE BASIC utasítás	$A81D
RETURN BASIC utasítás	$A8D2
RUN BASIC utasítás	$A871
SAVE BASIC utasítás	$E156
STOP BASIC utasítás	$A82F
SYS BASIC utasítás	$E12A
VERIFY BASIC utasítás	$E165
WAIT BASIC utasítás	$B82D
BASIC utasításszavak	$A09E
BASIC hibaüzenetek	$A19E
ABS BASIC függvény	$BC58
ASC BASIC függvény	$B78B
ATN BASIC függvény	$E30E
CHF$ BASIC függvény	$B6EC
COS BASIC függvény	$E264
EXP BASIC függvény	$BFED
FN BASIC függvény	$B3F4
FRE BASIC függvény	$B37D
INT BASIC függvény	$BCCC
LEFT$ BASIC függvény	$B700
LEN BASIC függvény	$B77C
LOG BASIC függvény	$B9EA
MID$ BASIC függvény	$B737
PEEK BASIC függvény	$B80D
POS BASIC függvény	$B39E
RIGHT$ BASIC függvény	$B72C
RND BASIC függvény	$E097
SGN BASIC függvény	$BC39
SIN BASIC függvény	$E26B
SQR BASIC függvény	$BF71
STR$ BASIC függvény	$B465
TAN BASIC függvény	$E2B4
VAL BASIC függvény	$B7AD
BASIC kód/szöveg konvertálás	$A717
AND BASIC művelet	$AFE9
NOT BASIC művelet	$AED4
OR BASIC művelet	$AFE6
BASIN BASIC rutin	$E112
BSOUT BASIC rutin	$E10C
CHKIN BASIC rutin	$E11E
CKOUT BASIC rutin	$E118

GETIN BASIC rutin	$E124
Egy BASIC utasítás végrehajtása	$E7ED
BASIC vektorok betöltése	$E453
A CIA báziscímének beolvasása	$E500
Az operációs rendszer üzenetei	$E45F
A képernyő törlése	$E544
A képernyő görgetése	$E8EA
Képernyő-reset	$E518
A képernyőformátum beolvasása	$E505
A képernyősor törlése	$E9FF
A bit felírása a szalagra	$FBA6
Bitenkénti szorzás	$BA59
A bitszámláló beállítása soros kihozatalra	$FB97
Egy blokk beolvasása szalagról	$F841
Blokkeltoló rutin	$A3B8
Egy byte kiírása a soros buszra	$ED40
Egy byte kiírása a soros buszra	$EDDD
Egy byte beolvasása a soros buszról	$EE13
Egy byte beolvasása a szalagról	$F199
Egy byte beolvasása az RS232-esről	$F1B8
Egy byte bevitele az X-be, GETBYT	$B79B
A BASIN rutin	$F157
A BSOUT rutin	$F1CA
A kurzor beállítása/beolvasása	$E50A
Cursor home	$E566
A kurzorpozíció kiszámítása	$E56C
CHKIN rutin	$F20E
CKOUT rutin	$F250
CLALL rutin	$F32F
CLOSE rutin	$F291
CLRCH rutin	$F333
Az RS232-es adatbitjeinek kiszámítása	$EF4A
Dimenzionált változó beolvasása	$B1D1
Osztás FAC = ARG/FAC	$BB12
Osztás FAC = konstans (A/Y)/FAC	$BB0F
Egy folytató sor betoldása	$E965
Egy sor bevitele	$A560
Input várakozási ciklus	$A480
Hibakezelés adatbevitelnél	$AB4D
A hibaüzenet kiadása	$A437
Az operációs rendszer hibaüzenete	$F6FB
A file-paraméterek beállítása	$F31F
A „rendszerüzenet" kapcsoló beállítása	$FE18
– 32768 lebegőpontos állandó	$B1A5
0.5 lebegőpontos állandó	$BF11
10 lebegőpontos állandó	$BAF9
FAC = FAC * 10	$BAE2
FAC = FAC + 0.5	$B849
FAC = FAC/10	$BAFE
FAC = konstans (A/Y)	$BBA2
FAC átvitele az akku# 3-ba	$BBCA
FAC átvitele az akku# 4-be	$BBC7
FAC átvitele az ARG-be	$BC0C

FAC/ASCII konvertálás és a $100-ba	$BDDD
FAC átvitele változóba	$BBD0
FN szintaktikus ellenőrzése	$B3E1
FRESTR	$B6A3
FRMEVL egy tetszőleges kifejezés kiértékelése	$AD9E
FRMNUM kifejezés beolvasása és numerikus ellenőrzése	$AD8A
Garbage Collection	$B526
GETADR és GETBYT, 16- és 8-bites érték beolvasása	$B7EB
GETADR, FAC átalakítása 16-bites számmá	$B7F7
GETIN rutin	$F13E
A hardver és az I/O vektorok beállítása/beolvasása	$FD15
A fejléc felírása a szalagra	$F76A
BASIC műveletek hierarchia-kódjai	$A080
Tömbszámításhoz szükséges segédrutin	$B34C
A háttérszín beállítása	$E4DA
Az interpreter ciklus	$A7AE
Az interrupt rutin	$EA31
A szalagról történő olvasás megszakító-rutinja	$F92C
A szalagra történő írás megszakítórutinja	$FBCD,$FC6A
IRQ beugrás	$FF48
Az IRQ vektor beállítása	$FCB8
IRQ vektorok	$FD98
A Kernal ugrási táblázat	$FF81
π konstans	$AEAD
Az ATN függvény konstansai	$E33E
Az EXP függvény konstansai	$BFBF
Lebegőpontos/ASCII átalakítás konstansai	$BF16
Lebegőpontos/ASCII átalakítás konstansai	$BDB3
A LOG függvény konstansai	$B98C
Az RND függvény konstansai	$E08D
A SIN és a COS függvény konstansai	$E2E0
A TI/TI$ konvertáláshoz szükséges konstansok	$BF3A
A CHRGET rutin másolata	$E3A2
LISTEN küldése	$ED0C
A logikai file-szám keresése	$F30F
Programsorok törlése és betoldása	$A49C
LOAD rutin	$F49E
FAC mantisszájának invertálása	$B947
Az operációs rendszer üzenetei	$F0BD
Az operációs rendszer üzeneteinek kiírása	$F12B
Az interpreter üzenetei	$A364
Mínusz FAC = ARG – FAC	$B853
Mínusz FAC = konstans (A/Y) – FAC	$B850
Szorzás FAC = ARG * FAC	$BA2B
Szorzás FAC = konstans (A/Y) * FAC	$BA28
Sorkezdetek MSB-inek újrakiszámítása	$E6B6
A következő sor keresése	$A909
Egy kifejezés következő elemének beolvasása	$AE83
A következő utasítás keresése	$A906
NMI beugras	$FE43
Az RS232-es NMI rutinja	$FED6
A RAM felső határának beállítása/beolvasása	$FE25
OPEN rutin	$F34A

Az aktív file paramétereinek beállítása	$FE00
A file-név paramétereinek beállítása	$FDF9
A LOAD és a SAVE utasítás paramétereinek beolvasása	$E1D4
Az OPEN utasítás paramétereinek behozása	$E219
Helyfenntartás a füzér számára	$B4F4
Plusz FAC = ARG + FAC	$B86A
Plusz FAC = konstans (A/Y) + FAC	$B867
1. polinomszámítás	$E043
2. polinomszámítás	$E059
Pozitív egész szám kiírása az A/X-be	$BDCD
Hatványozás FAC = ARG az FAC-edik hatványon	$B57B
Hatványozás FAC = ARG a konstans-adik hatványon (A/Y)	$BF78
Program betöltése szalagról	$F84A
Programfej olvasása szalagról	$F72C
„BASIC-start" programmutató	$A68E
Programsor betoldása	$A4ED
Programsor törlése	$A4A9
Programsorok újraláncolása	$A533
Numerikus ellenőrzés	$AD8D
Auto-Start-ROM ellenőrzés	$FD02
Betű ellenőrzés	$B113
A végcím elérésének ellenőrzése	$FCD1
A megnyitó zárójel ellenőrzése	$AEFA
A bezáró zárójel ellenőrzése	$AEF7
A vessző ellenőrzése	$AEFD
A szabad tárhely vizsgálata	$A3FB
A SHIFT, CTRL, COMMODORE ellenőrzés	$EB48
A vezérlőkarakterek vizsgálata	$EC44
A STOP billentyű ellenőrzése	$A82C
A füzér ellenőrzése	$AD8F
A rendszerváltozó ellenőrzése	$AF14
Összehasonlítás az aktuális karakterrel	$AEFD
Egy regiszter jobbraléptetése	$B983
A kazettás egység meghajtómotorjának kikapcsolása	$FCCA
Reset rutin	$FCE2
A BASIC RAM inicializálása	$E3BF
A ROM modul azonosítása	$FD10
RS232-es output	$EEBB
RS232-es output	$F208
RS232-es CHKIN	$F04D
RS232-es GET	$F086
Helyfoglalás a tárban	$A408
Másodlagos cím küldése a LISTEN-hez	$ED89
Másodlagos cím küldése a TALK-hoz	$EDC7
Veremkereső rutin	$A38E
Szalagpuffer kezdőcímének beolvasása	$F7D0
Egy programsor kezdőcímének kiszámítása	$A613
A státusz beolvasása	$FE07
A STOP billentyű lekérdezése	$F6ED
A szöveg kiírása	$AB1E
A szöveg beolvasása, mutató az A/Y-ba	$B487
A szöveg átvitele a fenntartott mezőbe	$B67A
A szövegparaméterek behozása	$B782

A szövegparaméterek beolvasása a veremből	$B761
Szövegek összehasonlítása	$B02E
Szövegek összekapcsolása	$B63D
Szövegkezelés, FRESTR	$B6A3
A szövegmutató kiszámítása	$B475
A SAVE rutin	$F5DD
A BASIC vektorok táblázata	$E447
A színkódok táblázata	$E8DA
A hardver és az I/O vektorok táblázata	$FD30
A képernyősor kezdetek LSB-táblázata	$ECF0
TALK küldése	$ED09
1. billentyűzet-dekódolási táblázat	$EB81
2. billentyűzet-dekódolási táblázat	$EBC2
3. billentyűzet-dekódolási táblázat	$EC03
4. billentyűzet-dekódolási táblázat	$EA78
Billentyűlekérdezés	$EA87
Zárójelek közé helyezett kifejezés beolvasása	$AEF1
A közvetlen üzemmód vizsgálata	$B3A6
Az idézőjel vizsgálata	$E684
A STOP billentyű vizsgálata	$F8D0
Az idő növelése	$F69B
Az idő beolvasása	$F6DD
Az idő beállítása	$F6E4
A soros busz timeout kapcsolójának beállítása	$FE21
Az RS232-es átviteli sebességének óra-állandói (NTSC változat)	$FEC2
Az RS232-es átviteli sebességének óra-állandói (PAL változat)	$E4EC
Egy sor átalakítása interpreter kódokká	$A579
ASCII átalakítása lebegőpontossá	$BCF3
Lebegőpontos formátum átalakítása egésszé	$B1B2
Lebegőpontos formátum átalakítása egésszé	$BC9B
UNLISTEN küldése	$EDFE
UNTALK küldése	$EDEF
A RAM alsó határának beállítása/beolvasása	$FE34
A változó tárolása	$B11D
A változó beolvasása	$AF28
A változó beolvasása	$B08B
Összehasonlítás	$B016
Konstans (A/Y) összehasonlítása a FAC-kel	$BC5B
A videovezérlő inicializálása	$E5A0
A FAC előjelének beolvasása	$BC2B
Várakozás a szalag billentyűre	$F817
Várakozás a szalag billentyűre írásnál	$F838
Várakozás a Commodore billentyűre	$E4E0
Billentyűzet-bevitel, várakozó ciklus	$E5CA
Füzér értékadása	$AA2C
Egész értékadás (értékhozzárendelés)	$A9C4
Valós értékadás	$A9D6
Füzér értékadás	$A9D9
Egy karakter kiírása a képernyőre	$E716
Egy karakter ellenőrzése, számjegy?	$AA1D
Egy karakter beolvasása a billentyűzet-pufferből	$E5B4
A karakter és a szín beállítása a képernyőn	$EA1C
A karakter beolvasása a képernyőről	$E632

Az első tömbelem mutatójának kiszámítása	$B194
A szín-RAM mutatójának kiszámítása	$EA24
Billentyűzet-dekódolási táblázat mutatója	$EB79
Sortolás felfelé	$E9C8
A sorszám beolvasása és konvertálása címformátummá	$A96B
Az idő beolvasása	$AF84

8.3 A Commodore 64 ROM listája

```
*****************************
A000  94 E3                     Start-vektor $E394
A002  7B E3                     NMI-vektor   $E37B

*****************************
A004  43 42 4D 42 41 53 49 43   "cbmbasic"

*****************************  BASIC-utasítások címei /minusz 1/
A00C  30 A8           $A8Ø      $A831   END
A00E  41 A7           $A7       $A742   FOR
A010  1D AD           $AD       $AD1E   NEXT
A012  F7 AB           $AB       $AEF8   DATA
A014  A4 AB           $AB       $ABA5   INPUT#
A016  BE AB           $AB       $ABBF   INPUT
A018  80 B0           $B0       $B081   DIM
A01A  05 AC           $AC       $AC06   READ
A01C  A4 A9           $A9       $A9A5   LET
A01E  9F A8           $A8       $A8AØ   GOTO
A020  70 A8           $A8       $A871   RUN
A022  27 A9           $A9       $A928   IF
A024  1C A8           $A8       $A81D   RESTORE
A026  82 A8           $A8       $A883   GOSUB
A028  D1 A8           $A8       $A8D2   RETURN
A02A  3A A9           $A9       $A93B   REM
A02C  2E A8           $A8       $A8F2   STOP
A02E  4A A9           $A9       $A94B   ON
A030  2C B8           $B8       $B82D   WAIT
A032  67 E1           $E1       $E168   LOAD
A034  55 E1           $E1       $E156   SAVE
A036  64 E1           $E1       $E165   VERIFY
A038  B2 B3           $B3       $B3B3   DEF
A03A  23 B8           $B8       $B824   POKE
A03C  7F AA           $AA       $AA8Ø   PRINT#
A03E  9F AA           $AA       $AAAØ   PRINT
A040  56 A8           $A8       $A857   CONT
A042  9B A6           $A6       $A69C   LIST
A044  5D A6           $A6       $A65E   CLR
A046  85 AA           $AA       $AA86   CMD
A048  29 E1           $E1       $E12A   SYS
A04A  BD E1           $E1       $E1BE   OPEN
A04C  C6 E1           $E1       $E1C7   CLOSE
A04E  7A AB           $AB       $AB7B   GET
A050  41 A6           $A6       $A642   NEW

*****************************  A BASIC-függvények címei
A052  39 BC           $B4       $BC39   SGN
A054  CC BC           $B5       $BCCC   INT
A056  58 BC           $B6       $BC58   ABS
A058  10 03           $B7       $Ø31Ø   USR
A05A  7D B3           $B8       $B37D   FRE
A05C  9E B3           $B9       $B39E   POS
A05E  71 BF           $BA       $BF71   SQR
A060  97 E0           $BB       $E097   RND
A062  EA B9           $BC       $B9EA   LOG
A064  ED BF           $BD       $BFED   EXP
```

```
A066  64 E2          $BE     $E264   COS
A068  6B E2          $BF     $E26B   SIN
A06A  B4 E2          $C0     $E2B4   TAN
A06C  0E E3          $C1     $E30E   ATN
A06E  0D B8          $C2     $B800   PEEK
A070  7C B7          $C3     $B77C   LEN
A072  65 B4          $C4     $B465   STR$
A074  AD B7          $C5     $B7AD   VAL
A076  8B B7          $C6     $B78B   ASC
A078  EC B6          $C7     $B6EC   CHR$
A07A  00 B7          $C8     $B700   LEFT$
A07C  2C B7          $C9     $B72C   RIGHT$
A07E  37 B7          $CA     $B737   MID$
```

********** A prioritáskódok és műveletek -1-es címei

```
A080  79 69 B8       $79, $B86A      összeadás
A083  79 52 B8       $79, $B853      kivonás
A086  7B 2A BA       $7B, $BA2B      szorzás
A089  7B 11 BB       $7B, $BB12      osztás
A08C  7F 7A BF       $7F, $BF7B      hatványozás
A08F  50 E8 AF       $50, $AFE9      AND
A092  46 E5 AF       $46, $AFE6      OR
A095  7D B3 BF       $7D, $BFB4      előjelváltás
A098  5A D3 AE       $5A, $AED4      NOT
A09B  64 15 B0       $64, $B016      összehasonlítás
```

********** A BASIC-utasításszavak

```
A09E  45 4E
A0A0  C4 46 4F D2 4E 45 58 D4    end          next
A0A8  44 41 54 C1 49 4E 50 55    for          input#
A0B0  54 A3 49 4E 50 55 D4 44    data         dim
A0B8  49 CD 52 45 41 C4 4C 45    input        let
A0C0  D4 47 4F 54 CF 52 55 CE    read         run
A0C8  49 C6 52 45 53 54 4F 52    goto         restore
A0D0  C5 47 4F 53 55 C2 52 45    if           return
A0D8  54 55 52 CE 52 45 CD 53    gosub        stop
A0E0  54 4F D0 4F CE 57 41 49    rem          wait
A0E8  D4 4C 4F 41 C4 53 41 56    on           save
A0F0  C5 56 45 52 49 46 D9 44    load         def
A0F8  45 C6 50 4F 4B C5 50 52    verify       print#
A100  49 4E 54 A3 50 52 49 4E    poke
A108  D4 43 4F 4E D4 4C 49 53    print        list
A110  D4 43 4C D2 43 4D C4 53    cont         cmd          sys
A118  59 D3 4F 50 45 CE 43 4C    clr          close
A120  4F 53 C5 47 45 D4 4E 45    open         new
A128  D7 54 41 42 A8 54 CF 46    get          to
A130  CE 53 50 43 A8 54 48 45    tab(         then
A138  CE 4E 4F D4 53 54 45 D0    spc(         stop
A140  AB AD AA AF DE 41 4E C4    not          and
A148  4F D2 BE BD BC 53 47 CE    + - * / ↑    sgn
A150  49 4E D4 41 42 D3 55 53    or < =       abs          usr
A158  D2 46 52 C5 50 4F D3 53    int          pos          sqr
A160  51 D2 52 4E C4 4C 4F C7    fre          rnd          log
A168  45 58 D0 43 4F D3 53 49    exp          cos          sin
A170  CE 54 41 CE 41 54 CE 50    tan          atn          peek
A178  45 45 CB 4C 45 CE 53 54    len          str$
```



```
A180  52 A4 56 41 CC 41 53 C3    val          asc
A188  43 48 52 A4 4C 45 46 54    shr$         left$
A190  A4 52 49 47 48 54 A4 4D    right$       mid$
A198  49 44 A4 47 CF 00          go
```

****

```
A19E  54 4F
A1A0  4F 20 4D 41 4E 59 20 46
A1A8  49 4C 45 D3 46 49 4C 45
A1B0  20 4F 50 45 CE 46 49 4C
A1B8  45 20 4E 4F 54 20 4F 50
A1C0  45 CE 46 49 4C 45 20 4E
A1C8  4F 54 20 46 4F 55 4E C4
A1D0  44 45 56 49 43 45 20 4E
A1D8  4F 54 20 50 52 45 53 45
A1E0  4E D4 4E 4F 54 20 49 4E
A1E8  50 55 54 20 46 49 4C C5
A1F0  4E 4F 54 20 4F 55 54 50
A1F8  55 54 20 46 49 4C C5 4D
A200  49 53 53 49 4E 47 20 46
A208  49 4C 45 20 4E 41 4D C5
A210  49 4C 4C 45 47 41 4C 20
A218  44 45 56 49 43 45 20 4E
A220  55 4D 42 45 D2 4E 45 58
A228  54 20 57 49 54 48 4F 55
A230  54 20 46 4F D2 53 59 4E
A238  54 41 D8 52 45 54 55 52
A240  4E 20 57 49 54 48 4F 55
A248  54 20 47 4F 53 55 C2 4F
A250  55 54 20 4F 46 20 44 41
A258  54 C1 49 4C 4C 45 47 41
A260  4C 20 51 55 41 4E 54 49
A268  54 D9 4F 56 45 52 46 4C
A270  4F D7 4F 55 54 20 4F 46
A278  20 4D 45 4D 4F 52 D9 55
A280  4E 44 45 46 27 44 20 53
A288  54 41 54 45 4D 45 4E D4
A290  42 41 44 20 53 55 42 53
A298  43 52 49 50 D4 52 45 44
A2A0  49 4D 27 44 20 41 52 52
A2A8  41 D9 44 49 56 49 53 49
A2B0  4F 4E 20 42 59 20 5A 45
A2B8  52 CF 49 4C 4C 45 47 41
A2C0  4C 20 44 49 52 45 43 D4
A2C8  54 59 50 45 20 4D 49 53
A2D0  4D 41 54 43 C8 53 54 52
A2D8  49 4E 47 20 54 4F 4F 20
A2E0  4C 4F 4E C7 46 49 4C 45
A2E8  20 44 41 54 C1 46 4F 52
A2F0  4D 55 4C 41 20 54 4F 4F
A2F8  20 43 4F 4D 50 4C 45 D8
A300  43 41 4E 27 54 20 43 4F
A308  4E 54 49 4E 55 C5 55 4E
A310  44 45 46 27 44 20 46 55
A318  4E 43 54 49 4F CE 56 45
A320  52 49 46 D9 4C 4F 41 C4
```

A BASIC-hibaüzenetek

```
1     too many files
2     file open
3     file not open
4     file not found
5     device not present
6     not input file
7     not output file
8     missing filename
9     illegal device number
10    next without for
11    syntax
12    return without gosub
13    out of data
14    illegal quantity
15    overflow
16    out of memory
17    undef'd statement
18    bad subscript
19    redim'd array
20    division by zero
21    illegal direct
22    type mismatch
23    string too long
24    file data
25    formula too complex
26    can't continue
27    undef'd function
28    verify
29    load
```

```
*****************************   A hibaüzenetek címei
A328  9E A1 AC A1 B5 A1 C2 A1
A330  D0 A1 E2 A1 F0 A1 FF A1
A338  10 A2 25 A2 35 A2 3B A2
A340  4F A2 5A A2 6A A2 72 A2
A348  7F A2 90 A2 9D A2 AA A2
A350  BA A2 C8 A2 D5 A2 E4 A2
A358  ED A2 00 A3 0E A3 1E A3
A360  24 A3 83 A3

*****************************   Az interpreter üzenetei
A364  0D 4F 4B 0D                ok
A368  00 20 20 45 52 52 4F 52    error
A370  00 20 49 4E 20 00 0D 0A    in
A378  52 45 41 44 59 2E 0D 0A    ready.
A380  00 0D 0A 42 52 45 41 4B    break
A388  00 A0

*****************************   Keresés a veremben.Rutin a FOR-NEXT,GOSUB
A38A  BA          TSX                          utasításokhoz
A38B  E8          INX
A38C  E8          INX
A38D  E8          INX
A38E  E8          INX
A38F  BD 01 01    LDA   0101,X
A392  C9 81       CMP   #81          a FOR kódja
A394  D0 21       BNE   A3B7
A396  A5 4A       LDA   4A           a változónevek összehasonlítása
A398  D0 0A       BNE   A3A4
A39A  BD 02 01    LDA   0102,X
A39D  85 49       STA   49
A39F  BD 03 01    LDA   0103,X
A3A2  85 4A       STA   4A
A3A4  DD 03 01    CMP   0103,X
A3A7  D0 07       BNE   A3B0
A3A9  A5 49       LDA   49
A3AB  DD 02 01    CMP   0102,X
A3AE  F0 07       BEQ   A3B7
A3B0  8A          TXA
A3B1  18          CLC
A3B2  69 12       ADC   #12          a veremmutató nő 18-cal
A3B4  AA          TAX
A3B5  D0 D8       BNE   A38F         a következő ciklus ellenőrzése
A3B7  60          RTS

*****************************   A blokk-léptető rutin szabad tárterületet keres
A3B8  20 08 A4    JSR   A408
A3BB  85 31       STA   31
A3BD  84 32       STY   32
A3BF  38          SEC                belépési pont
A3C0  A5 5A       LDA   5A           $5F/$60 a korábbi blokk-kezdet
A3C2  E5 5F       SBC   5F           $5A/$5B a korábbi blokkvég + 1
A3C4  85 22       STA   22           $58/$59 új blokkvég + 1
A3C6  A8          TAY
A3C7  A5 5B       LDA   5B
```



```
A3C9  E5 60       SBC   60
A3CB  AA          TAX
A3CC  E8          INX
A3CD  98          TYA
A3CE  F0 23       BEQ   A3F3
A3D0  A5 5A       LDA   5A
A3D2  38          SEC
A3D3  E5 22       SBC   22
A3D5  85 5A       STA   5A
A3D7  B0 03       BCS   A3DC
A3D9  C6 5B       DEC   5B
A3DB  38          SEC
A3DC  A5 58       LDA   58
A3DE  E5 22       SBC   22
A3E0  85 58       STA   58
A3E2  B0 08       BCS   A3EC
A3E4  C6 59       DEC   59
A3E6  90 04       BCC   A3EC
A3E8  B1 5A       LDA   (5A),Y
A3EA  91 58       STA   (58),Y
A3EC  88          DEY
A3ED  D0 F9       BNE   A3E8
A3EF  B1 5A       LDA   (5A),Y
A3F1  91 58       STA   (58),Y
A3F3  C6 5B       DEC   5B
A3F5  C6 59       DEC   59
A3F7  CA          DEX
A3F8  D0 F2       BNE   A3EC
A3FA  60          RTS

*****************************   Szabad hely keresése a veremben
A3FB  0A          ASL   A          Az akkumulátorban tároljuk a szükséges hely
A3FC  69 3E       ADC   #3E        értékének felét
A3FE  B0 35       BCS   A435               "out of memory"
A400  85 22       STA   22
A402  BA          TSX
A403  E4 22       CPX   22
A405  90 2E       BCC   A435               "out of memory"
A407  60          RTS

*****************************   A rutin helyet biztosít a beszúrt programsorok
A408  C4 34       CPY   34         és változók számára
A40A  90 28       BCC   A434
A40C  D0 04       BNE   A412               az A/Y=cím, a szükséges helyre utal
A40E  C5 33       CMP   33
A410  90 22       BCC   A434
A412  48          PHA
A413  A2 09       LDX   #09
A415  98          TYA
A416  48          PHA
A417  B5 57       LDA   57,X       Az artimetikai műveletek eredményének mentése
A419  CA          DEX
A41A  10 FA       BPL   A416
A41C  20 26 B5    JSR   B526               Garbage Collection /szemétgyűjtés/
A41F  A2 F7       LDX   #F7
A421  68          PLA
```

```
A422  95 61      STA   61,X      A regiszter tartalmának visszatöltése
A424  E8         INX
A425  30 FA      BMI   A421
A427  68         PLA
A428  A8         TAY
A429  68         PLA
A42A  C4 34      CPY   34
A42C  90 06      BCC   A434      ha ok, akkor kész
A42E  D0 05      BNE   A435      nincs hely: "out of memory"
A430  C5 33      CMP   33
A432  B0 01      BCS   A435
A434  60         RTS
A435  A2 10      LDX   #10       "out of memory" hibaszáma

******************************
A437  6C 00 03   JMP   (0300)    Ugrás a BASIC-melegindításhoz /JMP $E38B/

******************************  A hibaüzenet kiírása. A hibaszám az X-re-
A43A  8A         TXA             giszterben, 1-től 30-ig
A43B  0A         ASL   A
A43C  AA         TAX
A43D  BD 26 A3   LDA   A326,X
A440  85 22      STA   22
A442  BD 27 A3   LDA   A327,X    a hibaüzenet címének betöltése
A445  85 23      STA   23
A447  20 CC FF   JSR   FFCC      a CLRCH aktív I/O csatornák visszaállítása
A44A  A9 00      LDA   #00
A44C  85 13      STA   13        az I/O-kapcsoló visszaállítása
A44E  20 D7 AA   JSR   AAD7      (CR) és (LF) kiírása
A451  20 45 AB   JSR   AB45      "?" kiírása
A454  A0 00      LDY   #00
A456  B1 22      LDA   (22),Y    a hibaüzenet szövege
A458  48         PHA
A459  29 7F      AND   #7F
A45B  20 47 AB   JSR   AB47      a hibaüzenet kiadása
A45E  C8         INY
A45F  68         PLA
A460  10 F4      BPL   A456
A462  20 7A A6   JSR   A67A      a BASIC-mutató inicializ., a CONT letiltása
A465  A9 69      LDA   #69
A467  A0 A3      LDY   #A3       az A/Y-mutató átállítása "error"-ra
A469  20 1E AB   JSR   AB1E      a szöveg kiírása
A46C  A4 3A      LDY   3A        program mód?
A46E  C8         INY
A46F  F0 03      BEQ   A474      nem
A471  20 C2 BD   JSR   BDC2      a sorszám kiírása
A474  A9 76      LDA   #76
A476  A0 A3      LDY   #A3       a mutató átállítása "ready"-re
A478  20 1E AB   JSR   AB1E      a szöveg kiírása
A47B  A9 80      LDA   #80
A47D  20 90 FF   JSR   FF90      a parancsüzemmód-kapcsoló beállítása

******************************  Az Input-utasítás várakozási ciklusa
A480  6C 02 03   JMP   (0302)    JMP $A4B3
A483  20 60 A5   JSR   A560      a BASIC-sor beolvasása az input-pufferba
A486  86 7A      STX   7A
```



```
A488  84 7B      STY   7B        CHRGET - az input puffer-mutató
A48A  20 73 00   JSR   0073      CHRGET a következő karakter beolvasása
A48D  AA         TAX
A48E  F0 F0      BEQ   A480      ha a puffer üres, akkor várakozni!
A490  A2 FF      LDX   #FF
A492  86 3A      STX   3A        a parancsmód indexe
A494  90 06      BCC   A49C      ha számjegy, BASIC-sorként beszúrni
A496  20 79 A5   JSR   A579      a BASIC-sor kódolása
A499  4C E1 A7   JMP   A7E1      az utasítás végrehajtása

******************************  Programsorok törlése és beszúrása
A49C  20 6B A9   JSR   A96B      a sorszám átalakítása címformátummá $14/$15
A49F  20 79 A5   JSR   A579      a BASIC-sor kódolása
A4A2  84 0B      STY   0B        a mutató az input pufferban
A4A4  20 13 A6   JSR   A613      a BASIC-sor címének kiszámítása
A4A7  90 44      BCC   A4ED      ha még nincs ilyen, kiugrás a törlésből

******************************  A programsor törlése
A4A9  A0 01      LDY   #01
A4AB  B1 5F      LDA   (5F),Y
A4AD  85 23      STA   23
A4AF  A5 2D      LDA   2D
A4B1  85 22      STA   22
A4B3  A5 60      LDA   60
A4B5  85 25      STA   25
A4B7  A5 5F      LDA   5F
A4B9  88         DEY
A4BA  F1 5F      SBC   (5F),Y
A4BC  18         CLC
A4BD  65 2D      ADC   2D
A4BF  85 2D      STA   2D
A4C1  85 24      STA   24
A4C3  A5 2E      LDA   2E
A4C5  69 FF      ADC   #FF
A4C7  85 2E      STA   2E
A4C9  E5 60      SBC   60
A4CB  AA         TAX
A4CC  38         SEC
A4CD  A5 5F      LDA   5F
A4CF  E5 2D      SBC   2D
A4D1  A8         TAY
A4D2  B0 03      BCS   A4D7
A4D4  E8         INX
A4D5  C6 25      DEC   25
A4D7  18         CLC
A4D8  65 22      ADC   22
A4DA  90 03      BCC   A4DF
A4DC  C6 23      DEC   23
A4DE  18         CLC
A4DF  B1 22      LDA   (22),Y    a léptető ciklus
A4E1  91 24      STA   (24),Y
A4E3  C8         INY
A4E4  D0 F9      BNE   A4DF
A4E6  E6 23      INC   23
A4E8  E6 25      INC   25
A4EA  CA         DEX
```

```
A4EB  D0 F2       BNE   A4DF

*****************************
A4ED  20 59 A6    JSR   A659
A4F0  20 33 A5    JSR   A533
A4F3  AD 00 02    LDA   0200
A4F6  F0 88       BEQ   A480
A4F8  18          CLC
A4F9  A5 2D       LDA   2D
A4FB  85 5A       STA   5A
A4FD  65 0B       ADC   0B
A4FF  85 58       STA   58
A501  A4 2E       LDY   2E
A503  84 5B       STY   5B
A505  90 01       BCC   A508
A507  C8          INY
A508  84 59       STY   59
A50A  20 B8 A3    JSR   A3B8
A50D  A5 14       LDA   14
A50F  A4 15       LDY   15
A511  8D FE 01    STA   01FE
A514  8C FF 01    STY   01FF
A517  A5 31       LDA   31
A519  A4 32       LDY   32
A51B  85 2D       STA   2D
A51D  84 2E       STY   2E
A51F  A4 0B       LDY   0B
A521  88          DEY
A522  B9 FC 01    LDA   01FC,Y
A525  91 5F       STA   (5F),Y
A527  88          DEY
A528  10 F8       BPL   A522
A52A  20 59 A6    JSR   A659
A52D  20 33 A5    JSR   A533
A530  4C 80 A4    JMP   A480

*****************************
A533  A5 2B       LDA   2B
A535  A4 2C       LDY   2C
A537  85 22       STA   22
A539  84 23       STY   23
A53B  18          CLC
A53C  A0 01       LDY   #01
A53E  B1 22       LDA   (22),Y
A540  F0 1D       BEQ   A55F
A542  A0 04       LDY   #04
A544  C8          INY
A545  B1 22       LDA   (22),Y
A547  D0 FB       BNE   A544
A549  C8          INY
A54A  98          TYA
A54B  65 22       ADC   22
A54D  AA          TAX
A54E  A0 00       LDY   #00
A550  91 22       STA   (22),Y
A552  A5 23       LDA   23
```

Egy programsor beszúrása
a CLR-utasítás
a programsorok újraösszekapcsolása
karakter a pufferban?
ha nem, ugrás az input várakozási ciklusába

a BASIC-sorok eltolása

egy karakter bemásolása a pufferből a programba

a CLR-utasítás
a programsorok újraösszekapcsolása
ugrás az input várakozási ciklusához

A BASIC-programsorok újraláncolása


```
A554  69 00       ADC   #00
A556  C8          INY
A557  91 22       STA   (22),Y
A559  86 22       STX   22
A55B  85 23       STA   23
A55D  90 DD       BCC   A53C
A55F  60          RTS

*****************************
A560  A2 00       LDX   #00
A562  20 12 E1    JSR   E112
A565  C9 0D       CMP   #0D
A567  F0 0D       BEQ   A576
A569  9D 00 02    STA   0200,X
A56C  E8          INX
A56D  E0 59       CPX   #59
A56F  90 F1       BCC   A562
A571  A2 17       LDX   #17
A573  4C 37 A4    JMP   A437
A576  4C CA AA    JMP   AACA

*****************************
A579  6C 04 03    JMP   (0304)
A57C  A6 7A       LDX   7A
A57E  A0 04       LDY   #04
A580  84 0F       STY   0F
A582  BD 00 02    LDA   0200,X
A585  10 07       BPL   A58E
A587  C9 FF       CMP   #FF
A589  F0 3E       BEQ   A5C9
A58B  E8          INX
A58C  D0 F4       BNE   A582
A58E  C9 20       CMP   #20
A590  F0 37       BEQ   A5C9
A592  85 08       STA   08
A594  C9 22       CMP   #22
A596  F0 56       BEQ   A5EE
A598  24 0F       BIT   0F
A59A  70 2D       BVS   A5C9
A59C  C9 3F       CMP   #3F
A59E  D0 04       BNE   A5A4
A5A0  A9 99       LDA   #99
A5A2  D0 25       BNE   A5C9
A5A4  C9 30       CMP   #30
A5A6  90 04       BCC   A5AC
A5A8  C9 3C       CMP   #3C
A5AA  90 1D       BCC   A5C9
A5AC  84 71       STY   71
A5AE  A0 00       LDY   #00
A5B0  84 0B       STY   0B
A5B2  88          DEY
A5B3  86 7A       STX   7A
A5B5  CA          DEX
A5B6  C8          INY
A5B7  E8          INX
A5B8  BD 00 02    LDA   0200,X
```

Egy sor beírása

egy karakter beolvasása
RETURN-billentyű?
ha igen, a beolvasásnak vége
a karakter tárolása a pufferban

a 89. karakter?
ha nem, a beolvasás folytatása
a "string too long" sorszáma
a hibaüzenet kiadása
puffer lezárása $Ø-val, CR kiírása

Egy sor átalakítása interpreter kódokká
JMP $A57C

az egyszeres idézőjel kapcsolója
egy karakter betöltése a pufferből
nem BASIC-kód?
π-kód?

" " üres karakter

"" egyszeres idézőjel

"?" kérdőjel

a PRINT kóddal helyettesíteni

"Ø"
kisebb?

a pufferban lévő karaktert

```
A5BB  38          SEC
A5BC  F9 9E A0    SBC   A09E,Y
A5BF  F0 F5       BEQ   A5B6
A5C1  C9 80       CMP   #80
A5C3  D0 30       BNE   A5F5
A5C5  05 0B       ORA   0B
A5C7  A4 71       LDY   71
A5C9  E8          INX
A5CA  C8          INY
A5CB  99 FB 01    STA   01FB,Y
A5CE  B9 FB 01    LDA   01FB,Y
A5D1  F0 36       BEQ   A609
A5D3  38          SEC
A5D4  E9 3A       SBC   #3A
A5D6  F0 04       BEQ   A5DC
A5D8  C9 49       CMP   #49
A5DA  D0 02       BNE   A5DE
A5DC  85 0F       STA   0F
A5DE  38          SEC
A5DF  E9 55       SBC   #55
A5E1  D0 9F       BNE   A582
A5E3  85 08       STA   08
A5E5  BD 00 02    LDA   0200,X
A5E8  F0 DF       BEQ   A5C9
A5EA  C5 08       CMP   08
A5EC  F0 DB       BEQ   A5C9
A5EE  C8          INY
A5EF  99 FB 01    STA   01FB,Y
A5F2  E8          INX
A5F3  D0 F0       BNE   A5E5
A5F5  A6 7A       LDX   7A
A5F7  E6 0B       INC   0B
A5F9  C8          INY
A5FA  B9 9D A0    LDA   A09D,Y
A5FD  10 FA       BPL   A5F9
A5FF  B9 9E A0    LDA   A09E,Y
A602  D0 B4       BNE   A5B8
A604  BD 00 02    LDA   0200,X
A607  10 BE       BPL   A5C7
A609  99 FD 01    STA   01FD,Y
A60C  C6 7B       DEC   7B
A60E  A9 FF       LDA   #FF
A610  85 7A       STA   7A
A612  60          RTS

*****************************
A613  A5 2B       LDA   2B
A615  A6 2C       LDX   2C
A617  A0 01       LDY   #01
A619  85 5F       STA   5F
A61B  86 60       STX   60
A61D  B1 5F       LDA   (5F),Y
A61F  F0 1F       BEQ   A640
A621  C8          INY
A622  C8          INY
A623  A5 15       LDA   15
```

összehasonlítani a táblázatbeli utasításszavakkal

megvan a BASIC-kód =+$80 a számláló állása

a BASIC-kód tárolása
és az állapotregiszter beállítása
ha vége, akkor kész

":"
elválasztójel?
DATA-kód?

REM-kód?

összehasonlítás a táblázattal

input-puffer mutató -1

Egy programsor kezdőcímének kiszámítása

a programkezdet-mutató

ha a láncolási cím felső byte-ja=0,akkor vége, nem sikerült megtalálni

a keresett sorszám felső byte-jának


```
A625  D1 5F       CMP   (5F),Y
A627  90 18       BCC   A641
A629  F0 03       BEQ   A62E
A62B  88          DEY
A62C  D0 09       BNE   A637
A62E  A5 14       LDA   14
A630  88          DEY
A631  D1 5F       CMP   (5F),Y
A633  90 0C       BCC   A641
A635  F0 0A       BEQ   A641
A637  88          DEY
A638  B1 5F       LDA   (5F),Y
A63A  AA          TAX
A63B  88          DEY
A63C  B1 5F       LDA   (5F),Y
A63E  B0 D7       BCS   A617
A640  18          CLC
A641  60          RTS

*****************************
A642  D0 FD       BNE   A641
A644  A9 00       LDA   #00
A646  A8          TAY
A647  91 2B       STA   (2B),Y
A649  C8          INY
A64A  91 2B       STA   (2B),Y
A64C  A5 2B       LDA   2B
A64E  18          CLC
A64F  69 02       ADC   #02
A651  85 2D       STA   2D
A653  A5 2C       LDA   2C
A655  69 00       ADC   #00
A657  85 2E       STA   2E
A659  20 8E A6    JSR   A68E
A65C  A9 00       LDA   #00

*****************************
A65E  D0 2D       BNE   A68D
A660  20 E7 FF    JSR   FFE7
A663  A5 37       LDA   37
A665  A4 38       LDY   38
A667  85 33       STA   33
A669  84 34       STY   34
A66B  A5 2D       LDA   2D
A66D  A4 2E       LDY   2E
A66F  85 2F       STA   2F
A671  84 30       STY   30
A673  85 31       STA   31
A675  84 32       STY   32
A677  20 1D A8    JSR   A81D
A67A  A2 19       LDX   #19
A67C  86 16       STX   16
A67E  68          PLA
A67F  A8          TAY
A680  68          PLA
A681  A2 FA       LDX   #FA
```

összehasonlítása a pillanatnyi sorszámmal
ha kisebb, nem sikerült megtalálni
ha egyenlő, az alsó byte ellenőrzése

feltétlen ugrás

az alsó byte összehasonlítása
ha kisebb,
vagy ezzel egyenlő C=1

a következő programsor címe

továbbkeresés

A NEW BASIC-utasítás

kétszer $00 a programstartnál

változókezdet = programkezdet +2

a programkezdet CLRGET mutatója

A CLR BASIC-utasítás

CLALL az összes I/O csatorna visszaállítása

füzérkezdet a BASIC-RAM végére

a változóterület vége=változókezdet

a RESTORE-utasítás

a füzérleíró indexének visszaállítása

```
A683  9A        TXS
A684  48        PHA
A685  98        TYA
A686  48        PHA
A687  A9 00     LDA  #00
A689  85 3E     STA  3E
A68B  85 10     STA  10
A68D  60        RTS

*****************************
A68E  18        CLC
A68F  A5 2B     LDA  2B
A691  69 FF     ADC  #FF
A693  85 7A     STA  7A
A695  A5 2C     LDA  2C
A697  69 FF     ADC  #FF
A699  85 7B     STA  7B
A69B  60        RTS

*****************************
A69C  90 06     BCC  A6A4
A69E  F0 04     BEQ  A6A4
A6A0  C9 AB     CMP  #AB
A6A2  D0 E9     BNE  A68D
A6A4  20 6B A9  JSR  A96B
A6A7  20 13 A6  JSR  A613
A6AA  20 79 00  JSR  0079
A6AD  F0 0C     BEQ  A6BB
A6AF  C9 AB     CMP  #AB
A6B1  D0 8E     BNE  A641
A6B3  20 73 00  JSR  0073
A6B6  20 6B A9  JSR  A96B
A6B9  D0 86     BNE  A641
A6BB  68        PLA
A6BC  68        PLA
A6BD  A5 14     LDA  14
A6BF  05 15     ORA  15
A6C1  D0 06     BNE  A6C9
A6C3  A9 FF     LDA  #FF
A6C5  85 14     STA  14
A6C7  85 15     STA  15
A6C9  A0 01     LDY  #01
A6CB  84 0F     STY  0F
A6CD  B1 5F     LDA  (5F),Y
A6CF  F0 43     BEQ  A714
A6D1  20 2C AB  JSR  AB2C
A6D4  20 D7 AA  JSR  AAD7
A6D7  C8        INY
A6D8  B1 5F     LDA  (5F),Y
A6DA  AA        TAX
A6DB  C8        INY
A6DC  B1 5F     LDA  (5F),Y
A6DE  C5 15     CMP  15
A6E0  D0 04     BNE  A6E6
A6E2  E4 14     CPX  14
A6E4  F0 02     BEQ  A6E8
```

a veremmutató inicializálása

a CONT letiltása

a BASIC-kezdet program-mutató

a BASIC-kezdet

minusz 1

A LIST BASIC-utasítás
számjegy? /sor-sorszám/
csak LIST?
"-"kód
ha más kód, akkor SYNTAX ERROR
a sorszám beolvasása
a programsor kezdőcímének kiszámítása
CHRGOT az utolsó karakter beolvasása
nincs sorszám
"-" kód
nem, SYNTAX ERROR
CHRGET a következő karakter beolvasása
a sorszám beolvasása

a második sorszám nullával egyenlő?
nem

végig listázni!

a láncolási cím felső byte
ha igen, akkor kész
a stop-billentyű ellenőrzése
"CR" kiírása, új sor

a sorszám beolvasása

összehasonlítás az utolsó sorszámmal


```
A6E6  B0 2C     BCS  A714
A6E8  84 49     STY  49
A6EA  20 CD BD  JSR  BDCD
A6ED  A9 20     LDA  #20
A6EF  A4 49     LDY  49
A6F1  29 7F     AND  #7F
A6F3  20 47 AB  JSR  AB47
A6F6  C9 22     CMP  #22
A6F8  D0 06     BNE  A700
A6FA  A5 0F     LDA  0F
A6FC  49 FF     EOR  #FF
A6FE  85 0F     STA  0F
A700  C8        INY
A701  F0 11     BEQ  A714
A703  B1 5F     LDA  (5F),Y
A705  D0 10     BNE  A717
A707  A8        TAY
A708  B1 5F     LDA  (5F),Y
A70A  AA        TAX
A70B  C8        INY
A70C  B1 5F     LDA  (5F),Y
A70E  86 5F     STX  5F
A710  85 60     STA  60
A712  D0 B5     BNE  A6C9
A714  4C 86 E3  JMP  E386

*****************************
A717  6C 06 03  JMP  (0306)
A71A  10 D7     BPL  A6F3
A71C  C9 FF     CMP  #FF
A71E  F0 D3     BEQ  A6F3
A720  24 0F     BIT  0F
A722  30 CF     BMI  A6F3
A724  38        SEC
A725  E9 7F     SBC  #7F
A727  AA        TAX
A728  84 49     STY  49
A72A  A0 FF     LDY  #FF
A72C  CA        DEX
A72D  F0 08     BEQ  A737
A72F  C8        INY
A730  B9 9E A0  LDA  A09E,Y
A733  10 FA     BPL  A72F
A735  30 F5     BMI  A72C
A737  C8        INY
A738  B9 9E A0  LDA  A09E,Y
A73B  30 B2     BMI  A6EF
A73D  20 47 AB  JSR  AB47
A740  D0 F5     BNE  A737

*****************************
A742  A9 80     LDA  #80
A744  85 10     STA  10
A746  20 A5 A9  JSR  A9A5
A749  20 8A A3  JSR  A38A
A74C  D0 05     BNE  A753
```

ha nagyobb, akkor kész

a sorszám kiírása
" " üresjel

a karakter kiírása
"(") egyszeres idézőjel?

"egyszeres idézőjel" kapcsoló beállítása

255 karakter után sincs jel?
ha nincs, abbahagyni
a karakter beolvasása
ha nincs sorvégjel, akkor kiíratás

a következő sor kezdőcímének

tárolása és
folytatás
a BASIC-melegindításig

A BASIC-kód konvertálása szöveggé
JMP $A71A
ha nincs interpreter-kód, kiírás
π kód
a fenti kód esetén kiírás
"egyszeres idézőjel" - mód?
ha karakter, kiírás

az offset levonása
X szerinti kód
a karakter-mutató tárolása

első utasítássző?

az X-ik utasítássző rel. címének keresése

a 7. bit beállítva, áttérés következ. szóra

utasítássző olv. a táblázatból
ha ez volt az utolsó betű, akkor kész
karakter kiírása
következő betű kiírása

FOR BASIC-utasítás
egész tipusú /integer/ változó

a LET-utasítás beállitja a ciklusváltozót
változó ellenőrzés /FOR-NEXT ciklus/
nem sikerült megtalálni

```
A74E  8A          TXA
A74F  69 0F       ADC  #0F        a veremmutató növelése
A751  AA          TAX
A752  9A          TXS
A753  68          PLA             visszaugrási cím beolvasása a veremből
A754  68          PLA
A755  A9 09       LDA  #09
A757  20 FB A3    JSR  A3FB       van-e elegendő hely a veremben?
A75A  20 06 A9    JSR  A906       a következő BASIC-utasítás megkeresése
A75D  18          CLC
A75E  98          TYA             programmutató beállítása a köv. utasításra
A75F  65 7A       ADC  7A
A761  48          PHA             és tárolása a veremben
A762  A5 7B       LDA  7B
A764  69 00       ADC  #00
A766  48          PHA
A767  A5 3A       LDA  3A
A769  48          PHA             a verem futó változója
A76A  A5 39       LDA  39
A76C  48          PHA
A76D  A9 A4       LDA  #A4        "TO"  kód
A76F  20 FF AE    JSR  AEFF       a kód ellenőrzése
A772  20 8D AD    JSR  AD8D       numerikus ellenőrzés
A775  20 8A AD    JSR  AD8A       numerikus kifejezés betöltése a FAC után
A778  A5 66       LDA  66
A77A  09 7F       ORA  #7F
A77C  25 62       AND  62
A77E  85 62       STA  62
A780  A9 8B       LDA  #8B
A782  A0 A7       LDY  #A7        a visszaugrási cím tárolása
A784  85 22       STA  22
A786  84 23       STY  23
A788  4C 43 AE    JMP  AE43       a ciklus végértékének tárolása a veremben
A78B  A9 BC       LDA  #BC
A78D  A0 B9       LDY  #B9        mutató a konstans 1-re
A78F  20 A2 BB    JSR  BBA2       a ciklusvált.kezdőérték alapértelmezés /FAC-ba/
A792  20 79 00    JSR  0079       CHRGOT az utolsó karakter betöltése
A795  C9 A9       CMP  #A9        a "STEP" kódja
A797  D0 06       BNE  A79F       nem a STEP kódja
A799  20 73 00    JSR  0073       CHRGET a következő karakter betöltése
A79C  20 8A AD    JSR  AD8A       a FAC szerinti numerikus kifejezés betöltése
A79F  20 2B BC    JSR  BC2B       az előjel betöltése és
A7A2  20 38 AE    JSR  AE38       tárolása a lépésközzel együtt a veremben
A7A5  A5 4A       LDA  4A
A7A7  48          PHA             a változó neve
A7A8  A5 49       LDA  49
A7AA  48          PHA
A7AB  A9 81       LDA  #81        és a FOR kód a verembe
A7AD  48          PHA

*****************************     Interpreter ciklus
A7AE  20 2C AB    JSR  AB2C       a stop-billentyű ellenőrzése
A7B1  A5 7A       LDA  7A
A7B3  A4 7B       LDY  7B         programmutató
A7B5  C0 02       CPY  #02        parancsmód?
A7B7  EA          NOP
```



```
A7B8  F0 04       BEQ  A7BE       igen
A7BA  85 3D       STA  3D         CONT-mutatóként tárolni!
A7BC  84 3E       STY  3E
A7BE  A0 00       LDY  #00
A7C0  B1 7A       LDA  (7A),Y     a következő karakter
A7C2  D0 43       BNE  A807       nem sorvég?
A7C4  A0 02       LDY  #02
A7C6  B1 7A       LDA  (7A),Y     a program vége
A7C8  18          CLC             az END-kapcsoló beállítása
A7C9  D0 03       BNE  A7CE
A7CB  4C 4B A8    JMP  A84B       ha igen, az END végrehajtása
A7CE  C8          INY
A7CF  B1 7A       LDA  (7A),Y
A7D1  85 39       STA  39
A7D3  C8          INY             az aktuális sorszám tárolása
A7D4  B1 7A       LDA  (7A),Y
A7D6  85 3A       STA  3A
A7D8  98          TYA
A7D9  65 7A       ADC  7A         a programmutató beállítása a programsorra
A7DB  85 7A       STA  7A
A7DD  90 02       BCC  A7E1
A7DF  E6 7B       INC  7B
A7E1  6C 08 03    JMP  (0308)     JMP $A7E4,a BASIC-utasítás végrehajtása
A7E4  20 73 00    JSR  0073       CHRGET, a következő karakter beolvasása
A7E7  20 ED A7    JSR  A7ED       az utasítás végrehajtása
A7EA  4C AE A7    JMP  A7AE       vissza az interpreter ciklushoz!

*****************************     A BASIC-utasítás végrehajtása
A7ED  F0 3C       BEQ  A82B       ha sorvégjel, akkor kész
A7EF  E9 80       SBC  #80        token?
A7F1  90 11       BCC  A804       ha nem, akkor ugrás a LET-utasításhoz
A7F3  C9 23       CMP  #23
A7F5  B0 17       BCS  A80E       függvényszimbólum vagy GOTO
A7F7  0A          ASL  A          BASIC-utasítás, a kód kétszer
A7F8  A8          TAY
A7F9  B9 0D A0    LDA  A00D,Y     utasításcím beolvasása a táblázatból
A7FC  48          PHA             és tárolása
A7FD  B9 0C A0    LDA  A00C,Y     visszaugrási címként a verembe
A800  48          PHA             az utasítás végrehajtása
A801  4C 73 00    JMP  0073       ugrás a LET-utasításra
A804  4C A5 A9    JMP  A9A5

*****************************
A807  C9 3A       CMP  #3A        ":"
A809  F0 D6       BEQ  A7E1
A80B  4C 08 AF    JMP  AF08       "syntax error"

*****************************     "GO" "TO" kód ellenőrzése
A80E  C9 4B       CMP  #4B        "GO" /minusz $80//
A810  D0 F9       BNE  A80B
A812  20 73 00    JSR  0073       CHRGET rutin, az utolsó karakter beolvasása
A815  A9 A4       LDA  #A4        "TO"
A817  20 FF AE    JSR  AEFF       a kód ellenőrzése
A81A  4C A0 A8    JMP  A8A0       ugrás a GOTO-utasításra

*****************************     A RESTORE BASIC-utasítás
```

```
A81D  38          SEC
A81E  A5 2B       LDA   2B
A820  E9 01       SBC   #01
A822  A4 2C       LDY   2C
A824  B0 01       BCS   A827
A826  88          DEY
A827  85 41       STA   41
A829  84 42       STY   42
A82B  60          RTS

*****************************
A82C  20 E1 FF    JSR   FFE1

*****************************
A82F  B0 01       BCS   A832

*****************************
A831  18          CLC
A832  D0 3C       BNE   A870
A834  A5 7A       LDA   7A
A836  A4 7B       LDY   7B
A838  A6 3A       LDX   3A
A83A  E8          INX
A83B  F0 0C       BEQ   A849
A83D  85 3D       STA   3D
A83F  84 3E       STY   3E
A841  A5 39       LDA   39
A843  A4 3A       LDY   3A
A845  85 3B       STA   3B
A847  84 3C       STY   3C
A849  68          PLA
A84A  68          PLA
A84B  A9 81       LDA   #81
A84D  A0 A3       LDY   #A3
A84F  90 03       BCC   A854
A851  4C 69 A4    JMP   A469
A854  4C 86 E3    JMP   E386

*****************************
A857  D0 17       BNE   A870
A859  A2 1A       LDX   #1A
A85B  A4 3E       LDY   3E
A85D  D0 03       BNE   A862
A85F  4C 37 A4    JMP   A437
A862  A5 3D       LDA   3D
A864  85 7A       STA   7A
A866  84 7B       STY   7B
A868  A5 3B       LDA   3B
A86A  A4 3C       LDY   3C
A86C  85 39       STA   39
A86E  84 3A       STY   3A
A870  60          RTS

*****************************
A871  08          PHP
A872  A9 00       LDA   #00
```

programkezdet - 1

-

egyenlő a DATA-mutatóval

a stop-billentyű ellenőrzése
a stop-billentyű lekérdezése

a STOP BASIC-utasítás
C=1 : STOP

az END BASIC-utasítás
C=0 : END

a programmutató
parancsmód?

igen

a CONT-mutató beállítása

az aktuális sorszám

tárolása a CONT-hoz

visszaugrási cím betöltése a veremből

"break"-mutató
END-kapcsoló?
nem, "break in xxx" kiadása
ugrás BASIC- elegindításhoz

a CONT BASIC-utasítás

'CAN'T CONTINUE hibasorszám
CONT lehetetlen?
nem
hibaüzenet kiadása

a programmutató

és

a sorszám beállítása

a RUN BASIC-utasítás.


```
A874  20 90 FF    JSR   FF90
A877  28          PLP
A878  D0 03       BNE   A87D
A87A  4C 59 A6    JMP   A659
A87D  20 60 A6    JSR   A660
A880  4C 97 A8    JMP   A897

*****************************
A883  A9 03       LDA   #03
A885  20 FB A3    JSR   A3FB
A888  A5 7B       LDA   7B
A88A  48          PHA
A88B  A5 7A       LDA   7A
A88D  48          PHA
A88E  A5 3A       LDA   3A
A890  48          PHA
A891  A5 39       LDA   39
A893  48          PHA
A894  A9 8D       LDA   #8D
A896  48          PHA
A897  20 79 00    JSR   0079
A89A  20 A0 A8    JSR   A8A0
A89D  4C AE A7    JMP   A7AE

*****************************
A8A0  20 6B A9    JSR   A96B
A8A3  20 09 A9    JSR   A909
A8A6  38          SEC
A8A7  A5 39       LDA   39
A8A9  E5 14       SBC   14
A8AB  A5 3A       LDA   3A
A8AD  E5 15       SBC   15
A8AF  B0 0B       BCS   A8BC
A8B1  98          TYA
A8B2  38          SEC
A8B3  65 7A       ADC   7A
A8B5  A6 7B       LDX   7B
A8B7  90 07       BCC   A8C0
A8B9  E8          INX
A8BA  B0 04       BCS   A8C0
A8BC  A5 2B       LDA   2B
A8BE  A6 2C       LDX   2C
A8C0  20 17 A6    JSR   A617
A8C3  90 1E       BCC   A8E3
A8C5  A5 5F       LDA   5F
A8C7  E9 01       SBC   #01
A8C9  85 7A       STA   7A
A8CB  A5 60       LDA   60
A8CD  E9 00       SBC   #00
A8CF  85 7B       STA   7B
A8D1  60          RTS

*****************************
A8D2  D0 FD       BNE   A8D1
A8D4  A9 FF       LDA   #FF
A8D6  85 4A       STA   4A
```

a programmód-kapcsoló beállítása

a következő karakter sorszám?
programmutató a programkezdetre, CLR
CLR-utasítás
GOTO-utasítás

a GOSUB BASIC-utasítás

van elegendő hely a veremben?

programmutató

és a sorszám tárolása

a "GOSUB" kód a verembe

CHRGOT, az utolsó karakter betöltése
GOTO utasítás
ugrás az interpreter ciklusra

a GOTO BASIC-utasítás
a sorszám betöltése a $14/$15 után
következő sorkezdet keresése

kisebb a sorszám az aktuálisnál?

nem

keresés az aktuális sortól

keresés a programkezdettől

a programsor megkeresése
ha nem sikerült, akkor "undef'd statment"

a programmutató beáll. az új sorra

a RETURN BASIC-utasítás

```
      20 8A A3    JSR   A38A      a következő GOSUB-állomány keresése a veremben
A8DB  9A          TXS
A8DC  C9 8D       CMP   #8D       "GOSUB" kód
A8DE  F0 0B       BEQ   A8EB      sikerült megtalálni?
A8E0  A2 0C       LDX   #0C       "return without gosub" hiba sorszáma
A8E2  2C A2 11    BIT   11A2      "undef'd statement" sorszám
A8E5  4C 37 A4    JMP   A437      hibaüzenet kiírása
A8E8  4C 08 AF    JMP   AF08      "syntax error" kiírása
A8EB  68          PLA
A8EC  68          PLA
A8ED  85 39       STA   39        a sorszám beolvasása a veremből
A8EF  68          PLA
A8F0  85 3A       STA   3A
A8F2  68          PLA
A8F3  85 7A       STA   7A        a programmutató betöltése a veremből
A8F5  68          PLA
A8F6  85 7B       STA   7B

*****************************
A8F8  20 06 A9    JSR   A906      a DATA BASIC-utasítás
A8FB  98          TYA             a következő utasítás keresése
A8FC  18          CLC             relatív cím /offset/
A8FD  65 7A       ADC   7A
A8FF  85 7A       STA   7A
A901  90 02       BCC   A905      hozzáadás a programmutatóhoz
A903  E6 7B       INC   7B
A905  60          RTS

*****************************
A906  A2 3A       LDX   #3A       a következő elválasztójel rel.címének megker.
A908  2C A2 00    BIT   00A2      ":" kettőspont
A90B  86 07       STX   07        $00 sor vége
A90D  A0 00       LDY   #00       az Y tartalmazza az offsetet
A90F  84 08       STY   08
A911  A5 08       LDA   08
A913  A6 07       LDX   07        a keresett karakter
A915  85 07       STA   07
A917  86 08       STX   08
A919  B1 7A       LDA   (7A),Y    a karakter betöltése
A91B  F0 E8       BEQ   A905      ha sorvég, akkor kész
A91D  C5 08       CMP   08
A91F  F0 E4       BEQ   A905
A921  C8          INY             a mutató növelése
A922  C9 22       CMP   #22       "(")egyszeres idézőjel
A924  D0 F3       BNE   A919
A926  F0 E9       BEQ   A911

*****************************
A928  20 9E AD    JSR   AD9E      az IF BASIC-utasítás
A92B  20 79 00    JSR   0079      FRMEVL, kifejezés kiszámítása
A92E  C9 89       CMP   #89       CHRGOT rutin, utolsó karakter
A930  F0 05       BEQ   A937      "GOTO" kód
A932  A9 A7       LDA   #A7       igen
A934  20 FF AE    JSR   AEFF      "THEN" kód
A937  A5 61       LDA   61        a kód ellenőrzése
A939  D0 05       BNE   A940      az IF-kifejezés eredménye
                                  igaz a kifejezés?
```



```
*****************************
A93B  20 09 A9    JSR   A909      a REM BASIC-utasítás
A93E  F0 BB       BEQ   A8FB      ha nem, a következő sorkezdet keresése
A940  20 79 00    JSR   0079      a programmutató a következő sorra
A943  B0 03       BCS   A948      CHRGOT rutin, utolsó karakter beolvasása
A945  4C A0 A8    JMP   A8A0      nem számjegy?
A948  4C ED A7    JMP   A7ED      ugrás GOTO utasításra
                                  köv. utasítás dekódolása és végrehajtása

*****************************
A94B  20 9E B7    JSR   B79E      az ON BASIC-utasítás
A94E  48          PHA             egy byte beolvasása /$-tól 255-ig/
A94F  C9 8D       CMP   #8D       a kód tárolása
A951  F0 04       BEQ   A957      a "GOSUB"-kód
A953  C9 89       CMP   #89       igen
A955  D0 91       BNE   A8E8      a "GOTO" kód
A957  C6 65       DEC   65        ha nem, akkor "syntax error"
A959  D0 04       BNE   A95F      a számláló állásának csökkentése
A95B  68          PLA             még mindig nem nulla?
A95C  4C EF A7    JMP   A7EF      igen, a kód visszaolvasása
A95F  20 73 00    JSR   0073      és az utasítás végrehajtása
A962  20 6B A9    JSR   A96B      CHRGET rutin, a köv. karakter beolvasása
A965  C9 2C       CMP   #2C       a sorszám beolvasása
A967  F0 EE       BEQ   A957      "," vessző?
A969  68          PLA             ha igen, akkor tovább
A96A  60          RTS             ha nincs ugrás, a kód visszaolv. és kész

*****************************
A96B  A2 00       LDX   #00       a sorszám betöltése a $14/$15 címekre
A96D  86 14       STX   14        a sorszám kezdőértéke $
A96F  86 15       STX   15
A971  B0 F7       BCS   A96A      ha nem számjegy, akkor kész
A973  E9 2F       SBC   #2F       "$" kódja-1
A975  85 07       STA   07        a hexadecimális szám tárolása
A977  A5 15       LDA   15
A979  85 22       STA   22
A97B  C9 19       CMP   #19       a sorszám nagyobb vagy egyenlő mint 64000?
A97D  B0 D4       BCS   A953      ha igen, akkor "syntax error"
A97F  A5 14       LDA   14
A981  0A          ASL   A         a szám tízszerese
A982  26 22       ROL   22
A984  0A          ASL   A
A985  26 22       ROL   22
A987  65 14       ADC   14
A989  85 14       STA   14
A98B  A5 22       LDA   22
A98D  65 15       ADC   15
A98F  85 15       STA   15
A991  06 14       ASL   14
A993  26 15       ROL   15
A995  A5 14       LDA   14
A997  65 07       ADC   07        a számjegyek összeadása
A999  85 14       STA   14
A99B  90 02       BCC   A99F
A99D  E6 15       INC   15
A99F  20 73 00    JSR   0073      CHRGET rutin, a köv. karakter beolvasása
```

```
A9A2  4C 71 A9     JMP   A971

*****************************
A9A5  20 8B B0     JSR   B08B
A9A8  85 49        STA   49
A9AA  84 4A        STY   4A
A9AC  A9 B2        LDA   #B2
A9AE  20 FF AE     JSR   AEFF
A9B1  A5 0E        LDA   0E
A9B3  48           PHA
A9B4  A5 0D        LDA   0D
A9B6  48           PHA
A9B7  20 9E AD     JSR   AD9E
A9BA  68           PLA
A9BB  2A           ROL   A
A9BC  20 90 AD     JSR   AD90
A9BF  D0 18        BNE   A9D9
A9C1  68           PLA
A9C2  10 12        BPL   A9D6

*****************************
A9C4  20 1B BC     JSR   BC1B
A9C7  20 BF B1     JSR   B1BF
A9CA  A0 00        LDY   #00
A9CC  A5 64        LDA   64
A9CE  91 49        STA   (49),Y
A9D0  C8           INY
A9D1  A5 65        LDA   65
A9D3  91 49        STA   (49),Y
A9D5  60           RTS

*****************************
A9D6  4C D0 BB     JMP   BBD0

*****************************
A9D9  68           PLA
A9DA  A4 4A        LDY   4A
A9DC  C0 BF        CPY   #BF
A9DE  D0 4C        BNE   AA2C
A9E0  20 A6 B6     JSR   B6A6
A9E3  C9 06        CMP   #06
A9E5  D0 3D        BNE   AA24
A9E7  A0 00        LDY   #00
A9E9  84 61        STY   61
A9EB  84 66        STY   66
A9ED  84 71        STY   71
A9EF  20 1D AA     JSR   AA1D
A9F2  20 E2 BA     JSR   BAE2
A9F5  E6 71        INC   71
A9F7  A4 71        LDY   71
A9F9  20 1D AA     JSR   AA1D
A9FC  20 0C BC     JSR   BC0C
A9FF  AA           TAX
AA00  F0 05        BEQ   AA07
AA02  E8           INX
AA03  8A           TXA
```

és folytatás

a LET BASIC-utasítás
a változó keresése

a változó címének tárolása
"=" kód
a kód ellenőrzése
integer-kapcsoló

és a tipus /füzér, numerikus/ tárolása

a FRMEVL kifejezés betöltése

a tipus visszaolvasása
a tipus ellenőrzése

REAL?

INTEGER értékadás
FAC kerekítés
és konvertálás integer-re

az érték betöltése a változóba

REAL értékadás
FAC

füzér-értékadás

a cím felső byte-ja
a változó TIS?
nem
FRESTR
a füzér hossza= 6-tal
ha nem, akkor "illegal quantity"

a következő karakter számjegy
FAC = FAC * 10
helyiérték számláló növelése

következő karakter számjegy
ARG értéke FAC-be kerül

a FAC nullával egyenlő?


```
AA04  20 ED BA     JSR   BAED
AA07  A4 71        LDY   71
AA09  C8           INY
AA0A  C0 06        CPY   #06
AA0C  D0 DF        BNE   A9ED
AA0E  20 E2 BA     JSR   BAE2
AA11  20 9B BC     JSR   BC9B
AA14  A6 64        LDX   64
AA16  A4 63        LDY   63
AA18  A5 65        LDA   65
AA1A  4C DB FF     JMP   FFDB

*****************************
AA1D  B1 22        LDA   (22),Y
AA1F  20 80 00     JSR   0080
AA22  90 03        BCC   AA27
AA24  4C 48 B2     JMP   B248
AA27  E9 2F        SBC   #2F
AA29  4C 7E BD     JMP   BD7E

*****************************
AA2C  A0 02        LDY   #02
AA2E  B1 64        LDA   (64),Y
AA30  C5 34        CMP   34
AA32  90 17        BCC   AA4B
AA34  D0 07        BNE   AA3D
AA36  88           DEY
AA37  B1 64        LDA   (64),Y
AA39  C5 33        CMP   33
AA3B  90 0E        BCC   AA4B
AA3D  A4 65        LDY   65
AA3F  C4 2E        CPY   2E
AA41  90 08        BCC   AA4B
AA43  D0 0D        BNE   AA52
AA45  A5 64        LDA   64
AA47  C5 2D        CMP   2D
AA49  B0 07        BCS   AA52
AA4B  A5 64        LDA   64
AA4D  A4 65        LDY   65
AA4F  4C 68 AA     JMP   AA68
AA52  A0 00        LDY   #00
AA54  B1 64        LDA   (64),Y
AA56  20 75 B4     JSR   B475
AA59  A5 50        LDA   50
AA5B  A4 51        LDY   51
AA5D  85 6F        STA   6F
AA5F  84 70        STY   70
AA61  20 7A B6     JSR   B67A
AA64  A9 61        LDA   #61
AA66  A0 00        LDY   #00
AA68  85 50        STA   50
AA6A  84 51        STY   51
AA6C  20 DB B6     JSR   B6DB
AA6F  A0 00        LDY   #00
AA71  B1 50        LDA   (50),Y
AA73  91 49        STA   (49),Y
```

FAC = FAC + ARG
a helyiérték számláló
növelése
már 6 helyiérték?

FAC = FAC * 10
FAC jobbra igazítása

beírt idő

az idő beállítása

karakter ellenőrzése
a karakter számjegy
számjegy vizsgálata

"illegal quantity"
ASCII/hexadecimális konvertálás
átvitel a FAC-be és az ARG-be

a füzér értékadása

a füzércím felső byte
összehasonlítása a füzérek kezdőcímével
ha kisebb, a füzér a programon belül van

füzércím alsó byte
összehasonlítása
füzér a programban

a füzér hosszúsága
a tárcím ellenőrzése, füzérmutató beállítása

a füzér átvitele a füzérterületre

törlés a füzér-veremből

a hosszúság

```
AA75  C8          INY
AA76  B1 50       LDA  (50),Y     a cím alsó byte
AA78  91 49       STA  (49),Y
AA7A  C8          INY
AA7B  B1 50       LDA  (50),Y     és felső byte
AA7D  91 49       STA  (49),Y     betöltése a változóba
AA7F  60          RTS

*****************************     a PRINT# BASIC-utasítás
AA80  20 86 AA    JSR  AA86       a CMD-utasítás
AA83  4C B5 AB    JMP  ABB5       és a CLRCH

*****************************     a CMD BASIC-utasítás
AA86  20 9E B7    JSR  B79E       a byte beolvasása
AA89  F0 05       BEQ  AA90
AA8B  A9 2C       LDA  #2C        ","
AA8D  20 FF AE    JSR  AEFF       a vessző ellenőrzése
AA90  08          PHP
AA91  86 13       STX  13         az output egység számának tárolása
AA93  20 18 E1    JSR  E118       CKOUT, output egység beállítása
AA96  28          PLP
AA97  4C A0 AA    JMP  AAA0       a PRINT-utasítás
AA9A  20 21 AB    JSR  AB21       a füzér nyomtatása
AA9D  20 79 00    JSR  0079       CHRGOT rutin, az utolsó karakter

*****************************     a PRINT BASIC-utasítás
AAA0  F0 35       BEQ  AAD7
AAA2  F0 43       BEQ  AAE7
AAA4  C9 A3       CMP  #A3        "TAB( " kód
AAA6  F0 50       BEQ  AAF8
AAA8  C9 A6       CMP  #A6        "SPC(" kód
AAAA  18          CLC
AAAB  F0 4B       BEQ  AAF8
AAAD  C9 2C       CMP  #2C        ","
AAAF  F0 37       BEQ  AAE8
AAB1  C9 3B       CMP  #3B        ";"
AAB3  F0 5E       BEQ  AB13
AAB5  20 9E AD    JSR  AD9E       FRMEVL, a kifejezés beolvasása
AAB8  24 0D       BIT  0D         tipus
AABA  30 DE       BMI  AA9A       füzér?
AABC  20 DD BD    JSR  BDDD       FAC/ASCII a füzér átalakítása
AABF  20 87 B4    JSR  B487       a füzér-paraméter beolvasása
AAC2  20 21 AB    JSR  AB21       a füzér kinyomtatása
AAC5  20 3B AB    JSR  AB3B       a kurzor jobbra, ill. az üres jel kiírása
AAC8  D0 D3       BNE  AA9D       folytatás
AACA  A9 00       LDA  #00        az input-puffer
AACC  9D 00 02    STA  0200,X     00-val lezárása
AACF  A2 FF       LDX  #FF
AAD1  A0 01       LDY  #01        az input-puffer mutató beállítása
AAD3  A5 13       LDA  13         az output egység száma
AAD5  D0 10       BNE  AAE7
AAD7  A9 0D       LDA  #0D        "CR" carriage return
AAD9  20 47 AB    JSR  AB47       kiírása
AADC  24 13       BIT  13         logikai file-szám
AADE  10 05       BPL  AAE5       kisebb mint 128?
AAE0  A9 0A       LDA  #0A        "LF" line feed
```



```
AAE2  20 47 AB    JSR  AB47       kiírása
AAE5  49 FF       EOR  #FF
AAE7  60          RTS
AAE8  38          SEC             tizes-tabulátor, vesszővel
AAE9  20 F0 FF    JSR  FFF0       a kurzorpozició beolvasása
AAEC  98          TYA
AAED  38          SEC
AAEE  E9 0A       SBC  #0A        minusz 10
AAF0  B0 FC       BCS  AAEE       nem negatív?
AAF2  49 FF       EOR  #FF        invertálás
AAF4  69 01       ADC  #01        egyes hozzáadása
AAF6  D0 16       BNE  AB0E

*****************************     TAB( (C=1) és SPC( (C=0)
AAF8  08          PHP             kapcsoló tárolása
AAF9  38          SEC
AAFA  20 F0 FF    JSR  FFF0       a kurzorpozició beolvasása
AAFD  84 09       STY  09         és tárolása
AAFF  20 9B B7    JSR  B79B       a byte beolvasása
AB02  C9 29       CMP  #29        ") " bezáró zárójel?
AB04  D0 59       BNE  AB5F       nem, akkor "syntax error"
AB06  28          PLP
AB07  90 06       BCC  AB0F       ugrás az SPC( - re
AB09  8A          TXA             az akku-ban lévő TAB érték összehasonlítása
AB0A  E5 09       SBC  09         a kurzorpozicióval
AB0C  90 05       BCC  AB13       ha az előbbi kisebb, kész
AB0E  AA          TAX
AB0F  E8          INX
AB10  CA          DEX
AB11  D0 06       BNE  AB19
AB13  20 73 00    JSR  0073       CHRGET rutin, a köv. karakter beolvasása
AB16  4C A2 AA    JMP  AAA2       és folytatás
AB19  20 3B AB    JSR  AB3B       a kurzor jobbra, ill. üres jel kiírása
AB1C  D0 F2       BNE  AB10       ugrás a ciklus elejére

*****************************     a füzér kiírása
AB1E  20 87 B4    JSR  B487       a füzér paraméter beolvasása
AB21  20 A6 B6    JSR  B6A6       FRESTR
AB24  AA          TAX             a füzér hossza
AB25  A0 00       LDY  #00
AB27  E8          INX
AB28  CA          DEX
AB29  F0 BC       BEQ  AAE7       a füzér vége?
AB2B  B1 22       LDA  (22),Y     a karakterek kiírása
AB2D  20 47 AB    JSR  AB47
AB30  C8          INY
AB31  C9 0D       CMP  #0D        a "CR"  carriage return
AB33  D0 F3       BNE  AB28       ha nem, akkor tovább
AB35  20 E5 AA    JSR  AAE5       hiba! az LF kiírása
AB38  4C 28 AB    JMP  AB28       és folytatás

*****************************     egy üres jel vagy a kurzor jobbra kiírása
AB3B  A5 13       LDA  13         kiírás a file-ba?
AB3D  F0 03       BEQ  AB42       nem, képernyőre, ekkor kurzor jobbra
AB3F  A9 20       LDA  #20        " " üres jel
AB41  2C A9 1D    BIT  1DA9       kurzor jobbra
```

```
AB44  2C A9 3F     BIT   3FA9
AB47  20 0C E1     JSR   E10C
AB4A  29 FF        AND   #FF
AB4C  60           RTS

*******************************
AB4D  A5 11        LDA   11
AB4F  F0 11        BEQ   AB62
AB51  30 04        BMI   AB57
AB53  A0 FF        LDY   #FF
AB55  D0 04        BNE   AB5B

*******************************
AB57  A5 3F        LDA   3F
AB59  A4 40        LDY   40

*******************************
AB5B  85 39        STA   39
AB5D  84 3A        STY   3A
AB5F  4C 08 AF     JMP   AF08

*******************************
AB62  A5 13        LDA   13
AB64  F0 05        BEQ   AB6B
AB66  A2 18        LDX   #18
AB68  4C 37 A4     JMP   A437
AB6B  A9 0C        LDA   #0C
AB6D  A0 AD        LDY   #AD
AB6F  20 1E AB     JSR   AB1E
AB72  A5 3D        LDA   3D
AB74  A4 3E        LDY   3E
AB76  85 7A        STA   7A
AB78  84 7B        STY   7B
AB7A  60           RTS

*******************************
AB7B  20 A6 B3     JSR   B3A6
AB7E  C9 23        CMP   #23
AB80  D0 10        BNE   AB92
AB82  20 73 00     JSR   0073
AB85  20 9E B7     JSR   B79E
AB88  A9 2C        LDA   #2C
AB8A  20 FF AE     JSR   AEFF
AB8D  86 13        STX   13
AB8F  20 1E E1     JSR   E11E
AB92  A2 01        LDX   #01
AB94  A0 02        LDY   #02
AB96  A9 00        LDA   #00
AB98  8D 01 02     STA   0201
AB9B  A9 40        LDA   #40
AB9D  20 0F AC     JSR   AC0F
ABA0  A6 13        LDX   13
ABA2  D0 13        BNE   ABB7
ABA4  60           RTS

*******************************
```

"?" kérdőjel kiírása
a kapcsolók beállítása

a hibakezelés beolvasásnál
INPUT-/GET-/READ-kapcsoló
INPUT
READ

GET

hiba a READ-nél
a DATA sorszáma

hiba a GET-nél

ha egyenlő a hiba sorszámával,
akkor "syntax error"

hiba az INPUT-nál
az input egységek száma
billentyűzet?
"file data" sorszám
a hibaüzenet kiírása

"?redo from start"
szöveg kiírása

a programmutató visszaállítása

az INPUT-utasításra

a GET BASIC-utasítás
közvetlen mód ellenőrzése
"#"
nem?
CLRGET rutin, a köv. karakter beolvasása
a byte tartalma
"," vessző
a kód ellenőrzése

CLRCH

mutató a puffer végére=$02$1-en
a puffer lezárása 0-val

a GET-kapcsoló
értékadás
az input egység
ha nem billentyűzet, akkor CLRCH

az INPUT# BASIC-utasítás


```
ABA5  20 9E B7     JSR   B79E
ABA8  A9 2C        LDA   #2C
ABAA  20 FF AE     JSR   AEFF
ABAD  86 13        STX   13
ABAF  20 1E E1     JSR   E11E
ABB2  20 CE AB     JSR   ABCE
ABB5  A5 13        LDA   13
ABB7  20 CC FF     JSR   FFCC
ABBA  A2 00        LDX   #00
ABBC  86 13        STX   13
ABBE  60           RTS

*******************************
ABBF  C9 22        CMP   #22
ABC1  D0 0B        BNE   ABCE
ABC3  20 BD AE     JSR   AEBD
ABC6  A9 3B        LDA   #3B
ABC8  20 FF AE     JSR   AEFF
ABCB  20 21 AB     JSR   AB21
ABCE  20 A6 B3     JSR   B3A6
ABD1  A9 2C        LDA   #2C
ABD3  8D FF 01     STA   01FF
ABD6  20 F9 AB     JSR   ABF9
ABD9  A5 13        LDA   13
ABDB  F0 0D        BEQ   ABEA
ABDD  20 B7 FF     JSR   FFB7
ABE0  29 02        AND   #02
ABE2  F0 06        BEQ   ABEA
ABE4  20 B5 AB     JSR   ABB5
ABE7  4C F8 A8     JMP   A8F8
ABEA  AD 00 02     LDA   0200
ABED  D0 1E        BNE   AC0D
ABEF  A5 13        LDA   13
ABF1  D0 E3        BNE   ABD6
ABF3  20 06 A9     JSR   A906
ABF6  4C FB A8     JMP   A8FB
ABF9  A5 13        LDA   13
ABFB  D0 06        BNE   AC03
ABFD  20 45 AB     JSR   AB45
AC00  20 3B AB     JSR   AB3B
AC03  4C 60 A5     JMP   A560

*******************************
AC06  A6 41        LDX   41
AC08  A4 42        LDY   42
AC0A  A9 98        LDA   #98
AC0C  2C A9 00     BIT   00A9
AC0F  85 11        STA   11
AC11  86 43        STX   43
AC13  84 44        STY   44
AC15  20 8B B0     JSR   B08B
AC18  85 49        STA   49
AC1A  84 4A        STY   4A
AC1C  A5 7A        LDA   7A
AC1E  A4 7B        LDY   7B
AC20  85 4B        STA   4B
```

a byte beolvasása
","
a vessző ellenőrzése
az input egység
CHKIN
INPUT szöveg nélkül

a CLRCH visszaállítja az input egységet

az input egység ismét a billentyűzet

az INPUT BASIC-utasítás
"(")" egyszeres idézőjel
nem
a szöveg beolvasása
";" pontosvessző
a kód ellenőrzése
a füzér kiírása
a parancsmód ellenőrzése
"," vessző
a puffer kezdetnél
a kérdőjel kiírása
az input egység száma
billentyűzet?
a státusz beolvasása

Time-out?
igen, CLRCH, billentyűzetet újra aktivizálni!
a következő utasítás végrehajtása
az első karakter beolvasása
vége?
igen, az input egység
nem a billentyűzet?
a következő utasítás offsetje
a programmutató a következő utasításra
az input egység
nem a billentyűzet?
"?" kiírása
" " üresjel kiírása
beviteli sor beolvasása

a READ BASIC-utasítás

a DATA-mutató betöltése
a READ-kapcsoló

beállítása

az INPUT-mutató a beolvasásra
a változó keresése

a változó címének tárolása

a programmutató közbenső tárolása a $4B/$4C-ben

```
AC22  84 4C       STY   4C
AC24  A6 43       LDX   43
AC26  A4 44       LDY   44
AC28  86 7A       STX   7A
AC2A  84 7B       STY   7B
AC2C  20 79 00    JSR   0079
AC2F  D0 20       BNE   AC51
AC31  24 11       BIT   11
AC33  50 0C       BVC   AC41
AC35  20 24 E1    JSR   E124
AC38  8D 00 02    STA   0200
AC3B  A2 FF       LDX   #FF
AC3D  A0 01       LDY   #01
AC3F  D0 0C       BNE   AC4D
AC41  30 75       BMI   ACB8
AC43  A5 13       LDA   13
AC45  D0 03       BNE   AC4A
AC47  20 45 AB    JSR   AB45
AC4A  20 F9 AB    JSR   ABF9
AC4D  86 7A       STX   7A
AC4F  84 7B       STY   7B
AC51  20 73 00    JSR   0073
AC54  24 0D       BIT   0D
AC56  10 31       BPL   AC89
AC58  24 11       BIT   11
AC5A  50 09       BVC   AC65
AC5C  E8          INX
AC5D  86 7A       STX   7A
AC5F  A9 00       LDA   #00
AC61  85 07       STA   07
AC63  F0 0C       BEQ   AC71
AC65  85 07       STA   07
AC67  C9 22       CMP   #22
AC69  F0 07       BEQ   AC72
AC6B  A9 3A       LDA   #3A
AC6D  85 07       STA   07
AC6F  A9 2C       LDA   #2C
AC71  18          CLC
AC72  85 08       STA   08
AC74  A5 7A       LDA   7A
AC76  A4 7B       LDY   7B
AC78  69 00       ADC   #00
AC7A  90 01       BCC   AC7D
AC7C  C8          INY
AC7D  20 8D B4    JSR   B48D
AC80  20 E2 B7    JSR   B7E2
AC83  20 DA A9    JSR   A9DA
AC86  4C 91 AC    JMP   AC91
AC89  20 F3 BC    JSR   BCF3
AC8C  A5 0E       LDA   0E
AC8E  20 C2 A9    JSR   A9C2
AC91  20 79 00    JSR   0079
AC94  F0 07       BEQ   AC9D
AC96  C9 2C       CMP   #2C
AC98  F0 03       BEQ   AC9D
AC9A  4C 4D AB    JMP   AB4D
```

az INPUT-mutató
egyenlő a programmutatóval

CHRGOT rutin, az utolsó karakter beolvasása

a beviteli kapcsoló
nem GET?
GETIN
a karakter beírása a pufferba

az input egység
nem billentyűzet?
a kérdőjel kiírása
második kérdőjel kiírása
programmutató beállítása

CHRGET rutin, a köv. karakter beolvasása
a tipus

a beviteli kapcsoló
nem GET?

"(")" egyszeres idézőjel

":"

","

a füzér átvétele
a programmutató beállítása a füzér mögé
a füzér hozzárendelése a változóhoz
folytatás
számjegy betöltése a FAC-be
INTEGER/REAL kapcsoló
a FAC hozzárend. a numerikus változóhoz
CHRGOT rutin, az utolsó karakter beolvasása
vége?
","

ugrás a hibakezeléshez


```
AC9D  A5 7A       LDA   7A
AC9F  A4 7B       LDY   7B
ACA1  85 43       STA   43
ACA3  84 44       STY   44
ACA5  A5 4B       LDA   4B
ACA7  A4 4C       LDY   4C
ACA9  85 7A       STA   7A
ACAB  84 7B       STY   7B
ACAD  20 79 00    JSR   0079
ACB0  F0 2D       BEQ   ACDF
ACB2  20 FD AE    JSR   AEFD
ACB5  4C 15 AC    JMP   AC15
ACB8  20 06 A9    JSR   A906
ACBB  C8          INY
ACBC  AA          TAX
ACBD  D0 12       BNE   ACD1
ACBF  A2 0D       LDX   #0D
ACC1  C8          INY
ACC2  B1 7A       LDA   (7A),Y
ACC4  F0 6C       BEQ   AD32
ACC6  C8          INY
ACC7  B1 7A       LDA   (7A),Y
ACC9  85 3F       STA   3F
ACCB  C8          INY
ACCC  B1 7A       LDA   (7A),Y
ACCE  C8          INY
ACCF  85 40       STA   40
ACD1  20 FB AB    JSR   ABFB
ACD4  20 79 00    JSR   0079
ACD7  AA          TAX
ACD8  E0 83       CPX   #83
ACDA  D0 DC       BNE   ACB8
ACDC  4C 51 AC    JMP   AC51
ACDF  A5 43       LDA   43
ACE1  A4 44       LDY   44
ACE3  A6 11       LDX   11
ACE5  10 03       BPL   ACEA
ACE7  4C 27 A8    JMP   A827
ACEA  A0 00       LDY   #00
ACEC  B1 43       LDA   (43),Y
ACEE  F0 0B       BEQ   ACFB
ACF0  A5 13       LDA   13
ACF2  D0 07       BNE   ACFB
ACF4  A9 FC       LDA   #FC
ACF6  A0 AC       LDY   #AC
ACF8  4C 1E AB    JMP   AB1E
ACFB  60          RTS
ACFC  3F 45 58 54 52 41 20 49
AD04  47 4E 4F 52 45 44 0D 00
AD0C  3F 52 45 44 4F 20 46 52
AD14  4F 4D 20 53 54 41 52 54 0D
AD1C  00 00
**********************************
AD1E  D0 04       BNE   AD24
AD20  A0 00       LDY   #00
```

a programmutató

egyenlő a DATA-mutatóval

a programmutató
visszatöltése

CHRGOT rutin, az utolsó karakter beolvasása

CKCOM, a vessző ellenőrzése
tovább
a következő utasítás keresése

a sornak vége?
nem

program vége? "out of data" X=Ø

a programmutató a következő utasításra
CHRGOT rutin, az utolsó karakter beolvasása

"DATA"
nem, tovább keresés
adatolvasás

az input-mutató
az input-kapcsoló
nem DATA?
a DATA-mutató beállítása

"?extra ignored" szöveg kiírása

"?extra ignored"

"?redo from start"

a NEXT BASIC-utasítás

változónév következik?

```
AD22  F0 03      BEQ   AD27
AD24  20 8B B0   JSR   B08B      a változó keresése
AD27  85 49      STA   49
AD29  84 4A      STY   4A        a változó címe
AD2B  20 8A A3   JSR   A38A      a FOR-NEXT ciklus keres a veremben
AD2E  F0 05      BEQ   AD35      sikerült megtalálni
AD30  A2 0A      LDX   #0A       "next without for" sorszáma
AD32  4C 37 A4   JMP   A437      a hibaüzenet kiírása
AD35  9A         TXS
AD36  8A         TXA
AD37  18         CLC
AD38  69 04      ADC   #04
AD3A  48         PHA
AD3B  69 06      ADC   #06
AD3D  85 24      STA   24
AD3F  68         PLA
AD40  A0 01      LDY   #01
AD42  20 A2 BB   JSR   BBA2      a változó betöltése a veremből a FAC-be
AD45  BA         TSX
AD46  BD 09 01   LDA   0109,X
AD49  85 66      STA   66
AD4B  A5 49      LDA   49        a változó címe
AD4D  A4 4A      LDY   4A
AD4F  20 67 B8   JSR   B867      a STEP-érték hozzáadása a FAC-hez
AD52  20 D0 BB   JSR   BBD0      FAC/változó konvertálás
AD55  A0 01      LDY   #01
AD57  20 5D BC   JSR   BC5D      a FAC összehasonlítása a végértékkel
AD5A  BA         TSX
AD5B  38         SEC
AD5C  FD 09 01   SBC   0109,X
AD5F  F0 17      BEQ   AD78
AD61  BD 0F 01   LDA   010F,X
AD64  85 39      STA   39
AD66  BD 10 01   LDA   0110,X    a sorszám betöltése
AD69  85 3A      STA   3A
AD6B  BD 12 01   LDA   0112,X
AD6E  85 7A      STA   7A
AD70  BD 11 01   LDA   0111,X    a programmutató betöltése
AD73  85 7B      STA   7B
AD75  4C AE A7   JMP   A7AE      vissza az interpreter ciklushoz
AD78  8A         TXA
AD79  69 11      ADC   #11
AD7B  AA         TAX
AD7C  9A         TXS
AD7D  20 79 00   JSR   0079      CHRGOT, az utolsó karakter beolvasása
AD80  C9 2C      CMP   #2C       "," vessző
AD82  D0 F1      BNE   AD75      ha nem, akkor kész
AD84  20 73 00   JSR   0073      CHRGET, a következő karakter beolvasása
AD87  20 24 AD   JSR   AD24      a következő ciklusváltozó

******************************   FRMNUM kifejezés beolvasása
AD8A  20 9E AD   JSR   AD9E      és numerikus ellenőrzése FRMEVL kifej.betölt.

******************************   numerikus ellenőrzés
AD8D  18         CLC
AD8E  24         .BYTE #24
```



```
******************************   karakteres ellenőrzés
AD8F  38         SEC
AD90  24 0D      BIT   0D        tipus ellenőrzés
AD92  30 03      BMI   AD97
AD94  B0 03      BCS   AD99
AD96  60         RTS
AD97  B0 FD      BCS   AD96
AD99  A2 16      LDX   #16       a "type mismatch"
AD9B  4C 37 A4   JMP   A437      hibaüzenet kiírása

******************************   FRMEVL - egy tetszőleges kifejezés
AD9E  A6 7A      LDX   7A        kiértékelése
ADA0  D0 02      BNE   ADA4      a programmutató 1-gyel csökken
ADA2  C6 7B      DEC   7B
ADA4  C6 7A      DEC   7A
ADA6  A2 00      LDX   #00
ADA8  24 48      BIT   48
ADAA  8A         TXA
ADAB  48         PHA
ADAC  A9 01      LDA   #01
ADAE  20 FB A3   JSR   A3FB      van-e elegendő hely a veremben?
ADB1  20 83 AE   JSR   AE83      a következő elem beolvasása
ADB4  A9 00      LDA   #00
ADB6  85 4D      STA   4D        összehasonlító operátor maszkja
ADB8  20 79 00   JSR   0079      CHRGOT rutin, az utolsó karakter beolvasása
ADBB  38         SEC
ADBC  E9 B1      SBC   #B1
ADBE  90 17      BCC   ADD7
ADC0  C9 03      CMP   #03
ADC2  B0 13      BCS   ADD7
ADC4  C9 01      CMP   #01
ADC6  2A         ROL   A
ADC7  49 01      EOR   #01
ADC9  45 4D      EOR   4D
ADCB  C5 4D      CMP   4D
ADCD  90 61      BCC   AE30
ADCF  85 4D      STA   4D
ADD1  20 73 00   JSR   0073      CHRGET rutin, a köv. karakter beolvasása
ADD4  4C BB AD   JMP   ADBB      vissza
ADD7  A6 4D      LDX   4D
ADD9  D0 2C      BNE   AE07
ADDB  B0 7B      BCS   AE58
ADDD  69 07      ADC   #07
ADDF  90 77      BCC   AE58
ADE1  65 0D      ADC   0D
ADE3  D0 03      BNE   ADE8
ADE5  4C 3D B6   JMP   B63D      numerikus?
ADE8  69 FF      ADC   #FF      karakter-láncolás
ADEA  85 22      STA   22        a kód $AA
ADEC  0A         ASL   A
ADED  65 22      ADC   22        3-szorosának
ADEF  A8         TAY
ADF0  68         PLA
ADF1  D9 80 A0   CMP   A080,Y    összehasonlítása a hierarchia-kapcsolóval
ADF4  B0 67      BCS   AE5D
```

0 <
1 =
2 >

kisebb, egyenlő vagy nagyobb?

```
ADF6  20 8D AD    JSR   AD8D      numerikus ellenőrzés
ADF9  48          PHA
ADFA  20 20 AE    JSR   AE20      az operátor címe és az operandus a veremben
ADFD  68          PLA
ADFE  A4 4B       LDY   4B
AE00  10 17       BPL   AE19
AE02  AA          TAX
AE03  F0 56       BEQ   AE5B
AE05  D0 5F       BNE   AE66
AE07  46 0D       LSR   0D        a füzér-kapcsoló törlése
AE09  8A          TXA
AE0A  2A          ROL   A
AE0B  A6 7A       LDX   7A
AE0D  D0 02       BNE   AE11      a programmutató 1-gyel csökken
AE0F  C6 7B       DEC   7B
AE11  C6 7A       DEC   7A
AE13  A0 1B       LDY   #1B       hierarchia-kapcsoló offset /rel. címe/
AE15  85 4D       STA   4D        a kapcsoló vissza
AE17  D0 D7       BNE   ADF0
AE19  D9 80 A0    CMP   A080,Y    összehasonlítás a hierarchia-kapcsolóval
AE1C  B0 48       BCS   AE66
AE1E  90 D9       BCC   ADF9
AE20  B9 82 A0    LDA   A082,Y
AE23  48          PHA             az operátor címe a veremben
AE24  B9 81 A0    LDA   A081,Y
AE27  48          PHA
AE28  20 33 AE    JSR   AE33      operandus a veremben
AE2B  A5 4D       LDA   4D
AE2D  4C A9 AD    JMP   ADA9      vissza a ciklus kezdetéhez
AE30  4C 08 AF    JMP   AF08      "syntax error"
AE33  A5 66       LDA   66        előjel
AE35  BE 80 A0    LDX   A080,Y    a hierarchia-kapcsoló
AE38  A8          TAY
AE39  68          PLA
AE3A  85 22       STA   22
AE3C  E6 22       INC   22        visszaugrási cím tárolása
AE3E  68          PLA
AE3F  85 23       STA   23
AE41  98          TYA             előjel
AE42  48          PHA
AE43  20 1B BC    JSR   BC1B      FAC kerekítése
AE46  A5 65       LDA   65
AE48  48          PHA
AE49  A5 64       LDA   64
AE4B  48          PHA
AE4C  A5 63       LDA   63
AE4E  48          PHA             FAC a veremben
AE4F  A5 62       LDA   62
AE51  48          PHA
AE52  A5 61       LDA   61
AE54  48          PHA
AE55  6C 22 00    JMP   (0022)    ugrás a műveletre
AE58  A0 FF       LDY   #FF
AE5A  68          PLA
AE5B  F0 23       BEQ   AE80
AE5D  C9 64       CMP   #64
```



```
AE5F  F0 03       BEQ   AE64
AE61  20 8D AD    JSR   AD8D      numerikus ellenőrzés
AE64  84 4B       STY   4B
AE66  68          PLA
AE67  4A          LSR   A
AE68  85 12       STA   12
AE6A  68          PLA
AE6B  85 69       STA   69
AE6D  68          PLA
AE6E  85 6A       STA   6A
AE70  68          PLA             ARG betöltése a veremből
AE71  85 6B       STA   6B
AE73  68          PLA
AE74  85 6C       STA   6C
AE76  68          PLA
AE77  85 6D       STA   6D
AE79  68          PLA
AE7A  85 6E       STA   6E
AE7C  45 66       EOR   66
AE7E  85 6F       STA   6F
AE80  A5 61       LDA   61
AE82  60          RTS

*****************************
AE83  6C 0A 03    JMP   (030A)    egy kifejezés köv. elemének beolvasása
AE86  A9 00       LDA   #00       JMP $AE86
AE88  85 0D       STA   0D        a tipus numerikus ellenőrzése
AE8A  20 73 00    JSR   0073      CHRGET rutin, a köv. karakter beolvasása
AE8D  B0 03       BCS   AE92      számjegy?
AE8F  4C F3 BC    JMP   BCF3      a változó betöltése
AE92  20 13 B1    JSR   B113      betű?
AE95  90 03       BCC   AE9A      nem
AE97  4C 28 AF    JMP   AF28      a változó betöltése
AE9A  C9 FF       CMP   #FF      π BASIC-kódja?
AE9C  D0 0F       BNE   AEAD
AE9E  A9 A8       LDA   #A8       π konstans mutatója
AEA0  A0 AE       LDY   #AE
AEA2  20 A2 BB    JSR   BBA2      konstans betöltése
AEA5  4C 73 00    JMP   0073      CHRGET rutin, a köv. karakter beolvasása

*****************************
AEA8  82 49 0F DA A1              π konstans 3.14159265
AEAD  C9 2E       CMP   #2E       "."
AEAF  F0 DE       BEQ   AE8F
AEB1  C9 AB       CMP   #AB       "-"
AEB3  F0 58       BEQ   AF0D      előjel-cserél
AEB5  C9 AA       CMP   #AA       "+"
AEB7  F0 D1       BEQ   AE8A
AEB9  C9 22       CMP   #22       "(*)"
AEBB  D0 0F       BNE   AECC
AEBD  A5 7A       LDA   7A
AEBF  A4 7B       LDY   7B        a programmutató betöltése
AEC1  69 00       ADC   #00
AEC3  90 01       BCC   AEC6
AEC5  C8          INY
AEC6  20 87 B4    JSR   B487      a szöveg átvitele
```

```
AEC9  4C E2 B7     JMP   B7E2
AECC  C9 AB        CMP   #AB
AECE  D0 13        BNE   AEE3
AED0  A0 18        LDY   #18
AED2  D0 3B        BNE   AF0F

*****************************
AED4  20 BF B1     JSR   B1BF
AED7  A5 65        LDA   65
AED9  49 FF        EOR   #FF
AEDB  A8           TAY
AEDC  A5 64        LDA   64
AEDE  49 FF        EOR   #FF
AEE0  4C 91 B3     JMP   B391

*****************************
AEE3  C9 A5        CMP   #A5
AEE5  D0 03        BNE   AEEA
AEE7  4C F4 B3     JMP   B3F4

*****************************
AEEA  C9 B4        CMP   #B4
AEEC  90 03        BCC   AEF1
AEEE  4C A7 AF     JMP   AFA7

*****************************
AEF1  20 FA AE     JSR   AEFA
AEF4  20 9E AD     JSR   AD9E

*****************************
AEF7  A9 29        LDA   #29
AEF9  2C A9 28     BIT   28A9
AEFC  2C A9 2C     BIT   2CA9
AEFF  A0 00        LDY   #00
AF01  D1 7A        CMP   (7A),Y
AF03  D0 03        BNE   AF08
AF05  4C 73 00     JMP   0073
AF08  A2 0B        LDX   #0B
AF0A  4C 37 A4     JMP   A437
AF0D  A0 15        LDY   #15
AF0F  68           PLA
AF10  68           PLA
AF11  4C FA AD     JMP   ADFA

*****************************
AF14  38           SEC
AF15  A5 64        LDA   64
AF17  E9 00        SBC   #00
AF19  A5 65        LDA   65
AF1B  E9 A0        SBC   #A0
AF1D  90 08        BCC   AF27
AF1F  A9 A2        LDA   #A2
AF21  E5 64        SBC   64
AF23  A9 E3        LDA   #E3
AF25  E5 65        SBC   65
AF27  60           RTS
```

a programmutató= a szövegvég+1
"NOT" kód

hierarchia kapcs. [illegible] a táblázatban

a NOT BASIC utasítás
FAC/INTEGER konvertálás

összes bit megfordítása

átalakítás lebegőpontossá

"FN" kód

az FN végrehajtása

"SGN" kód
kisebb /nem szövegfüggvény/?
a füzér és az első paraméter beolvasása

a zárójelek közötti kifejezés beolvasása
zárójelek ellenőrzése
FRMEVL betölti a kifejezést

karakterek vizsgálata BASIC szövegben
")" bezáró zárójel
"(" nyitó zárójel
"," vessző

összehasonlítás az aktuális karakterrel
egyezik?
CHRGET rutin, köv. karakter beolvasása
"syntax error" sorszám
hibaüzenet kírása
a hierarchia kód offset az előjelcseréhez!

a kiértékeléshez

BASIC-en belüli változó kiértékelése

az azonosító /$64/$65/
$A$$$ és a $E32A között van?

ha igen, akkor C=1


```
*****************************
AF28  20 8B B0     JSR   B08B
AF2B  85 64        STA   64
AF2D  84 65        STY   65
AF2F  A6 45        LDX   45
AF31  A4 46        LDY   46
AF33  A5 0D        LDA   0D
AF35  F0 26        BEQ   AF5D
AF37  A9 00        LDA   #00
AF39  85 70        STA   70
AF3B  20 14 AF     JSR   AF14
AF3E  90 1C        BCC   AF5C
AF40  E0 54        CPX   #54
AF42  D0 18        BNE   AF5C
AF44  C0 C9        CPY   #C9
AF46  D0 14        BNE   AF5C
AF48  20 84 AF     JSR   AF84
AF4B  84 5E        STY   5E
AF4D  88           DEY
AF4E  84 71        STY   71
AF50  A0 06        LDY   #06
AF52  84 5D        STY   5D
AF54  A0 24        LDY   #24
AF56  20 68 BE     JSR   BE68
AF59  4C 6F B4     JMP   B46F
AF5C  60           RTS
AF5D  24 0E        BIT   0E
AF5F  10 0D        BPL   AF6E

*****************************
AF61  A0 00        LDY   #00
AF63  B1 64        LDA   (64),Y
AF65  AA           TAX
AF66  C8           INY
AF67  B1 64        LDA   (64),Y
AF69  A8           TAY
AF6A  8A           TXA
AF6B  4C 91 B3     JMP   B391

*****************************
AF6E  20 14 AF     JSR   AF14
AF71  90 2D        BCC   AFA0
AF73  E0 54        CPX   #54
AF75  D0 1B        BNE   AF92
AF77  C0 49        CPY   #49
AF79  D0 25        BNE   AFA0
AF7B  20 84 AF     JSR   AF84
AF7E  98           TYA
AF7F  A2 A0        LDX   #A0
AF81  4C 4F BC     JMP   BC4F

*****************************
AF84  20 DE FF     JSR   FFDE
AF87  86 64        STX   64
AF89  84 63        STY   63
```

a változó betöltése
a változó keresése

a változóra vagy a füzér azonosítójára mutat

a változó neve
tipusa
numerikus?

az azonosító az interpreterben van?
nem
"T"

"I$"

az idő betöltése

a TI$ füzér hossza 6

a TI$ füzért állítja elő

INTEGER/REAL kapcsoló
REAL?

az integer változó beolvasása

integer konstans beolvasása

és átváltása lebegőpontossá

REAL változó beolvasása
az azonosító az interpreterben?
nem
"T"

"T"

a TIME betöltése a FAC-ba

az idő beolvasása
a TIME beolvasása

```
AF8B  85 65       STA  65          és tárolása a lebegőpontos akku-ban
AF8D  A0 00       LDY  #00
AF8F  84 62       STY  62
AF91  60          RTS
AF92  E0 53       CPX  #53         "S"
AF94  D0 0A       BNE  AFA0
AF96  C0 54       CPY  #54         "T"
AF98  D0 06       BNE  AFA0
AF9A  20 B7 FF    JSR  FFB7        a státusz beolvasása
AF9D  4C 3C BC    JMP  BC3C        az akku tartalmának átalakítása lebegőpon-
                                   tos formátumra
**********************************
                                   a REAL változó beolvasása
AFA0  A5 64       LDA  64          a változó címe
AFA2  A4 65       LDY  65
AFA4  4C A2 BB    JMP  BBA2        a változó beolvasása a FAC-be

**********************************
                                   függvényszámítás
AFA7  0A          ASL  A           a függvény kódja   2-szer
AFA8  48          PHA
AFA9  AA          TAX
AFAA  20 73 00    JSR  0073        CHRGET, a következő karakter
AFAD  E0 8F       CPX  #8F
AFAF  90 20       BCC  AFD1        numerikus függvény?

**********************************
                                   szövegfüggvény, szöveg és az első paraméter
AFB1  20 FA AE    JSR  AEFA        a zárójel ellenőrzése
AFB4  20 9E AD    JSR  AD9E        FRMEVL-tetszőleges kifejezés betöltése
AFB7  20 FD AE    JSR  AEFD        a vessző ellenőrzése
AFBA  20 8F AD    JSR  AD8F        a szöveg ellenőrzése
AFBD  68          PLA              left$, right$, mid$ függvények token-je
AFBE  AA          TAX
AFBF  A5 65       LDA  65          a füzér azonosítójának címe
AFC1  48          PHA
AFC2  A5 64       LDA  64
AFC4  48          PHA
AFC5  8A          TXA
AFC6  48          PHA              a token tárolása a veremben
AFC7  20 9E B7    JSR  B79E        a fgv. 2. paraméterének beolvasása
AFCA  68          PLA              a token visszaolvasása
AFCB  A8          TAY
AFCC  8A          TXA
AFCD  48          PHA              a byte a verembe
AFCE  4C D6 AF    JMP  AFD6        a rutin futtatása

**********************************
                                   numerikus függvény kiértékelése
AFD1  20 F1 AE    JSR  AEF1        a zárójeles kifejezés beolvasása
AFD4  68          PLA              a függvény BASIC kódja
AFD5  A8          TAY
AFD6  B9 EA 9F    LDA  9FEA,Y
AFD9  85 55       STA  55          a vektor beállítása
AFDB  B9 EB 9F    LDA  9FEB,Y      a függvény kiértékeléséhez
AFDE  85 56       STA  56
AFE0  20 54 00    JSR  0054        a függvény végrehajtása
AFE3  4C 8D AD    JMP  AD8D        numerikus ellenőrzés

**********************************  az OR BASIC-utasítás
```



```
AFE6  A0 FF       LDY  #FF         az OR-kapcsoló
AFE8  2C          .BYTE 2C

**********************************
AFE9  A0 00       LDY  #00         az AND BASIC-utasítás
AFEB  84 0B       STY  0B          az AND-kapcsoló
AFED  20 BF B1    JSR  B1BF        a kapcsoló beállítása
AFF0  A5 64       LDA  64          a FAC átalakítása INTEGER-ré
AFF2  45 0B       EOR  0B
AFF4  85 07       STA  07
AFF6  A5 65       LDA  65          a flag-gel összekapcsolva a $7/$8-on
AFF8  45 0B       EOR  0B          tárolni!
AFFA  85 08       STA  08
AFFC  20 FC BB    JSR  BBFC        az ARG a FAC-be
AFFF  20 BF B1    JSR  B1BF        a FAC az INTEGER-be
B002  A5 65       LDA  65
B004  45 0B       EOR  0B          logikai összekapcsolás
B006  25 0B       AND  0B
B008  45 0B       EOR  0B
B00A  A8          TAY
B00B  A5 64       LDA  64
B00D  45 0B       EOR  0B
B00F  25 07       AND  07
B011  45 0B       EOR  0B
B013  4C 91 B3    JMP  B391        visszaalakítás lebegőpontossá

**********************************
B016  20 90 AD    JSR  AD90        összehasonlítás
B019  B0 13       BCS  B02E        változó-tipus ellenőrzése
B01B  A5 6E       LDA  6E          füzér? ha igen, akkor tovább
B01D  09 7F       ORA  #7F         ARG tárformátumban
B01F  25 6A       AND  6A
B021  85 6A       STA  6A
B023  A9 69       LDA  #69         az ARG címe
B025  A0 00       LDY  #00
B027  20 5B BC    JSR  BC5B        az ARG összehasonlítása a FAC-cal
B02A  AA          TAX
B02B  4C 61 B0    JMP  B061        az eredmény tárolása a FAC-ben

**********************************
B02E  A9 00       LDA  #00         füzérek összehasonlítása
B030  85 0D       STA  0D          a füzér-kapcsoló törlése
B032  C6 4D       DEC  4D
B034  20 A6 B6    JSR  B6A6        FRESTR
B037  85 61       STA  61          a füzér hossza
B039  86 62       STX  62
B03B  84 63       STY  63          a füzér címe
B03D  A5 6C       LDA  6C
B03F  A4 6D       LDY  6D          a második füzér mutatója
B041  20 AA B6    JSR  B6AA        FRESTR
B044  86 6C       STX  6C
B046  84 6D       STY  6D          a 2. füzér címe
B048  AA          TAX
B049  38          SEC
B04A  E5 61       SBC  61          a hosszak összehasonlítása
B04C  F0 08       BEQ  B056        egyenlő?
B04E  A9 01       LDA  #01
B050  90 04       BCC  B056        a 2. füzér rövidebb
```

```
B052  A6 61       LDX   61
B054  A9 FF       LDA  #FF
B056  85 66       STA   66
B058  A0 FF       LDY  #FF
B05A  E8          INX
B05B  C8          INY
B05C  CA          DEX
B05D  D0 07       BNE   B066
B05F  A6 66       LDX   66
B061  30 0F       BMI   B072
B063  18          CLC
B064  90 0C       BCC   B072
B066  B1 6C       LDA   (6C),Y
B068  D1 62       CMP   (62),Y
B06A  F0 EF       BEQ   B05B
B06C  A2 FF       LDX  #FF
B06E  B0 02       BCS   B072
B070  A2 01       LDX  #01
B072  E8          INX
B073  8A          TXA
B074  2A          ROL   A
B075  25 12       AND   12
B077  F0 02       BEQ   B07B
B079  A9 FF       LDA  #FF
B07B  4C 3C BC    JMP   BC3C
B07E  20 FD AE    JSR   AEFD

*****************************
B081  AA          TAX
B082  20 90 B0    JSR   B090
B085  20 79 00    JSR   0079
B088  D0 F4       BNE   B07E
B08A  60          RTS

*****************************
B08B  A2 00       LDX  #00
B08D  20 79 00    JSR   0079
B090  86 0C       STX   0C
B092  85 45       STA   45
B094  20 79 00    JSR   0079
B097  20 13 B1    JSR   B113
B09A  B0 03       BCS   B09F
B09C  4C 08 AF    JMP   AF08
B09F  A2 00       LDX  #00
B0A1  86 0D       STX   0D
B0A3  86 0E       STX   0E
B0A5  20 73 00    JSR   0073
B0A8  90 05       BCC   B0AF
B0AA  20 13 B1    JSR   B113
B0AD  90 0B       BCC   B0BA
B0AF  AA          TAX
B0B0  20 73 00    JSR   0073
B0B3  90 FB       BCC   B0B0
B0B5  20 13 B1    JSR   B113
B0B8  B0 F6       BCS   B0B0
B0BA  C9 24       CMP  #24
```

1. füzérnél

a füzérek összehasonlítása
jelenként

eredmény beolvasása a FAC-be
a CHRCOM vesszőt keres

DIM BASIC-utasítás

a változó dimenzionálása
CHRGOT, az utolsó karakter beolvasása
ha nincs vége, vesszük a köv. változót

a változó beolvasása
a kapcsoló:"nem dimenzionált"-ra
CHRGOT, az utolsó karakter beolvasása
a DIM-kapcsoló beállítása
a változó neve
CHRGOT rutin, az utolsó karakter beolv.
betű keresése
igen
"syntax error"

a füzér-kapcsoló törlése
az integer-kapcsoló törlése
CHRGET, köv. karakter beolvasása
számjegy?
betű keresése
nem
a név második betűje
CHRGET, következő karakter beolvasása
számjegy?
betű keresése
ha betű, a további karakterek átolvasása
"$"


```
B0BC  D0 06       BNE   B0C4
B0BE  A9 FF       LDA  #FF
B0C0  85 0D       STA   0D
B0C2  D0 10       BNE   B0D4
B0C4  C9 25       CMP  #25
B0C6  D0 13       BNE   B0DB
B0C8  A5 10       LDA   10
B0CA  D0 D0       BNE   B09C
B0CC  A9 80       LDA  #80
B0CE  85 0E       STA   0E
B0D0  05 45       ORA   45
B0D2  85 45       STA   45
B0D4  8A          TXA
B0D5  09 80       ORA  #80
B0D7  AA          TAX
B0D8  20 73 00    JSR   0073
B0DB  86 46       STX   46
B0DD  38          SEC
B0DE  05 10       ORA   10
B0E0  E9 28       SBC  #28
B0E2  D0 03       BNE   B0E7
B0E4  4C D1 B1    JMP   B1D1
B0E7  A0 00       LDY  #00
B0E9  84 10       STY   10
B0EB  A5 2D       LDA   2D
B0ED  A6 2E       LDX   2E
B0EF  86 60       STX   60
B0F1  85 5F       STA   5F
B0F3  E4 30       CPX   30
B0F5  D0 04       BNE   B0FB
B0F7  C5 2F       CMP   2F
B0F9  F0 22       BEQ   B11D
B0FB  A5 45       LDA   45
B0FD  D1 5F       CMP   (5F),Y
B0FF  D0 08       BNE   B109
B101  A5 46       LDA   46
B103  C8          INY
B104  D1 5F       CMP   (5F),Y
B106  F0 7D       BEQ   B185
B108  88          DEY
B109  18          CLC
B10A  A5 5F       LDA   5F
B10C  69 07       ADC  #07
B10E  90 E1       BCC   B0F1
B110  E8          INX
B111  D0 DC       BNE   B0EF

*****************************
B113  C9 41       CMP  #41
B115  90 05       BCC   B11C
B117  E9 5B       SBC  #5B
B119  38          SEC
B11A  E9 A5       SBC  #A5
B11C  60          RTS

*****************************
```

nem

a füzér-kapcsoló beállítása
ugrás
"%"
nem
megengedett az integer?
nem, "syntax error"

integer-kapcsoló beállítása
a 7. bit beállítása a névben

a név második betűje

CHRGET, a következő karakter beolvasása
a második betű tárolása

"("
nem nyitózárójel?
dimenzionált változó beolvasása
00 45 VÉV KI A/Y CÍM
02 46
változó-kezdet mutató

tárolása kereséshez

vége a változóterületnek?
nem sikerült megtalálni, újra beolvasni
a név első betűjét
keresés a táblázatban
ha nem egyenlő, tovább keresni
a második betű

összehasonlítása

a mutató növelése 7-tel /2+5 byte REAL vált/

továbbkeresés

az "A" betű keresése

"Z"+1
ha igen, akkor C=1
ha nem, akkor C=0

a változó tárolása

```
B11D  68          PLA
B11E  48          PHA
B11F  C9 2A       CMP  #2A
B121  D0 05       BNE  B128
B123  A9 13       LDA  #13
B125  A0 BF       LDY  #BF
B127  60          RTS
B128  A5 45       LDA  45
B12A  A4 46       LDY  46
B12C  C9 54       CMP  #54
B12E  D0 0B       BNE  B13B
B130  C0 C9       CPY  #C9
B132  F0 EF       BEQ  B123
B134  C0 49       CPY  #49
B136  D0 03       BNE  B13B
B138  4C 08 AF    JMP  AF08
B13B  C9 53       CMP  #53
B13D  D0 04       BNE  B143
B13F  C0 54       CPY  #54
B141  F0 F5       BEQ  B138
B143  A5 2F       LDA  2F
B145  A4 30       LDY  30
B147  85 5F       STA  5F
B149  84 60       STY  60
B14B  A5 31       LDA  31
B14D  A4 32       LDY  32
B14F  85 5A       STA  5A
B151  84 5B       STY  5B
B153  18          CLC
B154  69 07       ADC  #07
B156  90 01       BCC  B159
B158  C8          INY
B159  85 58       STA  58
B15B  84 59       STY  59
B15D  20 B8 A3    JSR  A3B8
B160  A5 58       LDA  58
B162  A4 59       LDY  59
B164  C8          INY
B165  85 2F       STA  2F
B167  84 30       STY  30
B169  A0 00       LDY  #00
B16B  A5 45       LDA  45
B16D  91 5F       STA  (5F),Y
B16F  C8          INY
B170  A5 46       LDA  46
B172  91 5F       STA  (5F),Y
B174  A9 00       LDA  #00
B176  C8          INY
B177  91 5F       STA  (5F),Y
B179  C8          INY
B17A  91 5F       STA  (5F),Y
B17C  C8          INY
B17D  91 5F       STA  (5F),Y
B17F  C8          INY
B180  91 5F       STA  (5F),Y
B182  C8          INY
```

hívási cím ellenőrzése
FRMEVL hívása?
ha nem, újra tárolni
a mutató a Ø konstansra!

a változó neve
"T"

"I$"
igen, TI$
"I"
nem
"syntax error"
"S"

"T"
ha ST, akkor "syntax error"

tömb-táblázat mutatójának

tárolása

tömb-vázlat vége - mutató

tárolása

7-tel eltolni egy új elem tárolásához

új blokkvégződés
a blokk eltolása

a tömb-táblázat mutatójának újrabeállítása

a név első betűje

és második betűjének tárolása

5 nullabyte a változóba


```
B183  91 5F       STA  (5F),Y
B185  A5 5F       LDA  5F
B187  18          CLC
B188  69 02       ADC  #02
B18A  A4 60       LDY  60
B18C  90 01       BCC  B18F
B18E  C8          INY
B18F  85 47       STA  47
B191  84 48       STY  48
B193  60          RTS

*****************************
B194  A5 0B       LDA  0B
B196  0A          ASL  A
B197  69 05       ADC  #05
B199  65 5F       ADC  5F
B19B  A4 60       LDY  60
B19D  90 01       BCC  B1A0
B19F  C8          INY
B1A0  85 58       STA  58
B1A2  84 59       STY  59
B1A4  60          RTS

*****************************
B1A5  90 80 00 00 00

*****************************
B1AA  20 BF B1    JSR  B1BF
B1AD  A5 64       LDA  64
B1AF  A4 65       LDY  65
B1B1  60          RTS

*****************************
B1B2  20 73 00    JSR  0073
B1B5  20 9E AD    JSR  AD9E
B1B8  20 8D AD    JSR  AD8D
B1BB  A5 66       LDA  66
B1BD  30 0D       BMI  B1CC
B1BF  A5 61       LDA  61
B1C1  C9 90       CMP  #90
B1C3  90 09       BCC  B1CE
B1C5  A9 A5       LDA  #A5
B1C7  A0 B1       LDY  #B1
B1C9  20 5B BC    JSR  BC5B
B1CC  D0 7A       BNE  B248
B1CE  4C 9B BC    JMP  BC9B

*****************************
B1D1  A5 0C       LDA  0C
B1D3  05 0E       ORA  0E
B1D5  48          PHA
B1D6  A5 0D       LDA  0D
B1D8  48          PHA
B1D9  A0 00       LDY  #00
B1DB  98          TYA
B1DC  48          PHA
```

2 hozzáadása a névhez
a változó mutatója

$47/$48 szerint

az első tömbelem mutatójának kiszámítása
a dimenziók száma
2-szer
plusz 5

$5F/$6Ø-hoz hozzáadni

az új mutató

-32768 konstans

FAC/INTEGER konvertálás
FAC/INTEGER konvertálás
alsó byte
felső byte

a kifejezés beolvasása az integerbe
CHRGET, a köv. karakter beolvasása /integer/
FRMEVL, a kifejezés kiértékelése
numerikus ellenőrzés
előjel?
ha negatív, akkor "illegal quantity"
hatványkitevő
az eredmény nagyobb 32768-nál?
nem

a mutató a -32768 konstansra
FAC összehasonlítása a konstanssal
ha nem egyenlő, "illegal quantity"
lebegőpontos/egész konvertálás

a deklarált változó beolvasása
a DIM-kapcsoló
az integer-kapcsoló

a füzér-kapcsoló

az indexek száma

```
B1DD  A5 46      LDA  46        a változónév 2. betűje
B1DF  48         PHA
B1E0  A5 45      LDA  45        a változónév első betűje
B1E2  48         PHA
B1E3  20 B2 B1   JSR  B1B2      az index behozatala integer-be
B1E6  68         PLA
B1E7  85 45      STA  45
B1E9  68         PLA            a változónév visszaolvasása
B1EA  85 46      STA  46
B1EC  68         PLA            az indexek száma
B1ED  A8         TAY
B1EE  BA         TSX
B1EF  BD 02 01   LDA  0102,X
B1F2  48         PHA            a kapcsolók beolvasása a veremből
B1F3  BD 01 01   LDA  0101,X
B1F6  48         PHA
B1F7  A5 64      LDA  64
B1F9  9D 02 01   STA  0102,X
B1FC  A5 65      LDA  65        az alsó és felső byte a verembe
B1FE  9D 01 01   STA  0101,X
B201  C8         INY            az indexek számának növelése
B202  20 79 00   JSR  0079      CHRGOT, az utolsó karakter beolvasása
B205  C9 2C      CMP  #2C       "," vessző?
B207  F0 D2      BEQ  B1DB      ha igen, akkor a következő index
B209  84 0B      STY  0B        az indexek számának tárolása
B20B  20 F7 AE   JSR  AEF7      bezáró zárójel ellenőrzése
B20E  68         PLA
B20F  85 0D      STA  0D
B211  68         PLA            a kapcsolók visszaolvasása
B212  85 0E      STA  0E
B214  29 7F      AND  #7F
B216  85 0C      STA  0C
B218  A6 2F      LDX  2F
B21A  A5 30      LDA  30        a tömb-táblázat mutatója
B21C  86 5F      STX  5F
B21E  85 60      STA  60        a mutató tárolása
B220  C5 32      CMP  32
B222  D0 04      BNE  B228
B224  E4 31      CPX  31        vége a táblázatnak?
B226  F0 39      BEQ  B261      ha igen, nem sikerült megtalálni
B228  A0 00      LDY  #00
B22A  B1 5F      LDA  (5F),Y    a táblázatból származó név
B22C  C8         INY
B22D  C5 45      CMP  45        összehasonlítása a keresett névvel
B22F  D0 06      BNE  B237
B231  A5 46      LDA  46
B233  D1 5F      CMP  (5F),Y    második betűt
B235  F0 16      BEQ  B24D      sikerült megtalálni
B237  C8         INY
B238  B1 5F      LDA  (5F),Y
B23A  18         CLC            mezőhossz hozzáadása
B23B  65 5F      ADC  5F
B23D  AA         TAX
B23E  C8         INY
B23F  B1 5F      LDA  (5F),Y
B241  65 60      ADC  60
```



```
B243  90 D7      BCC  B21C      és továbbkeresés
B245  A2 12      LDX  #12       "bad subscript" sorszám
B247  2C         .BYTE 2C

B248  A2 0E      LDX  #0E       "illegal quantity" sorszám
B24A  4C 37 A4   JMP  A437      hibaüzenet kiírása
B24D  A2 13      LDX  #13       "redim'd array" sorszám
B24F  A5 0C      LDA  0C        a DIM-kapcsoló nulla?
B251  D0 F7      BNE  B24A      ha nem, akkor hibaüzenet
B253  20 94 B1   JSR  B194      a mutató az első tömb-elemre
B256  A5 0B      LDA  0B        a megtalált dimenziók számának
B258  A0 04      LDY  #04
B25A  D1 5F      CMP  (5F),Y    összehasonlítása a tömb-dimenziókkal
B25C  D0 E7      BNE  B245      egyenlőtlenség esetén "bad subscript"
B25E  4C EA B2   JMP  B2EA      a kívánt tömb-elemek keresése

**********************************
B261  20 94 B1   JSR  B194      a tömbváltozó tárolása
B264  20 08 A4   JSR  A408      a tömbfejléc hossza
B267  A0 00      LDY  #00       van elegendő hely a memóriában?
B269  84 72      STY  72
B26B  A2 05      LDX  #05       a /REAL/ változó hossza, alapértelmezés
B26D  A5 45      LDA  45        a név első betűje
B26F  91 5F      STA  (5F),Y    a tömbtáblázatban
B271  10 01      BPL  B274      nem integer?
B273  CA         DEX
B274  C8         INY
B275  A5 46      LDA  46        a második betű
B277  91 5F      STA  (5F),Y    beírása a táblázatba
B279  10 02      BPL  B27D      nem füzér vagy egész?
B27B  CA         DEX
B27C  CA         DEX
B27D  86 71      STX  71        a végleges hossz 2, 3 vagy 5
B27F  A5 0B      LDA  0B        a dimenziók száma
B281  C8         INY
B282  C8         INY
B283  C8         INY
B284  91 5F      STA  (5F),Y    tárolás
B286  A2 0B      LDX  #0B       11, az alapértelmezés
B288  A9 00      LDA  #00
B28A  24 0C      BIT  0C        a rutint a DIM-ből hívták?
B28C  50 08      BVC  B296      nem
B28E  68         PLA            dimenzió beolvasása a veremből
B28F  18         CLC
B290  69 01      ADC  #01       egy hozzáadása
B292  AA         TAX
B293  68         PLA
B294  69 00      ADC  #00
B296  C8         INY
B297  91 5F      STA  (5F),Y    és tárolása
B299  C8         INY
B29A  8A         TXA
B29B  91 5F      STA  (5F),Y
B29D  20 4C B3   JSR  B34C      további dimenziókhoz szüks. hely kiszámítása
B2A0  86 71      STX  71
B2A2  85 72      STA  72        változóvég mutató tárolása
B2A4  A4 22      LDY  22
B2A6  C6 0B      DEC  0B        további dimenziók?
B2A8  D0 DC      BNE  B286      igen
B2AA  65 59      ADC  59
B2AC  B0 5D      BCS  B30B      a mezőhossz plusz a kezdőcím
```

```
B2AE  85 59      STA  59
B2B0  A8         TAY
B2B1  8A         TXA
B2B2  65 58      ADC  58
B2B4  90 03      BCC  B2B9
B2B6  C8         INY
B2B7  F0 52      BEQ  B30B
B2B9  20 08 A4   JSR  A408      elegendő tárterület?
B2BC  85 31      STA  31        mutató a tömbtáblázat végére, a tömb
B2BE  84 32      STY  32        a tömbtáblázat vége, mutatótömb feltölt.
B2C0  A9 00      LDA  #00       nullával
B2C2  E6 72      INC  72        nullázása
B2C4  A4 71      LDY  71
B2C6  F0 05      BEQ  B2CD
B2C8  88         DEY
B2C9  91 58      STA  (58),Y
B2CB  D0 FB      BNE  B2C8
B2CD  C6 59      DEC  59
B2CF  C6 72      DEC  72
B2D1  D0 F5      BNE  B2C8
B2D3  E6 59      INC  59
B2D5  38         SEC
B2D6  A5 31      LDA  31
B2D8  E5 5F      SBC  5F
B2DA  A0 02      LDY  #02
B2DC  91 5F      STA  (5F),Y    a tömbhossz alsó byte
B2DE  A5 32      LDA  32
B2E0  C8         INY
B2E1  E5 60      SBC  60
B2E3  91 5F      STA  (5F),Y    a tömbhossz felső byte
B2E5  A5 0C      LDA  0C        a DIM-utasítás hívása?
B2E7  D0 62      BNE  B34B      igen, RTS

****************************   tömbelem keresése
B2E9  C8         INY
B2EA  B1 5F      LDA  (5F),Y    dimenziók számának
B2EC  85 0B      STA  0B        tárolása
B2EE  A9 00      LDA  #00
B2F0  85 71      STA  71
B2F2  85 72      STA  72
B2F4  C8         INY
B2F5  68         PLA
B2F6  AA         TAX
B2F7  85 64      STA  64        az index beolvasása a veremből
B2F9  68         PLA
B2FA  85 65      STA  65
B2FC  D1 5F      CMP  (5F),Y    összehasonlítás
B2FE  90 0E      BCC  B30E      kisebb?
B300  D0 06      BNE  B308      ha nagyobb, akkor "bad subscript"
B302  C8         INY
B303  8A         TXA
B304  D1 5F      CMP  (5F),Y    egyenlőség esetén az alsó byte összehas.
B306  90 07      BCC  B30F      ha kisebb, akkor tovább
B308  4C 45 B2   JMP  B245      "bad subscript"
B30B  4C 35 A4   JMP  A435      "out of memory"

****************************   egy tömbelem címének kiszámítása
```



```
B30E  C8         INY
B30F  A5 72      LDA  72
B311  05 71      ORA  71
B313  18         CLC
B314  F0 0A      BEQ  B320
B316  20 4C B3   JSR  B34C      szorzás
B319  8A         TXA
B31A  65 64      ADC  64
B31C  AA         TAX
B31D  98         TYA
B31E  A4 22      LDY  22
B320  65 65      ADC  65
B322  86 71      STX  71
B324  C6 0B      DEC  0B        a dimenziók száma
B326  D0 CA      BNE  B2F2
B328  85 72      STA  72
B32A  A2 05      LDX  #05       változóhossz, alapértelmezés /5,REAL/
B32C  A5 45      LDA  45        a név első betűje
B32E  10 01      BPL  B331
B330  CA         DEX
B331  A5 46      LDA  46        a név második betűje
B333  10 02      BPL  B337
B335  CA         DEX
B336  CA         DEX
B337  86 28      STX  28        változó hossza 2, 3 vagy 5
B339  A9 00      LDA  #00
B33B  20 55 B3   JSR  B355      az offset kiszámítása a tömbben
B33E  8A         TXA
B33F  65 58      ADC  58
B341  85 47      STA  47
B343  98         TYA
B344  65 59      ADC  59
B346  85 48      STA  48
B348  A8         TAY
B349  A5 47      LDA  47
B34B  60         RTS

****************************   a számítás segédrutinja
B34C  84 22      STY  22
B34E  B1 5F      LDA  (5F),Y
B350  85 28      STA  28
B352  88         DEY
B353  B1 5F      LDA  (5F),Y
B355  85 29      STA  29
B357  A9 10      LDA  #10
B359  85 5D      STA  5D
B35B  A2 00      LDX  #00
B35D  A0 00      LDY  #00
B35F  8A         TXA
B360  0A         ASL  A
B361  AA         TAX
B362  98         TYA
B363  2A         ROL  A
B364  A8         TAY
B365  B0 A4      BCS  B30B      "out of memory"
B367  06 71      ASL  71
```

```
B369  26 72        ROL   72
B36B  90 0B        BCC   B378
B36D  18           CLC
B36E  8A           TXA
B36F  65 28        ADC   28
B371  AA           TAX
B372  98           TYA
B373  65 29        ADC   29
B375  A8           TAY
B376  B0 93        BCS   B30B        "out of memory"
B378  C6 5D        DEC   5D
B37A  D0 E3        BNE   B35F
B37C  60           RTS

*****************************
B37D  A5 0D        LDA   0D          FRE BASIC-funkció
B37F  F0 03        BEQ   B384        tipus-kapcsoló
B381  20 A6 B6     JSR   B6A6        nem füzér
B384  20 26 B5     JSR   B526        FRESTR
B387  38           SEC               Garbage Collection
B388  A5 33        LDA   33
B38A  E5 31        SBC   31          a füzér-kezdet
B38C  A8           TAY
B38D  A5 34        LDA   34
B38F  E5 32        SBC   32          minusz a változó-vég
B391  A2 00        LDX   #00
B393  86 0D        STX   0D          a kapcsoló beállítása /numerikus/
B395  85 62        STA   62
B397  84 63        STY   63          az eredmény tárolása
B399  A2 90        LDX   #90
B39B  4C 44 BC     JMP   BC44        és konvertálás lebegőpontossá

*****************************
B39E  38           SEC               a POS BASIC-függvény
B39F  20 F0 FF     JSR   FFF0        C=1 a kurzorpozició beolvasása
B3A2  A9 00        LDA   #00         a kurzorpozició beolvasása
B3A4  F0 EB        BEQ   B391        folytatás a fentiek szerint

*****************************
B3A6  A6 3A        LDX   3A          parancsmód - teszt
B3A8  E8           INX
B3A9  D0 A0        BNE   B34B        ha nem, akkor RTS
B3AB  A2 15        LDX   #15         "illegal direct" kódja
B3AD  2C A2 1B     BIT   1BA2        "undef'd function" kódja
B3B0  4C 37 A4     JMP   A437        a hibaüzenet kiírása

*****************************
B3B3  20 E1 B3     JSR   B3E1        a DEF FN BASIC-utasítás
B3B6  20 A6 B3     JSR   B3A6        az FN szintaxis ellenőrzése
B3B9  20 FA AE     JSR   AEFA        parancsmód?
B3BC  A9 80        LDA   #80         a nyitó zárójel ellenőrzése
B3BE  85 10        STA   10          az INTEGER tárolása
B3C0  20 8B B0     JSR   B08B        a változó keresése
B3C3  20 8D AD     JSR   AD8D        numerikus ellenőrzés
B3C6  20 F7 AE     JSR   AEF7        a bezáró zárójel ellenőrzése
B3C9  A9 B2        LDA   #B2         "=" BASIC-kód
```



```
B3CB  20 FF AE     JSR   AEFF        "=" ellenőrzés
B3CE  48           PHA
B3CF  A5 48        LDA   48          az FN-változó címe a verembe
B3D1  48           PHA
B3D2  A5 47        LDA   47
B3D4  48           PHA
B3D5  A5 7B        LDA   7B          a programmutató a verembe
B3D7  48           PHA
B3D8  A5 7A        LDA   7A
B3DA  48           PHA
B3DB  20 F8 A8     JSR   A8F8        a programmutató a következő utasításra
B3DE  4C 4F B4     JMP   B44F        az FN-változó beolvasása a veremből

*****************************
B3E1  A9 A5        LDA   #A5         az FN szintaxis ellenőrzése
B3E3  20 FF AE     JSR   AEFF        FN-kód
B3E6  09 80        ORA   #80         az FN-kód ellenőrzése
B3E8  85 10        STA   10          az INTEGER változó tárolása
B3EA  20 92 B0     JSR   B092        a változó keresése
B3ED  85 4E        STA   4E
B3EF  84 4F        STY   4F          FN-változó mutató beállítása
B3F1  4C 8D AD     JMP   AD8D        numerikus ellenőrzés

*****************************
B3F4  20 E1 B3     JSR   B3E1        az FN BASIC-függvény
B3F7  A5 4F        LDA   4F          az FN-szintaxis ellenőrzése
B3F9  48           PHA
B3FA  A5 4E        LDA   4E          FN-változó mutató a verembe
B3FC  48           PHA
B3FD  20 F1 AE     JSR   AEF1        a zárójeles kifejezés beolvasása
B400  20 8D AD     JSR   AD8D        numerikus ellenőrzés
B403  68           PLA
B404  85 4E        STA   4E
B406  68           PLA               FN-változó mutató visszatöltése
B407  85 4F        STA   4F
B409  A0 02        LDY   #02
B40B  B1 4E        LDA   (4E),Y
B40D  85 47        STA   47
B40F  AA           TAX
B410  C8           INY
B411  B1 4E        LDA   (4E),Y
B413  F0 99        BEQ   B3AE        "undef'd function" hiba
B415  85 48        STA   48
B417  C8           INY
B418  B1 47        LDA   (47),Y
B41A  48           PHA
B41B  88           DEY
B41C  10 FA        BPL   B418
B41E  A4 48        LDY   48
B420  20 D4 BB     JSR   BBD4        a FAC átvitele az FN-változóba
B423  A5 7B        LDA   7B
B425  48           PHA
B426  A5 7A        LDA   7A
B428  48           PHA
B429  B1 4E        LDA   (4E),Y
B42B  85 7A        STA   7A
```

```
B42D  C8           INY
B42E  B1 4E        LDA  (4E),Y
B430  85 7B        STA  7B
B432  A5 48        LDA  48
B434  48           PHA
B435  A5 47        LDA  47
B437  48           PHA
B438  20 8A AD     JSR  AD8A
B43B  68           PLA
B43C  85 4E        STA  4E
B43E  68           PLA
B43F  85 4F        STA  4F
B441  20 79 00     JSR  0079
B444  F0 03        BEQ  B449
B446  4C 08 AF     JMP  AF08
B449  68           PLA
B44A  85 7A        STA  7A
B44C  68           PLA
B44D  85 7B        STA  7B
B44F  A0 00        LDY  #00
B451  68           PLA
B452  91 4E        STA  (4E),Y
B454  68           PLA
B455  C8           INY
B456  91 4E        STA  (4E),Y
B458  68           PLA
B459  C8           INY
B45A  91 4E        STA  (4E),Y
B45C  68           PLA
B45D  C8           INY
B45E  91 4E        STA  (4E),Y
B460  68           PLA
B461  C8           INY
B462  91 4E        STA  (4E),Y
B464  60           RTS

*****************************
B465  20 8D AD     JSR  AD8D
B468  A0 00        LDY  #00
B46A  20 DF BD     JSR  BDDF
B46D  68           PLA
B46E  68           PLA
B46F  A9 FF        LDA  #FF
B471  A0 00        LDY  #00
B473  F0 12        BEQ  B487

*****************************
B475  A6 64        LDX  64
B477  A4 65        LDY  65
B479  86 50        STX  50
B47B  84 51        STY  51
B47D  20 F4 B4     JSR  B4F4
B480  86 62        STX  62
B482  84 63        STY  63
B484  85 61        STA  61
B486  60           RTS
```

a programmutató az FN kifejezésre

FRMNUM, numerikus kifejezés beolvasása

a CHRGOT, az utolsó karakter beolvasása
nincs további karakter?
"syntax error" hiba

a programmutató

és az FN-változó beolvasása a veremből

az STR$ BASIC-függvény
numerikus ellenőrzés

FAC/ASCII konvertálás

a füzérek kezdőcíme = $FF

a füzér-mutató kiszámítása
az akku tartalmazza a füzér hosszát

a füzér azonosítójának mutatója

az új füzér tárolása, hossz "A"-ban
alsó cím
felső cím
a hossz


```
*****************************
B487  A2 22        LDX  #22
B489  86 07        STX  07
B48B  86 08        STX  08
B48D  85 6F        STA  6F
B48F  84 70        STY  70
B491  85 62        STA  62
B493  84 63        STY  63
B495  A0 FF        LDY  #FF
B497  C8           INY
B498  B1 6F        LDA  (6F),Y
B49A  F0 0C        BEQ  B4A8
B49C  C5 07        CMP  07
B49E  F0 04        BEQ  B4A4
B4A0  C5 08        CMP  08
B4A2  D0 F3        BNE  B497
B4A4  C9 22        CMP  #22
B4A6  F0 01        BEQ  B4A9
B4A8  18           CLC
B4A9  84 61        STY  61
B4AB  98           TYA
B4AC  65 6F        ADC  6F
B4AE  85 71        STA  71
B4B0  A6 70        LDX  70
B4B2  90 01        BCC  B4B5
B4B4  E8           INX
B4B5  86 72        STX  72
B4B7  A5 70        LDA  70
B4B9  F0 04        BEQ  B4BF
B4BB  C9 02        CMP  #02
B4BD  D0 0B        BNE  B4CA
B4BF  98           TYA
B4C0  20 75 B4     JSR  B475
B4C3  A6 6F        LDX  6F
B4C5  A4 70        LDY  70
B4C7  20 88 B6     JSR  B688

*****************************
B4CA  A6 16        LDX  16
B4CC  E0 22        CPX  #22
B4CE  D0 05        BNE  B4D5
B4D0  A2 19        LDX  #19
B4D2  4C 37 A4     JMP  A437
B4D5  A5 61        LDA  61
B4D7  95 00        STA  00,X
B4D9  A5 62        LDA  62
B4DB  95 01        STA  01,X
B4DD  A5 63        LDA  63
B4DF  95 02        STA  02,X
B4E1  A0 00        LDY  #00
B4E3  86 64        STX  64
B4E5  84 65        STY  65
B4E7  84 70        STY  70
B4E9  88           DEY
B4EA  84 0D        STY  0D
```

a füzér beolvasása,a mutató az A/Y-ban
"(")"

a füzér kezdőcíme

a mutató növelése
a füzér következő karaktere
végjel?

")"

a füzér hossza

végcím alsó + 1

végcím felső +'1
kezdőcím felső
nulla?
kettő?
nem
a hossz az akku-ban
a füzér-mutató kiszámítása

a kezdőcím beolvasása
a füzér bemásolása a füzérterületre.

a füzér-mutató bevitele a füzérleíró verembe
a füzér-azonosító mutatója
megtelt a füzér-verem?
nem
"formula too complex" sorszám
a hibaüzenet kiírása

a füzér hossza

és a cím

tárolása a füzér-verembe

a mutató az azonosítóra

a füzér-kapcsoló beállítása - $FF

```
B4EC  86 17     STX  17        az utolsó füzér-azonosító indexét
B4EE  E8        INX
B4EF  E8        INX            3-mal növelni
B4F0  E8        INX
B4F1  86 16     STX  16        új indexként tárolni!
B4F3  60        RTS

*****************************
B4F4  46 0F     LSR  0F        helyfoglalás a füzéreknek, hossz az A-ban
B4F6  48        PHA            garbage collect-kapcsoló visszaállítása
B4F7  49 FF     EOR  #FF       a füzér hossza
B4F9  38        SEC
B4FA  65 33     ADC  33
B4FC  A4 34     LDY  34
B4FE  B0 01     BCS  B501
B500  88        DEY
B501  C4 32     CPY  32
B503  90 11     BCC  B516      ha nincs elég hely, Garbage Collect
B505  D0 04     BNE  B50B
B507  C5 31     CMP  31
B509  90 0B     BCC  B516
B50B  85 33     STA  33
B50D  84 34     STY  34
B50F  85 35     STA  35
B511  84 36     STY  36
B513  AA        TAX
B514  68        PLA
B515  60        RTS            a füzér-hossz visszaolvasása

B516  A2 10     LDX  #10       "out of memory" sorszáma
B518  A5 0F     LDA  0F        garbage collect -kapcsoló
B51A  30 B6     BMI  B4D2      ha kész, akkor "out of memory"
B51C  20 26 B5  JSR  B526      garbage collect
B51F  A9 80     LDA  #80       kapcsoló beállítása
B521  85 0F     STA  0F
B523  68        PLA            a füzér hossza
B524  D0 D0     BNE  B4F6

*****************************
B526  A6 37     LDX  37        garbage collection
B528  A5 38     LDA  38        érvénytelen füzérek megszüntetése
B52A  86 33     STX  33
B52C  85 34     STA  34
B52E  A0 00     LDY  #00
B530  84 4F     STY  4F
B532  84 4E     STY  4E
B534  A5 31     LDA  31
B536  A6 32     LDX  32
B538  85 5F     STA  5F
B53A  86 60     STX  60
B53C  A9 19     LDA  #19
B53E  A2 00     LDX  #00
B540  85 22     STA  22
B542  86 23     STX  23
B544  C5 16     CMP  16
B546  F0 05     BEQ  B54D
B548  20 C7 B5  JSR  B5C7
```



```
B54B  F0 F7     BEQ  B544
B54D  A9 07     LDA  #07
B54F  85 53     STA  53
B551  A5 2D     LDA  2D
B553  A6 2E     LDX  2E
B555  85 22     STA  22
B557  86 23     STX  23
B559  E4 30     CPX  30
B55B  D0 04     BNE  B561
B55D  C5 2F     CMP  2F
B55F  F0 05     BEQ  B566
B561  20 BD B5  JSR  B5BD
B564  F0 F3     BEQ  B559
B566  85 58     STA  58
B568  86 59     STX  59
B56A  A9 03     LDA  #03
B56C  85 53     STA  53
B56E  A5 58     LDA  58
B570  A6 59     LDX  59
B572  E4 32     CPX  32
B574  D0 07     BNE  B57D
B576  C5 31     CMP  31
B578  D0 03     BNE  B57D
B57A  4C 06 B6  JMP  B606
B57D  85 22     STA  22
B57F  86 23     STX  23
B581  A0 00     LDY  #00
B583  B1 22     LDA  (22),Y
B585  AA        TAX
B586  C8        INY
B587  B1 22     LDA  (22),Y
B589  08        PHP
B58A  C8        INY
B58B  B1 22     LDA  (22),Y
B58D  65 58     ADC  58
B58F  85 58     STA  58
B591  C8        INY
B592  B1 22     LDA  (22),Y
B594  65 59     ADC  59
B596  85 59     STA  59
B598  28        PLP
B599  10 D3     BPL  B56E
B59B  8A        TXA
B59C  30 D0     BMI  B56E
B59E  C8        INY
B59F  B1 22     LDA  (22),Y
B5A1  A0 00     LDY  #00
B5A3  0A        ASL  A
B5A4  69 05     ADC  #05
B5A6  65 22     ADC  22
B5A8  85 22     STA  22
B5AA  90 02     BCC  B5AE
B5AC  E6 23     INC  23
B5AE  A6 23     LDX  23
B5B0  E4 59     CPX  59
B5B2  D0 04     BNE  B5B8
```

```
B5B4  C5 58       CMP   58
B5B6  F0 BA       BEQ   B572
B5B8  20 C7 B5    JSR   B5C7
B5BB  F0 F3       BEQ   B5B0

*****************************
B5BD  B1 22       LDA   (22),Y
B5BF  30 35       BMI   B5F6
B5C1  C8          INY
B5C2  B1 22       LDA   (22),Y
B5C4  10 30       BPL   B5F6
B5C6  C8          INY
B5C7  B1 22       LDA   (22),Y
B5C9  F0 2B       BEQ   B5F6
B5CB  C8          INY
B5CC  B1 22       LDA   (22),Y
B5CE  AA          TAX
B5CF  C8          INY
B5D0  B1 22       LDA   (22),Y
B5D2  C5 34       CMP   34
B5D4  90 06       BCC   B5DC
B5D6  D0 1E       BNE   B5F6
B5D8  E4 33       CPX   33
B5DA  B0 1A       BCS   B5F6
B5DC  C5 60       CMP   60
B5DE  90 16       BCC   B5F6
B5E0  D0 04       BNE   B5E6
B5E2  E4 5F       CPX   5F
B5E4  90 10       BCC   B5F6
B5E6  86 5F       STX   5F
B5E8  85 60       STA   60
B5EA  A5 22       LDA   22
B5EC  A6 23       LDX   23
B5EE  85 4E       STA   4E
B5F0  86 4F       STX   4F
B5F2  A5 53       LDA   53
B5F4  85 55       STA   55
B5F6  A5 53       LDA   53
B5F8  18          CLC
B5F9  65 22       ADC   22
B5FB  85 22       STA   22
B5FD  90 02       BCC   B601
B5FF  E6 23       INC   23
B601  A6 23       LDX   23
B603  A0 00       LDY   #00
B605  60          RTS

*****************************
B606  A5 4F       LDA   4F
B608  05 4E       ORA   4E
B60A  F0 F5       BEQ   B601
B60C  A5 55       LDA   55
B60E  29 04       AND   #04
B610  4A          LSR   A
B611  A8          TAY
B612  85 55       STA   55
```

megszüntetési lehetőség ellenőrzése

a füzérek összekapcsolása


```
B614  B1 4E       LDA   (4E),Y
B616  65 5F       ADC   5F
B618  85 5A       STA   5A
B61A  A5 60       LDA   60
B61C  69 00       ADC   #00
B61E  85 5B       STA   5B
B620  A5 33       LDA   33
B622  A6 34       LDX   34
B624  85 58       STA   58
B626  86 59       STX   59
B628  20 BF A3    JSR   A3BF
B62B  A4 55       LDY   55
B62D  C8          INY
B62E  A5 58       LDA   58
B630  91 4E       STA   (4E),Y
B632  AA          TAX
B633  E6 59       INC   59
B635  A5 59       LDA   59
B637  C8          INY
B638  91 4E       STA   (4E),Y
B63A  4C 2A B5    JMP   B52A

*****************************
B63D  A5 65       LDA   65
B63F  48          PHA
B640  A5 64       LDA   64
B642  48          PHA
B643  20 83 AE    JSR   AE83
B646  20 8F AD    JSR   AD8F
B649  68          PLA
B64A  85 6F       STA   6F
B64C  68          PLA
B64D  85 70       STA   70
B64F  A0 00       LDY   #00
B651  B1 6F       LDA   (6F),Y
B653  18          CLC
B654  71 64       ADC   (64),Y
B656  90 05       BCC   B65D
B658  A2 17       LDX   #17
B65A  4C 37 A4    JMP   A437
B65D  20 75 B4    JSR   B475
B660  20 7A B6    JSR   B67A
B663  A5 50       LDA   50
B665  A4 51       LDY   51
B667  20 AA B6    JSR   B6AA
B66A  20 8C B6    JSR   B68C
B66D  A5 6F       LDA   6F
B66F  A4 70       LDY   70
B671  20 AA B6    JSR   B6AA
B674  20 CA B4    JSR   B4CA
B677  4C B8 AD    JMP   ADB8

*****************************
B67A  A0 00       LDY   #00
B67C  B1 6F       LDA   (6F),Y
B67E  48          PHA
```

$5F/$60=mutató a régi blokk-kezdeten

$5A/$5B=mutató a régi blokkvégen

$58/$59=mutató az új blokkvégen

a füzér eltolása

"+" füzérek összekapcsolása

első füzér azonosítójának tárolása

második füzér beolvasása
a füzér-változó keresése

az azonosító visszatöltése

első füzér hossza

plusz a második füzér hossza
256-nál kisebb
"string too long" sorszáma
hibaüzenet kiírása
helyfogl. az összekapcsolt füzér számára
első füzér áthelyezése

mutató a második füzér azonosítójára
FRESTR
a második füzér hozzákapcs. az elsőhöz

FRESTR
az azonosító a füzér-veremben
vissza a kiértékeléshez

a füzér áthelyezése a lefoglalt területre

a füzér hosszának tárolása

```
B67F  C8          INY
B680  B1 6F       LDA  (6F),Y      a füzér-cím alsó byte
B682  AA          TAX
B683  C8          INY
B684  B1 6F       LDA  (6F),Y      a füzér-cím felső byte
B686  A8          TAY
B687  68          PLA              a füzér hossza
B688  86 22       STX  22          a füzér mutatója
B68A  84 23       STY  23
B68C  A8          TAY
B68D  F0 0A       BEQ  B699        ha a hossz nulla, akkor kész
B68F  48          PHA
B690  88          DEY
B691  B1 22       LDA  (22),Y      a füzér áthelyezése
B693  91 35       STA  (35),Y      a füzér-területre
B695  98          TYA
B696  D0 F8       BNE  B690
B698  68          PLA
B699  18          CLC
B69A  65 35       ADC  35          mutató, plusz a füzér hossza
B69C  85 35       STA  35
B69E  90 02       BCC  B6A2
B6A0  E6 36       INC  36
B6A2  60          RTS

*****************************
B6A3  20 8F AD    JSR  AD8F        a FRESTR füzérek kezelése
B6A6  A5 64       LDA  64          a füzér-változó keresése
B6A8  A4 65       LDY  65
B6AA  85 22       STA  22          az azonosító mutatója
B6AC  84 23       STY  23
B6AE  20 DB B6    JSR  B6DB        az azonosító törlése a veremből
B6B1  08          PHP
B6B2  A0 00       LDY  #00
B6B4  B1 22       LDA  (22),Y
B6B6  48          PHA
B6B7  C8          INY
B6B8  B1 22       LDA  (22),Y
B6BA  AA          TAX
B6BB  C8          INY
B6BC  B1 22       LDA  (22),Y
B6BE  A8          TAY
B6BF  68          PLA
B6C0  28          PLP
B6C1  D0 13       BNE  B6D6        a füzér nem volt a füzér-veremben
B6C3  C4 34       CPY  34
B6C5  D0 0F       BNE  B6D6
B6C7  E4 33       CPX  33
B6C9  D0 0B       BNE  B6D6
B6CB  48          PHA
B6CC  18          CLC
B6CD  65 33       ADC  33          a $33/$34 most a füzér-kezdetre utal
B6CF  85 33       STA  33
B6D1  90 02       BCC  B6D5
B6D3  E6 34       INC  34
B6D5  68          PLA
```



```
B6D6  86 22       STX  22
B6D8  84 23       STY  23
B6DA  60          RTS

*****************************
B6DB  C4 18       CPY  18          a füzér-mutató eltávolítása a veremből
B6DD  D0 0C       BNE  B6EB
B6DF  C5 17       CMP  17          a füzér-veremben van az azonosító?
B6E1  D0 08       BNE  B6EB
B6E3  85 16       STA  16          igen, a bejegyzés törlése
B6E5  E9 03       SBC  #03
B6E7  85 17       STA  17
B6E9  A0 00       LDY  #00
B6EB  60          RTS

*****************************
B6EC  20 A1 B7    JSR  B7A1        a CHR$ BASIC-függvény
B6EF  8A          TXA              a byte betöltése /0-tól 255-ig/
B6F0  48          PHA              a kód az akku-ban
B6F1  A9 01       LDA  #01         a füzér hossza=1
B6F3  20 7D B4    JSR  B47D        helyfoglalás
B6F6  68          PLA              az ASCII-kód visszaolvasása
B6F7  A0 00       LDY  #00
B6F9  91 62       STA  (62),Y      a füzér tárolása karakterenként
B6FB  68          PLA
B6FC  68          PLA
B6FD  4C CA B4    JMP  B4CA

*****************************
B700  20 61 B7    JSR  B761        az azonosító bevitele a verembe
B703  D1 50       CMP  (50),Y      a LEFT$ BASIC-függvény
B705  98          TYA              a füzér-paraméter és beolvasása a veremből
B706  90 04       BCC  B70C        hossz. összehas. a LEFT$-paraméterrel
B708  B1 50       LDA  (50),Y      LEFT$-paraméter kisebb a hossznál?
B70A  AA          TAX              hossz
B70B  98          TYA
B70C  48          PHA
B70D  8A          TXA
B70E  48          PHA
B70F  20 7D B4    JSR  B47D        helyfoglalás
B712  A5 50       LDA  50          az azonosító mutatója
B714  A4 51       LDY  51          FRESTR
B716  20 AA B6    JSR  B6AA
B719  68          PLA              az új füzér hossza
B71A  A8          TAY
B71B  68          PLA              plusz a régi füzér címe
B71C  18          CLC
B71D  65 22       ADC  22
B71F  85 22       STA  22
B721  90 02       BCC  B725
B723  E6 23       INC  23
B725  98          TYA
B726  20 8C B6    JSR  B68C        új füzér áthelyezése
B729  4C CA B4    JMP  B4CA        az azonosító tárolása a füzér-veremben

*****************************
```

a RIGHT$ BASIC-függvény

```
B72C  20 61 B7     JSR   B761
B72F  18           CLC
B730  F1 50        SBC   (50),Y
B732  49 FF        EOR   #FF
B734  4C 06 B7     JMP   B706

*****************************
B737  A9 FF        LDA   #FF
B739  85 65        STA   65
B73B  20 79 00     JSR   0079
B73E  C9 29        CMP   #29
B740  F0 06        BEQ   B748
B742  20 FD AE     JSR   AEFD
B745  20 9E B7     JSR   B79E
B748  20 61 B7     JSR   B761
B74B  F0 4B        BEQ   B798
B74D  CA           DEX
B74E  8A           TXA
B74F  48           PHA
B750  18           CLC
B751  A2 00        LDX   #00
B753  F1 50        SBC   (50),Y
B755  B0 B6        BCS   B70D
B757  49 FF        EOR   #FF
B759  C5 65        CMP   65
B75B  90 B1        BCC   B70E
B75D  A5 65        LDA   65
B75F  B0 AD        BCS   B70E

*****************************
B761  20 F7 AE     JSR   AEF7
B764  68           PLA
B765  A8           TAY
B766  68           PLA
B767  85 55        STA   55
B769  68           PLA
B76A  68           PLA
B76B  68           PLA
B76C  AA           TAX
B76D  68           PLA
B76E  85 50        STA   50
B770  68           PLA
B771  85 51        STA   51
B773  A5 55        LDA   55
B775  48           PHA
B776  98           TYA
B777  48           PHA
B778  A0 00        LDY   #00
B77A  8A           TXA
B77B  60           RTS

*****************************
B77C  20 82 B7     JSR   B782
B77F  4C A2 B3     JMP   B3A2

*****************************
```

a füzér-paraméter és hossz beolv. veremből

levonás a füzér hosszából
az első elem sorszáma a régi füzérben
tovább, mint a LEFT$-nál

a MID$ BASIC-függvény

CHRGOT, az utolsó karakter beolvasása
") " bezáró zárójel

vessző ellenőrzése
a byte és a második paraméter beolvasása
a füzér és a kezdőpozició beolvasása
ha az 1. paraméter Ø., "illegal quantity"

az első elem sorszáma a régi füzérben

régi füzér hossza
kisebb az első MID-paraméternél
az új hossz

feltétlen ugrás

a füzér, a sorszám, és az érték beolvasása
a bezáró zárójel ellenőrzése

a behívási cím tárolása

1. paraméter

az azonosító címének alsó és felső byte-ja

a behívási cím a verembe

hossz, második paraméter

a LEN BASIC-függvény
FRESTR, a füzér hosszán betöltése
konvertálás lebegőpontossá

a füzér-paraméter beolvasasa


```
B782  20 A3 B6     JSR   B6A3
B785  A2 00        LDX   #00
B787  86 0D        STX   0D
B789  A8           TAY
B78A  60           RTS

*****************************
B78B  20 82 B7     JSR   B782
B78E  F0 08        BEQ   B798
B790  A0 00        LDY   #00
B792  B1 22        LDA   (22),Y
B794  A8           TAY
B795  4C A2 B3     JMP   B3A2
B798  4C 48 B2     JMP   B248

*****************************
B79B  20 73 00     JSR   0073
B79E  20 8A AD     JSR   AD8A
B7A1  20 B8 B1     JSR   B1B8
B7A4  A6 64        LDX   64
B7A6  D0 F0        BNE   B798
B7A8  A6 65        LDX   65
B7AA  4C 79 00     JMP   0079

*****************************
B7AD  20 82 B7     JSR   B782
B7B0  D0 03        BNE   B7B5
B7B2  4C F7 B8     JMP   B8F7
B7B5  A6 7A        LDX   7A
B7B7  A4 7B        LDY   7B
B7B9  86 71        STX   71
B7BB  84 72        STY   72
B7BD  A6 22        LDX   22
B7BF  86 7A        STX   7A
B7C1  18           CLC
B7C2  65 22        ADC   22
B7C4  85 24        STA   24
B7C6  A6 23        LDX   23
B7C8  86 7B        STX   7B
B7CA  90 01        BCC   B7CD
B7CC  E8           INX
B7CD  86 25        STX   25
B7CF  A0 00        LDY   #00
B7D1  B1 24        LDA   (24),Y
B7D3  48           PHA
B7D4  98           TYA
B7D5  91 24        STA   (24),Y
B7D7  20 79 00     JSR   0079
B7DA  20 F3 BC     JSR   BCF3
B7DD  68           PLA
B7DE  A0 00        LDY   #00
B7E0  91 24        STA   (24),Y
B7E2  A6 71        LDX   71
B7E4  A4 72        LDY   72
B7E6  86 7A        STX   7A
B7E8  84 7B        STY   7B
```

FRESTR, a füzér beolvasása, hossz "A"-ban

a tipus-kapcsoló numerikus
a hossz az Y-ban

az ASC BASIC-függvény
a füzér beolvasása, mutató $22/$23, hossz Y-ba
ha a hossz=Ø, akkor "illegal quantity"

első karakter beolvasása
ASCII-kód
lebegőpontossá konvertálása
"illegal quantity"

az A-ben tárolt byte betöltése
CHRGET, a következő karakter beolvasása
FRENUM, numerikus érték a FAC-be
ellenőrzés és átváltás egész tipusúra
felső byte
ha nem =Ø, akkor "illegal quantity"

CHRGOT, az utolsó karakter beolvasása

a VAL BASIC-függvény
a füzér címe és hossza
a füzér hossza nem egyenlő nullával?
nulla a FAC-ben

a programmutató tárolása

a füzér kezdőcímének bevitele a mutatóba

a füzér végcíme + 1

a füzér első byte-ja
a verembe

és helyettesítése nullával
CHRGOT, az utolsó karakter beolvasása
a füzér konvertálása lebegőpontossá

ismét visszaállítani

a programmutató visszatöltése

```
B7EA  60          RTS

*****************************
B7EB  20 8A AD    JSR   AD8A
B7EE  20 F7 B7    JSR   B7F7
B7F1  20 FD AE    JSR   AEFD
B7F4  4C 9E B7    JMP   B79E

*****************************
B7F7  A5 66       LDA   66
B7F9  30 9D       BMI   B798
B7FB  A5 61       LDA   61
B7FD  C9 91       CMP   #91
B7FF  B0 97       BCS   B798
B801  20 9B BC    JSR   BC9B
B804  A5 64       LDA   64
B806  A4 65       LDY   65
B808  84 14       STY   14
B80A  85 15       STA   15
B80C  60          RTS

*****************************
B80D  A5 15       LDA   15
B80F  48          PHA
B810  A5 14       LDA   14
B812  48          PHA
B813  20 F7 B7    JSR   B7F7
B816  A0 00       LDY   #00
B818  B1 14       LDA   (14),Y
B81A  A8          TAY
B81B  68          PLA
B81C  85 14       STA   14
B81E  68          PLA
B81F  85 15       STA   15
B821  4C A2 B3    JMP   B3A2

*****************************
B824  20 EB B7    JSR   B7EB
B827  8A          TXA
B828  A0 00       LDY   #00
B82A  91 14       STA   (14),Y
B82C  60          RTS

*****************************
B82D  20 EB B7    JSR   B7EB
B830  86 49       STX   49
B832  A2 00       LDX   #00
B834  20 79 00    JSR   0079
B837  F0 03       BEQ   B83C
B839  20 F1 B7    JSR   B7F1
B83C  86 4A       STX   4A
B83E  A0 00       LDY   #00
B840  B1 14       LDA   (14),Y
B842  45 4A       EOR   4A
B844  25 49       AND   49
B846  F0 F6       BEQ   B840
```

GETADR és GETBYT 16 és 8 bites értéket tölt be
a FRMNUM pedig a numerikus értéket
a FAC konvertálás cím-formátumra $14/$15
a CHKCOM vesszőt keres
a byte betöltése az A szerint

GETADR FAC átalakítása poz. 16 bites számmá
előjel
ha negatív, akkor "illegal quantity"
hatványkitevő
szám összehas. 65536-tal
ha nagyobb, akkor "illegal quantity"
a FAC átalakítása cím-formátummá

az érték beolvasása
és tárolása $14/$15

a PEEK BASIC-függvény

a $14/$15 cím tárolása

a FAC átalakítása cím-formátummá

a PEEK-érték beolvasása
az Y-ba

a cím visszaolvasása

Y átalakítása lebegőpontos formátumra

a POKE BASIC-utasítás
a poke-cím és az érték beolvasása
a poke-érték az akku-ban

beírása a tárcímre

a WAIT BASIC-utasítás
cím és az érték beolvasása

a harmadik paraméter hibás
CHRGOT az utolsó karakter
nincs harmadik paraméter?
vessző keresése, a paraméter beolvasása

a wait-cím

logikai összekapcsolás
további várakozás


```
B848  60          RTS
*****************************
*****************************
B849  A9 11       LDA   #11
B84B  A0 BF       LDY   #BF
B84D  4C 67 B8    JMP   B867

*****************************
B850  20 8C BA    JSR   BA8C

*****************************
B853  A5 66       LDA   66
B855  49 FF       EOR   #FF
B857  85 66       STA   66
B859  45 6E       EOR   6E
B85B  85 6F       STA   6F
B85D  A5 61       LDA   61
B85F  4C 6A B8    JMP   B86A

*****************************
B862  20 99 B9    JSR   B999
B865  90 3C       BCC   B8A3

*****************************
B867  20 8C BA    JSR   BA8C

*****************************
B86A  D0 03       BNE   B86F
B86C  4C FC BB    JMP   BBFC
B86F  A6 70       LDX   70
B871  86 56       STX   56
B873  A2 69       LDX   #69
B875  A5 69       LDA   69
B877  A8          TAY
B878  F0 CE       BEQ   B848
B87A  38          SEC
B87B  E5 61       SBC   61
B87D  F0 24       BEQ   B8A3
B87F  90 12       BCC   B893
B881  84 61       STY   61
B883  A4 6E       LDY   6E
B885  84 66       STY   66
B887  49 FF       EOR   #FF
B889  69 00       ADC   #00
B88B  A0 00       LDY   #00
B88D  84 56       STY   56
B88F  A2 61       LDX   #61
B891  D0 04       BNE   B897
B893  A0 00       LDY   #00
B895  84 70       STY   70
B897  C9 F9       CMP   #F9
B899  30 C7       BMI   B862
B89B  A8          TAY
B89C  A5 70       LDA   70
B89E  56 01       LSR   01,X
B8A0  20 B0 B9    JSR   B9B0
```

aritmetikai mutatók
FAC = FAC + $0.5

a mutató a $0.5 állandóra
FAC=FAC + konstans (A/Y)

minusz FAC = konstans (A/Y) - FAC
az (A/Y) konstans az ARG-ban

minusz FAC = ARG - FAC

az előjel megfordítása

FAC = FAC + ARG

FAC és ARG hatványkitevőinek összehasonlítása

plusz FAC = konstans (A/Y) + FAC
az (A/Y) tárolása ARG-ben

plusz FAC = FAC + ARG
FAC egyenlő nullával?
ha igen, akkor FAC = ARG

```
B8A3  24 6F       BIT   6F
B8A5  10 57       BPL   B8FE
B8A7  A0 61       LDY   #61
B8A9  E0 69       CPX   #69
B8AB  F0 02       BEQ   B8AF
B8AD  A0 69       LDY   #69
B8AF  38          SEC
B8B0  49 FF       EOR   #FF
B8B2  65 56       ADC   56
B8B4  85 70       STA   70
B8B6  B9 04 00    LDA   0004,Y
B8B9  F5 04       SBC   04,X
B8BB  85 65       STA   65
B8BD  B9 03 00    LDA   0003,Y
B8C0  F5 03       SBC   03,X
B8C2  85 64       STA   64
B8C4  B9 02 00    LDA   0002,Y
B8C7  F5 02       SBC   02,X
B8C9  85 63       STA   63
B8CB  B9 01 00    LDA   0001,Y
B8CE  F5 01       SBC   01,X
B8D0  85 62       STA   62
B8D2  B0 03       BCS   B8D7
B8D4  20 47 B9    JSR   B947
B8D7  A0 00       LDY   #00
B8D9  98          TYA
B8DA  18          CLC
B8DB  A6 62       LDX   62
B8DD  D0 4A       BNE   B929
B8DF  A6 63       LDX   63
B8E1  86 62       STX   62
B8E3  A6 64       LDX   64
B8E5  86 63       STX   63
B8E7  A6 65       LDX   65
B8E9  86 64       STX   64
B8EB  A6 70       LDX   70
B8ED  86 65       STX   65
B8EF  84 70       STY   70
B8F1  69 08       ADC   #08
B8F3  C9 20       CMP   #20
B8F5  D0 E4       BNE   B8DB
B8F7  A9 00       LDA   #00
B8F9  85 61       STA   61
B8FB  85 66       STA   66
B8FD  60          RTS
B8FE  65 56       ADC   56
B900  85 70       STA   70
B902  A5 65       LDA   65
B904  65 6D       ADC   6D
B906  85 65       STA   65
B908  A5 64       LDA   64
B90A  65 6C       ADC   6C
B90C  85 64       STA   64
B90E  A5 63       LDA   63
B910  65 6B       ADC   6B
B912  85 63       STA   63
```

FAC mantisszájának invertálása


```
B914  A5 62       LDA   62
B916  65 6A       ADC   6A
B918  85 62       STA   62
B91A  4C 36 B9    JMP   B936
B91D  69 01       ADC   #01
B91F  06 70       ASL   70
B921  26 65       ROL   65
B923  26 64       ROL   64
B925  26 63       ROL   63
B927  26 62       ROL   62
B929  10 F2       BPL   B91D
B92B  38          SEC
B92C  E5 61       SBC   61
B92E  B0 C7       BCS   B8F7
B930  49 FF       EOR   #FF
B932  69 01       ADC   #01
B934  85 61       STA   61
B936  90 0E       BCC   B946
B938  E6 61       INC   61
B93A  F0 42       BEQ   B97E
B93C  66 62       ROR   62
B93E  66 63       ROR   63
B940  66 64       ROR   64
B942  66 65       ROR   65
B944  66 70       ROR   70
B946  60          RTS

*****************************
B947  A5 66       LDA   66
B949  49 FF       EOR   #FF
B94B  85 66       STA   66
B94D  A5 62       LDA   62
B94F  49 FF       EOR   #FF
B951  85 62       STA   62
B953  A5 63       LDA   63
B955  49 FF       EOR   #FF
B957  85 63       STA   63
B959  A5 64       LDA   64
B95B  49 FF       EOR   #FF
B95D  85 64       STA   64
B95F  A5 65       LDA   65
B961  49 FF       EOR   #FF
B963  85 65       STA   65
B965  A5 70       LDA   70
B967  49 FF       EOR   #FF
B969  85 70       STA   70
B96B  E6 70       INC   70
B96D  D0 0E       BNE   B97D
B96F  E6 65       INC   65
B971  D0 0A       BNE   B97D
B973  E6 64       INC   64
B975  D0 06       BNE   B97D
B977  E6 63       INC   63
B979  D0 02       BNE   B97D
B97B  E6 62       INC   62
B97D  60          RTS
```

FAC mantisszájának invertálása

az átviteleket figyelembevenni!

```
B97E  A2 0F       LDX  #0F        "overflow" sorszáma
B980  4C 37 A4    JMP  A437       a hibaüzenet kiírása

*****************************
B983  A2 25       LDX  #25        egy regiszter jobbraléptetése
B985  B4 04       LDY  04,X       a regiszter rel.cím mutatója /offset pointer/
B987  84 70       STY  70
B989  B4 03       LDY  03,X
B98B  94 04       STY  04,X
B98D  B4 02       LDY  02,X
B98F  94 03       STY  03,X
B991  B4 01       LDY  01,X
B993  94 02       STY  02,X
B995  A4 68       LDY  68
B997  94 01       STY  01,X
B999  69 08       ADC  #08
B99B  30 E8       BMI  B985
B99D  F0 E6       BEQ  B985
B99F  E9 08       SBC  #08
B9A1  A8          TAY
B9A2  A5 70       LDA  70
B9A4  B0 14       BCS  B9BA
B9A6  16 01       ASL  01,X
B9A8  90 02       BCC  B9AC
B9AA  F6 01       INC  01,X
B9AC  76 01       ROR  01,X
B9AE  76 01       ROR  01,X
B9B0  76 02       ROR  02,X
B9B2  76 03       ROR  03,X
B9B4  76 04       ROR  04,X
B9B6  6A          ROR  A
B9B7  C8          INY
B9B8  D0 EC       BNE  B9A6
B9BA  18          CLC
B9BB  60          RTS

*****************************
B9BC  81 00 00 00 00    a LOG - konstansok
B9C1  03                1
B9C2  7F 5E 56 CB 79    ha a fokszám=3, akkor 4 együttható
B9C7  80 13 9B 0B 64    .434255942
B9CC  80 76 38 93 16    .576584541
B9D1  82 38 AA 3B 20    .961800759
B9D6  80 35 04 F3 34    2.88539007
B9DB  81 35 04 F3 34    .707106781 = 1/SQR(2)
B9E0  80 80 00 00 00    1.41421356 = SQR(2)
B9E5  80 31 72 17 F8    -.5
                        .693147181 = LOG( 2)

*****************************
B9EA  20 2B BC    JSR  BC2B     a LOG BASIC-függvény
B9ED  F0 02       BEQ  B9F1     az előjel beolvasása
B9EF  10 03       BPL  B9F4     ha nulla, akkor kész
B9F1  4C 48 B2    JMP  B248     ha pozitív, akkor ok
B9F4  A5 61       LDA  61       "illegal quantity"
B9F6  E9 7F       SBC  #7F      a hatványkitevő
B9F8  48          PHA           normalizálása
                                és tárolása
```



```
B9F9  A9 80       LDA  #80      a szám leképezése
B9FB  85 61       STA  61       0.5-től 1-ig terjedő tartományba
B9FD  A9 D6       LDA  #D6
B9FF  A0 B9       LDY  #B9      1/SQR(2) konstans-mutató hozzáadása
BA01  20 67 B8    JSR  B867     a FAC-hez
BA04  A9 DB       LDA  #DB
BA06  A0 B9       LDY  #B9      SQR(2) konstans mutatója
BA08  20 0F BB    JSR  BB0F     SQR(2) osztása FAC-vel
BA0B  A9 BC       LDA  #BC
BA0D  A0 B9       LDY  #B9      1 konstans mutatója
BA0F  20 50 B8    JSR  B850     1 minusz FAC
BA12  A9 C1       LDA  #C1
BA14  A0 B9       LDY  #B9      a polinom együtthatójának mutatója
BA16  20 43 E0    JSR  E043     a polinom értékének számítása
BA19  A9 E0       LDA  #E0
BA1B  A0 B9       LDY  #B9      -0.5 konstans mutatója
BA1D  20 67 B8    JSR  B867     hozzáadása a FAC-hez
BA20  68          PLA           a hatványkitevő visszaolvasása
BA21  20 7E BD    JSR  BD7E     FAC = (FAC + FAC)
BA24  A9 E5       LDA  #E5      FAC = LOG(2)*FAC
BA26  A0 B9       LDY  #B9      LOG(2) konstans mutatója

*****************************
BA28  20 8C BA    JSR  BA8C     FAC = konstans ( A/Y ) * FAC

*****************************
BA2B  D0 03       BNE  BA30     FAC = ARG * FAC szorzás
BA2D  4C 8B BA    JMP  BA8B     nem nulla?
BA30  20 B7 BA    JSR  BAB7     RTS
BA33  A9 00       LDA  #00      a hatványkitevő kiszámítása
BA35  85 26       STA  26
BA37  85 27       STA  27
BA39  85 28       STA  28       a függvény-regiszter törlése
BA3B  85 29       STA  29
BA3D  A5 70       LDA  70
BA3F  20 59 BA    JSR  BA59     bitenkénti szorzás
BA42  A5 65       LDA  65
BA44  20 59 BA    JSR  BA59     bitenkénti szorzás
BA47  A5 64       LDA  64
BA49  20 59 BA    JSR  BA59     bitenkénti szorzás
BA4C  A5 63       LDA  63
BA4E  20 59 BA    JSR  BA59     bitenkénti szorzás
BA51  A5 62       LDA  62
BA53  20 5E BA    JSR  BA5E     bitenkénti szorzás
BA56  4C 8F BB    JMP  BB8F     reg. tartalma a FAC-be, eltolás balra

*****************************
BA59  D0 03       BNE  BA5E     bitenkénti szorzás
BA5B  4C 83 B9    JMP  B983
BA5E  4A          LSR  A        a reg. jobbraléptetése
BA5F  09 80       ORA  #80
BA61  A8          TAY
BA62  90 19       BCC  BA7D
BA64  18          CLC
BA65  A5 29       LDA  29
BA67  65 6D       ADC  6D
```

```
BA69  85 29       STA   29
BA6B  A5 28       LDA   28
BA6D  65 6C       ADC   6C
BA6F  85 28       STA   28
BA71  A5 27       LDA   27
BA73  65 6B       ADC   6B
BA75  85 27       STA   27
BA77  A5 26       LDA   26
BA79  65 6A       ADC   6A
BA7B  85 26       STA   26
BA7D  66 26       ROR   26
BA7F  66 27       ROR   27
BA81  66 28       ROR   28
BA83  66 29       ROR   29
BA85  66 70       ROR   70
BA87  98          TYA
BA88  4A          LSR   A
BA89  D0 D6       BNE   BA61
BA8B  60          RTS

*****************************
BA8C  85 22       STA   22        ARG = (A/Y) konstans
BA8E  84 23       STY   23        mutató
BA90  A0 04       LDY  #04        ARG = (22/23)
BA92  B1 22       LDA  (22),Y
BA94  85 6D       STA   6D
BA96  88          DEY
BA97  B1 22       LDA  (22),Y
BA99  85 6C       STA   6C
BA9B  88          DEY             a konstans ARG-be
BA9C  B1 22       LDA  (22),Y
BA9E  85 6B       STA   6B
BAA0  88          DEY
BAA1  B1 22       LDA  (22),Y
BAA3  85 6E       STA   6E
BAA5  45 66       EOR   66
BAA7  85 6F       STA   6F
BAA9  A5 6E       LDA   6E
BAAB  09 80       ORA  #80        előjel
BAAD  85 6A       STA   6A
BAAF  88          DEY
BAB0  B1 22       LDA  (22),Y
BAB2  85 69       STA   69        hatványkitevő
BAB4  A5 61       LDA   61
BAB6  60          RTS
BAB7  A5 69       LDA   69
BAB9  F0 1F       BEQ   BADA
BABB  18          CLC
BABC  65 61       ADC   61
BABE  90 04       BCC   BAC4
BAC0  30 1D       BMI   BADF
BAC2  18          CLC
BAC3  2C 10 14    BIT  1410
BAC6  69 80       ADC  #80
BAC8  85 61       STA   61
BACA  D0 03       BNE   BACF
```



```
BACC  4C FB B8    JMP   B8FB        FAC = 0
BACF  A5 6F       LDA   6F
BAD1  85 66       STA   66
BAD3  60          RTS
BAD4  A5 66       LDA   66
BAD6  49 FF       EOR  #FF
BAD8  30 05       BMI   BADF
BADA  68          PLA
BADB  68          PLA
BADC  4C F7 B8    JMP   B8F7        FAC = 0
BADF  4C 7E B9    JMP   B97E        "overflow error"

*****************************
BAE2  20 0C BC    JSR   BC0C        FAC = FAC * 10
BAE5  AA          TAX               a FAC kerekítése, tárolása ARG-be
BAE6  F0 10       BEQ   BAF8        ha a FAC nullával egyenlő, akkor kész
BAE8  18          CLC
BAE9  69 02       ADC  #02          a hatványkitevő +2, azaz négyszerezés
BAEB  B0 F2       BCS   BADF        átvitel?
BAED  A2 00       LDX  #00
BAEF  86 6F       STX   6F
BAF1  20 77 B8    JSR   B877        FAC = FAC + ARG = 5-szörözés
BAF4  E6 61       INC   61          hatványkitevő növelése, kétszerezés
BAF6  F0 E7       BEQ   BADF        ha átvitel, akkor "overflow"
BAF8  60          RTS

*****************************
BAF9  84 20 00 00 00              10 lebegőpontos konstans

*****************************
BAFE  20 0C BC    JSR   BC0C        FAC = FAC / 10
BB01  A9 F9       LDA  #F9          FAC kerekítése , tárolása ARG-be
BB03  A0 BA       LDY  #BA          FAC = ARG / 10
BB05  A2 00       LDX  #00          mutató a 10 konstansra
BB07  86 6F       STX   6F          FAC = ARG / (A/Y)
BB09  20 A2 BB    JSR   BBA2        a 10 konstans FAC-be
BB0C  4C 12 BB    JMP   BB12        FAC = ARG / FAC

*****************************
BB0F  20 8C BA    JSR   BA8C        FAC = a konstans (A/Y) / FAC
                                    a konstans (A/Y) az ARG-be
*****************************
BB12  F0 76       BEQ   BB8A        FAC = ARG / FAC
BB14  20 1B BC    JSR   BC1B        ha FAC nullával egyenlő "division by zero"
BB17  A9 00       LDA  #00          FAC kerekítése
BB19  38          SEC
BB1A  E5 61       SBC   61
BB1C  85 61       STA   61
BB1E  20 B7 BA    JSR   BAB7        az eredmény hatványkitevőjének meghatározása
BB21  E6 61       INC   61
BB23  F0 BA       BEQ   BADF        kitevő-túlcsordulás, "overflow"
BB25  A2 FC       LDX  #FC          a függvény-regiszter mutatója
BB27  A9 01       LDA  #01
BB29  A4 6A       LDY   6A
BB2B  C4 62       CPY   62
BB2D  D0 10       BNE   BB3F
```

```
BB2F  A4 6B       LDY   6B        az ARG byte-onkénti összehasonlítása FAC-vel
BB31  C4 63       CPY   63
BB33  D0 0A       BNE   BB3F
BB35  A4 6C       LDY   6C
BB37  C4 64       CPY   64
BB39  D0 04       BNE   BB3F
BB3B  A4 6D       LDY   6D
BB3D  C4 65       CPY   65
BB3F  08          PHP               a státusz tárolása
BB40  2A          ROL   A
BB41  90 09       BCC   BB4C
BB43  E8          INX
BB44  95 29       STA   29,X
BB46  F0 32       BEQ   BB7A
BB48  10 34       BPL   BB7E
BB4A  A9 01       LDA   #01
BB4C  28          PLP
BB4D  B0 0E       BCS   BB5D
BB4F  06 6D       ASL   6D
BB51  26 6C       ROL   6C
BB53  26 6B       ROL   6B
BB55  26 6A       ROL   6A
BB57  B0 E6       BCS   BB3F
BB59  30 CE       BMI   BB29
BB5B  10 E2       BPL   BB3F
BB5D  A8          TAY
BB5E  A5 6D       LDA   6D
BB60  E5 65       SBC   65
BB62  85 6D       STA   6D
BB64  A5 6C       LDA   6C
BB66  E5 64       SBC   64
BB68  85 6C       STA   6C
BB6A  A5 6B       LDA   6B
BB6C  E5 63       SBC   63
BB6E  85 6B       STA   6B
BB70  A5 6A       LDA   6A
BB72  E5 62       SBC   62
BB74  85 6A       STA   6A
BB76  98          TYA
BB77  4C 4F BB    JMP   BB4F
BB7A  A9 40       LDA   #40
BB7C  D0 CE       BNE   BB4C
BB7E  0A          ASL   A           akku * 64
BB7F  0A          ASL   A
BB80  0A          ASL   A
BB81  0A          ASL   A
BB82  0A          ASL   A
BB83  0A          ASL   A
BB84  85 70       STA   70
BB86  28          PLP
BB87  4C 8F BB    JMP   BB8F        a segédregiszter tartalma a FAC-be

*****************************
BB8A  A2 14       LDX   #14         "division by zero" sorszáma
BB8C  4C 37 A4    JMP   A437        hibaüzenet kiírása

*****************************      segédreg. tartalmának átvitele FAC-be /$26-$29/
```



```
BB8F  A5 26       LDA   26
BB91  85 62       STA   62
BB93  A5 27       LDA   27
BB95  85 63       STA   63
BB97  A5 28       LDA   28
BB99  85 64       STA   64
BB9B  A5 29       LDA   29
BB9D  85 65       STA   65
BB9F  4C D7 B8    JMP   B8D7        FAC eltolása balra

*****************************
BBA2  85 22       STA   22          (A/Y) konstans átvitele a FAC-be
BBA4  84 23       STY   23
BBA6  A0 04       LDY   #04         a mutató beállítása
BBA8  B1 22       LDA   (22),Y      FAC = (22,23)
BBAA  85 65       STA   65
BBAC  88          DEY
BBAD  B1 22       LDA   (22),Y      mantissza
BBAF  85 64       STA   64
BBB1  88          DEY
BBB2  B1 22       LDA   (22),Y
BBB4  85 63       STA   63
BBB6  88          DEY
BBB7  B1 22       LDA   (22),Y
BBB9  85 66       STA   66
BBBB  09 80       ORA   #80         mantissza-előjel
BBBD  85 62       STA   62
BBBF  88          DEY
BBC0  B1 22       LDA   (22),Y      hatványkitevő
BBC2  85 61       STA   61
BBC4  84 70       STY   70
BBC6  60          RTS

*****************************
BBC7  A2 5C       LDX   #5C         a FAC átvitele az Akku# 4-be
BBC9  2C                            az akku# 4 címének alsó byte-ja

*****************************
BBCA  A2 57       LDX   #57         a FAC átvitele az Akku# 3-ba
BBCC  A0 00       LDY   #00         akku# 3 cím alsó byte
BBCE  F0 04       BEQ   BBD4        felső byte (X,0) = FAC
                                    feltétlen ugrás

*****************************
BBD0  A6 49       LDX   49          a FAC átvitele a változóba
BBD2  A4 4A       LDY   4A
BBD4  20 1B BC    JSR   BC1B        a változó címe
BBD7  86 22       STX   22          a FAC kerekítése (X,Y) = FAC
BBD9  84 23       STY   23
BBDB  A0 04       LDY   #04         a célcím mutatója
BBDD  A5 65       LDA   65          (22,23) = FAC
BBDF  91 22       STA   (22),Y
BBE1  88          DEY
BBE2  A5 64       LDA   64
BBE4  91 22       STA   (22),Y
BBE6  88          DEY
BBE7  A5 63       LDA   63
```

```
BBE9  91 22       STA  (22),Y
BBEB  88          DEY
BBEC  A5 66       LDA   66
BBEE  09 7F       ORA  #7F
BBF0  25 62       AND   62          előjel átalakítása
BBF2  91 22       STA  (22),Y
BBF4  88          DEY
BBF5  A5 61       LDA   61
BBF7  91 22       STA  (22),Y
BBF9  84 70       STY   70
BBFB  60          RTS

*****************************     az ARG átvitele a FAC-be
BBFC  A5 6E       LDA   6E
BBFE  85 66       STA   66
BC00  A2 05       LDX  #05        5 byte
BC02  B5 68       LDA   68,X
BC04  95 60       STA   60,X
BC06  CA          DEX
BC07  D0 F9       BNE   BC02
BC09  86 70       STX   70
BC0B  60          RTS

*****************************     a FAC átvitele az ARG-be
BC0C  20 1B BC    JSR   BC1B      FAC kerekítése
BC0F  A2 06       LDX  #06
BC11  B5 60       LDA   60,X
BC13  95 68       STA   68,X
BC15  CA          DEX
BC16  D0 F9       BNE   BC11
BC18  86 70       STX   70
BC1A  60          RTS

*****************************     a FAC kerekítése
BC1B  A5 61       LDA   61        ha a hatványkitevő nulla, akkor kész
BC1D  F0 FB       BEQ   BC1A
BC1F  06 70       ASL   70        a kerekítési hely nagyobb $F7-nél?
BC21  90 F7       BCC   BC1A      ha nem, akkor kész
BC23  20 6F B9    JSR   B96F      mantissza növelése 1-gyel
BC26  D0 F2       BNE   BC1A      most nulla?
BC28  4C 38 B9    JMP   B938      jobbraléptetés, hatványkitevő növelése

*****************************     FAC előjelének beolvasása
BC2B  A5 61       LDA   61        nulla?
BC2D  F0 09       BEQ   BC38
BC2F  A5 66       LDA   66
BC31  2A          ROL   A
BC32  A9 FF       LDA  #FF        negatív
BC34  B0 02       BCS   BC38
BC36  A9 01       LDA  #01        pozitív
BC38  60          RTS

*****************************     az SGN BASIC-függvény
BC39  20 2B BC    JSR   BC2B      előjel beolvasása
BC3C  85 62       STA   62
BC3E  A9 00       LDA  #00
```



```
BC40  85 63       STA   63
BC42  A2 88       LDX  #88
BC44  A5 62       LDA   62
BC46  49 FF       EOR  #FF
BC48  2A          ROL   A
BC49  A9 00       LDA  #00
BC4B  85 65       STA   65
BC4D  85 64       STA   64
BC4F  86 61       STX   61
BC51  85 70       STA   70
BC53  85 66       STA   66
BC55  4C D2 B8    JMP   B8D2

*****************************     az ABS BASIC-függvény
BC58  46 66       LSR   66        az előjelbit törlése
BC5A  60          RTS

*****************************     az (A/Y) konstans összehasonlítása a FAC-vel
BC5B  85 24       STA   24
BC5D  84 25       STY   25        a konstans mutatója (24,25)
BC5F  A0 00       LDY  #00
BC61  B1 24       LDA  (24),Y 83  a hatványkitevő
BC63  C8          INY
BC64  AA          TAX
BC65  F0 C4       BEQ   BC2B      ha nulla, akkor a FAC előjelének beolvasása
BC67  B1 24       LDA  (24),Y 0
BC69  45 66       EOR   66
BC6B  30 C2       BMI   BC2F      a különböző előjelek /eltérő előjelek/
BC6D  E4 61       CPX   61
BC6F  D0 21       BNE   BC92
BC71  B1 24       LDA  (24),Y     az 1. byte összehasonlítása
BC73  09 80       ORA  #80
BC75  C5 62       CMP   62
BC77  D0 19       BNE   BC92
BC79  C8          INY
BC7A  B1 24       LDA  (24),Y     a 2. byte összehasonlítása
BC7C  C5 63       CMP   63
BC7E  D0 12       BNE   BC92
BC80  C8          INY
BC81  B1 24       LDA  (24),Y     a 3. byte összehasonlítása
BC83  C5 64       CMP   64
BC85  D0 0B       BNE   BC92
BC87  C8          INY
BC88  A9 7F       LDA  #7F
BC8A  C5 70       CMP   70
BC8C  B1 24       LDA  (24),Y     a 4. byte összehasonlítása
BC8E  E5 65       SBC   65
BC90  F0 28       BEQ   BCBA
BC92  A5 66       LDA   66
BC94  90 02       BCC   BC98
BC96  49 FF       EOR  #FF        ha az eredmény kisebb, akkor invertálás
BC98  4C 31 BC    JMP   BC31      az eredmény-kapcsoló beállítása

*****************************     lebegőpontos szám átalakítása egész típusra
BC9B  A5 61       LDA   61        hatványkitevő
BC9D  F0 4A       BEQ   BCE9      nulla?
```

```
BC9F  38          SEC
BCA0  E9 A0       SBC   #A0
BCA2  24 66       BIT    66
BCA4  10 09       BPL    BCAF
BCA6  AA          TAX
BCA7  A9 FF       LDA   #FF
BCA9  85 68       STA    68
BCAB  20 4D B9    JSR    B94D      a FAC mantisszájának invertálása
BCAE  8A          TXA
BCAF  A2 61       LDX   #61
BCB1  C9 F9       CMP   #F9
BCB3  10 06       BPL    BCBB
BCB5  20 99 B9    JSR    B999      a FAC jobbraléptetése
BCB8  84 68       STY    68
BCBA  60          RTS
BCBB  A8          TAY
BCBC  A5 66       LDA    66
BCBE  29 80       AND   #80
BCC0  46 62       LSR    62
BCC2  05 62       ORA    62
BCC4  85 62       STA    62
BCC6  20 B0 B9    JSR    B9B0      FAC bitenkénti jobbraléptetése
BCC9  84 68       STY    68
BCCB  60          RTS

*****************************
BCCC  A5 61       LDA    61        az INT BASIC-függvény
BCCE  C9 A0       CMP   #A0        hatványkitevő
BCD0  B0 20       BCS    BCF2      egész szám?
BCD2  20 9B BC    JSR    BC9B      ha igen, akkor kész
BCD5  84 70       STY    70        FAC/INTEGER konvertálás
BCD7  A5 66       LDA    66
BCD9  84 66       STY    66
BCDB  49 80       EOR   #80
BCDD  2A          ROL    A
BCDE  A9 A0       LDA   #A0
BCE0  85 61       STA    61
BCE2  A5 65       LDA    65
BCE4  85 07       STA    07
BCE6  4C D2 B8    JMP    B8D2      a FAC balratolása
BCE9  85 62       STA    62        mantissza kitöltése nullákkal
BCEB  85 63       STA    63
BCED  85 64       STA    64
BCEF  85 65       STA    65
BCF1  A8          TAY
BCF2  60          RTS

*****************************
BCF3  A0 00       LDY   #00        az ASCII átalakítása lebegőpontossá
BCF5  A2 0A       LDX   #0A
BCF7  94 5D       STY    5D,X      $5D-től $66-ig terjedő tartomány törlése
BCF9  CA          DEX
BCFA  10 FB       BPL    BCF7
BCFC  90 0F       BCC    BD0D
BCFE  C9 2D       CMP   #2D        "-"
BD00  D0 04       BNE    BD06
```



```
BD02  86 67       STX    67        a negatív kapcsoló
BD04  F0 04       BEQ    BD0A
BD06  C9 2B       CMP   #2B        "+"
BD08  D0 05       BNE    BD0F      CHRGET, a következő karakter beolvasása
BD0A  20 73 00    JSR    0073
BD0D  90 5B       BCC    BD6A
BD0F  C9 2E       CMP   #2E        "."
BD11  F0 2E       BEQ    BD41
BD13  C9 45       CMP   #45        "E"
BD15  D0 30       BNE    BD47
BD17  20 73 00    JSR    0073      CHRGET, a következő karakter beolvasása
BD1A  90 17       BCC    BD33
BD1C  C9 AB       CMP   #AB        "-" BASIC-kód
BD1E  F0 0E       BEQ    BD2E
BD20  C9 2D       CMP   #2D        "-"
BD22  F0 0A       BEQ    BD2E
BD24  C9 AA       CMP   #AA        "+" BASIC-kód
BD26  F0 08       BEQ    BD30
BD28  C9 2B       CMP   #2B        "+"
BD2A  F0 04       BEQ    BD30
BD2C  D0 07       BNE    BD35
BD2E  66 60       ROR    60        a 7. bit beállítása
BD30  20 73 00    JSR    0073      CHRGET, a következő karakter beolvasása
BD33  90 5C       BCC    BD91
BD35  24 60       BIT    60        a 7. bit beállítva?
BD37  10 0E       BPL    BD47      nem
BD39  A9 00       LDA   #00
BD3B  38          SEC
BD3C  E5 5E       SBC    5E
BD3E  4C 49 BD    JMP    BD49
BD41  66 5F       ROR    5F        tizedespont általi hívás
BD43  24 5F       BIT    5F
BD45  50 C3       BVC    BD0A
BD47  A5 5E       LDA    5E
BD49  38          SEC
BD4A  E5 5D       SBC    5D
BD4C  85 5E       STA    5E
BD4E  F0 12       BEQ    BD62
BD50  10 09       BPL    BD5B
BD52  20 FE BA    JSR    BAFE      FAC = FAC / 10
BD55  E6 5E       INC    5E
BD57  D0 F9       BNE    BD52
BD59  F0 07       BEQ    BD62
BD5B  20 E2 BA    JSR    BAE2      FAC = FAC * 10
BD5E  C6 5E       DEC    5E
BD60  D0 F9       BNE    BD5B
BD62  A5 67       LDA    67
BD64  30 01       BMI    BD67
BD66  60          RTS
BD67  4C B4 BF    JMP    BFB4      FAC = -FAC előjelváltás
BD6A  48          PHA
BD6B  24 5F       BIT    5F
BD6D  10 02       BPL    BD71
BD6F  E6 5D       INC    5D
BD71  20 E2 BA    JSR    BAE2      FAC = FAC * 10
BD74  68          PLA
```

```
BD75  38          SEC
BD76  E9 30       SBC  #30
BD78  20 7E BD    JSR  BD7E
BD7B  4C 0A BD    JMP  BD0A

BD7E  48          PHA
BD7F  20 0C BC    JSR  BC0C
BD82  68          PLA
BD83  20 3C BC    JSR  BC3C
BD86  A5 6E       LDA  6E
BD88  45 66       EOR  66
BD8A  85 6F       STA  6F
BD8C  A6 61       LDX  61
BD8E  4C 6A B8    JMP  B86A

BD91  A5 5E       LDA  5E
BD93  C9 0A       CMP  #0A
BD95  90 09       BCC  BDA0
BD97  A9 64       LDA  #64
BD99  24 60       BIT  60
BD9B  30 11       BMI  BDAE
BD9D  4C 7E B9    JMP  B97E

BDA0  0A          ASL  A
BDA1  0A          ASL  A
BDA2  18          CLC
BDA3  65 5E       ADC  5E
BDA5  0A          ASL  A
BDA6  18          CLC
BDA7  A0 00       LDY  #00
BDA9  71 7A       ADC  (7A),Y
BDAB  38          SEC
BDAC  E9 30       SBC  #30
BDAE  85 5E       STA  5E
BDB0  4C 30 BD    JMP  BD30

******************************
BDB3  9B 3E BC 1F FD
BDB8  9E 6E 6B 27 FD
BDBD  9E 6E 6B 28 00

******************************
BDC2  A9 71       LDA  #71
BDC4  A0 A3       LDY  #A3
BDC6  20 DA BD    JSR  BDDA
BDC9  A5 3A       LDA  3A
BDCB  A6 39       LDX  39

******************************
BDCD  85 62       STA  62
BDCF  86 63       STX  63
BDD1  A2 90       LDX  #90
BDD3  38          SEC
BDD4  20 49 BC    JSR  BC49
BDD7  20 DF BD    JSR  BDDF
BDDA  4C 1E AB    JMP  AB1E

******************************
BDDD  A0 01       LDY  #01
```

"0" levonása, hexadecimális eredmény
a köv. helyiérték hozzáadása a FAC-hez
a következő karakter

FAC/ARG

FAC = FAC + ARG
"E" általi hívás

"overflow error"

"0"

a következő karakter beolvasása

a lebegőpontos formátum/ASCII konv. konstansai
99999999.9
999999999
1E9

sorszám kiírása hibaüzenetnél

"in"-mutató
a füzér kiírása

az aktuális sorszám beolvasása

pozitív egész kiírása A/X-be

tárolás a FAC-ben

az egész átalakítása lebegőpontosra
FAC/ASCII konvertálás
a füzér kiírása

FAC konvertálása ASCII-formátumra


```
BDDF  A9 20       LDA  #20
BDE1  24 66       BIT  66
BDE3  10 02       BPL  BDE7
BDE5  A9 2D       LDA  #2D
BDE7  99 FF 00    STA  00FF,Y
BDEA  85 66       STA  66
BDEC  84 71       STY  71
BDEE  C8          INY
BDEF  A9 30       LDA  #30
BDF1  A6 61       LDX  61
BDF3  D0 03       BNE  BDF8
BDF5  4C 04 BF    JMP  BF04

BDF8  A9 00       LDA  #00
BDFA  E0 80       CPX  #80
BDFC  F0 02       BEQ  BE00
BDFE  B0 09       BCS  BE09
BE00  A9 BD       LDA  #BD
BE02  A0 BD       LDY  #BD
BE04  20 28 BA    JSR  BA28
BE07  A9 F7       LDA  #F7
BE09  85 5D       STA  5D
BE0B  A9 B8       LDA  #B8
BE0D  A0 BD       LDY  #BD
BE0F  20 5B BC    JSR  BC5B
BE12  F0 1E       BEQ  BE32
BE14  10 12       BPL  BE28
BE16  A9 B3       LDA  #B3
BE18  A0 BD       LDY  #BD
BE1A  20 5B BC    JSR  BC5B
BE1D  F0 02       BEQ  BE21
BE1F  10 0E       BPL  BE2F
BE21  20 E2 BA    JSR  BAE2
BE24  C6 5D       DEC  5D
BE26  D0 EE       BNE  BE16
BE28  20 FE BA    JSR  BAFE
BE2B  E6 5D       INC  5D
BE2D  D0 DC       BNE  BE0B
BE2F  20 49 B8    JSR  B849
BE32  20 9B BC    JSR  BC9B
BE35  A2 01       LDX  #01
BE37  A5 5D       LDA  5D
BE39  18          CLC
BE3A  69 0A       ADC  #0A
BE3C  30 09       BMI  BE47
BE3E  C9 0B       CMP  #0B
BE40  B0 06       BCS  BE48
BE42  69 FF       ADC  #FF
BE44  AA          TAX
BE45  A9 02       LDA  #02
BE47  38          SEC
BE48  E9 02       SBC  #02
BE4A  85 5E       STA  5E
BE4C  86 5D       STX  5D
BE4E  8A          TXA
BE4F  F0 02       BEQ  BE53
BE51  10 13       BPL  BE66
```

" " pozitív szám üres jele
előjel
pozitív?
"-" negatív szám minusz előjele
beírása a pufferba

"0"
a hatványkitevő
a szám nem nulla?
ha igen, akkor kész

FAC összehasonlítása 1-gyel

FAC 1-nél nagyobb

az 1E9 konstans mutató
konstans (A/Y mutató) * FAC

a 999999999 konstans mutatója
a konstans összehas.(A/Y mutató)a FAC-vel
egyenlő

mutató a 99999999.9-re
a konstans(A/Y mutató) összehas.a FAC-vel

FAC = FAC * 10

FAC = FAC / 10

FAC = FAC + .5, kerekítés
FAC/INTEGER

az összeg kisebb 0.1-nél?
az összeg nagyobb 1E9-nél?

```
BE53  A4 71       LDY  71          "."
BE55  A9 2E       LDA  #2E
BE57  C8          INY
BE58  99 FF 00    STA  00FF,Y
BE5B  8A          TXA
BE5C  F0 06       BEQ  BE64
BE5E  A9 30       LDA  #30         "0"
BE60  C8          INY
BE61  99 FF 00    STA  00FF,Y
BE64  84 71       STY  71
BE66  A0 00       LDY  #00         az egyes számjegyek kiszámítása
BE68  A2 80       LDX  #80
BE6A  A5 65       LDA  65
BE6C  18          CLC
BE6D  79 19 BF    ADC  BF19,Y
BE70  85 65       STA  65
BE72  A5 64       LDA  64
BE74  79 18 BF    ADC  BF18,Y
BE77  85 64       STA  64
BE79  A5 63       LDA  63
BE7B  79 17 BF    ADC  BF17,Y
BE7E  85 63       STA  63
BE80  A5 62       LDA  62
BE82  79 16 BF    ADC  BF16,Y
BE85  85 62       STA  62
BE87  E8          INX
BE88  B0 04       BCS  BE8E
BE8A  10 DE       BPL  BE6A
BE8C  30 02       BMI  BE90
BE8E  30 DA       BMI  BE6A
BE90  8A          TXA
BE91  90 04       BCC  BE97
BE93  49 FF       EOR  #FF
BE95  69 0A       ADC  #0A         10
BE97  69 2F       ADC  #2F         "0" - 1
BE99  C8          INY
BE9A  C8          INY
BE9B  C8          INY
BE9C  C8          INY
BE9D  84 47       STY  47
BE9F  A4 71       LDY  71
BEA1  C8          INY
BEA2  AA          TAX
BEA3  29 7F       AND  #7F
BEA5  99 FF 00    STA  00FF,Y
BEA8  C6 5D       DEC  5D
BEAA  D0 06       BNE  BEB2
BEAC  A9 2E       LDA  #2E         "."
BEAE  C8          INY
BEAF  99 FF 00    STA  00FF,Y
BEB2  84 71       STY  71
BEB4  A4 47       LDY  47
BEB6  8A          TXA
BEB7  49 FF       EOR  #FF
BEB9  29 80       AND  #80
BEBB  AA          TAX
```



```
BEBC  C0 24       CPY  #24         táblázat vége FAC-konvertálásnál
BEBE  F0 04       BEQ  BEC4
BEC0  C0 3C       CPY  #3C         táblázat vége TI$ számításnál
BEC2  D0 A6       BNE  BE6A
BEC4  A4 71       LDY  71
BEC6  B9 FF 00    LDA  00FF,Y
BEC9  88          DEY
BECA  C9 30       CMP  #30         "0"
BECC  F0 F8       BEQ  BEC6
BECE  C9 2E       CMP  #2E         "."
BED0  F0 01       BEQ  BED3
BED2  C8          INY
BED3  A9 2B       LDA  #2B         "+"
BED5  A6 5E       LDX  5E
BED7  F0 2E       BEQ  BF07
BED9  10 0B       BPL  BEE3
BEDB  A9 00       LDA  #00
BEDD  38          SEC
BEDE  E5 5E       SBC  5E
BEE0  AA          TAX
BEE1  A9 2D       LDA  #2D         "-"
BEE3  99 01 01    STA  0101,Y
BEE6  A9 45       LDA  #45         "E"
BEE8  99 00 01    STA  0100,Y
BEEB  8A          TXA
BEEC  A2 2F       LDX  #2F         "0" - 1
BEEE  38          SEC
BEEF  E8          INX
BEF0  E9 0A       SBC  #0A         10
BEF2  B0 FB       BCS  BEEF
BEF4  69 3A       ADC  #3A         "9" + 1
BEF6  99 03 01    STA  0103,Y
BEF9  8A          TXA
BEFA  99 02 01    STA  0102,Y
BEFD  A9 00       LDA  #00         a puffer lezárása $00-val
BEFF  99 04 01    STA  0104,Y
BF02  F0 08       BEQ  BF0C
BF04  99 FF 00    STA  00FF,Y
BF07  A9 00       LDA  #00         a puffer lezárása $00-val
BF09  99 00 01    STA  0100,Y
BF0C  A9 00       LDA  #00
BF0E  A0 01       LDY  #01         mutató a $0100-ra /puffer/
BF10  60          RTS

*****************************
BF11  80 00 00 00 00

*****************************
BF16  FA 0A 1F 00                  0.5 konstans az SQR függvényhez
BF1A  00 98 96 80                  32 bites bináris számok, előjellel
BF1E  FF F0 BD C0                  -100 000 000
BF22  00 01 86 A0                   10 000 000
BF26  FF FF D8 F0                   -1 000 000
BF2A  00 00 03 E8                     100 000
BF2E  FF FF FF 9C                     -10 000
BF32  00 00 00 0A                       1 000
BF36  FF FF FF FF                       -100
                                          10
                                          -1

*****************************      konstansok a TI/TI$ konvertáláshoz
```

```
BF3A  FF DF 0A 80
BF3E  00 03 4B C0
BF42  FF FF 73 60
BF46  00 00 0E 10
BF4A  FF FF FD A8
BF4E  00 00 00 3C
BF52  EC
BF53  AA  ...
BF70  ...  AA

*****************************

          **********
BF71  20 0C BC     JSR   BC0C
BF74  A9 11        LDA   #11
BF76  A0 BF        LDY   #BF

*****************************
BF7B  20 A2 BB     JSR   BBA2

*****************************
BF7B  F0 70        BEQ   BFED
BF7D  A5 69        LDA   69
BF7F  D0 03        BNE   BF84
BF81  4C F9 B8     JMP   B8F9
BF84  A2 4E        LDX   #4E
BF86  A0 00        LDY   #00
BF88  20 D4 BB     JSR   BBD4
BF8B  A5 6E        LDA   6E
BF8D  10 0F        BPL   BF9E
BF8F  20 CC BC     JSR   BCCC
BF92  A9 4E        LDA   #4E
BF94  A0 00        LDY   #00
BF96  20 5B BC     JSR   BC5B
BF99  D0 03        BNE   BF9E
BF9B  98           TYA
BF9C  A4 07        LDY   07
BF9E  20 FE BB     JSR   BBFE
BFA1  98           TYA
BFA2  48           PHA
BFA3  20 EA B9     JSR   B9EA
BFA6  A9 4E        LDA   #4E
BFA8  A0 00        LDY   #00
BFAA  20 28 BA     JSR   BA28
BFAD  20 ED BF     JSR   BFED
BFB0  68           PLA
BFB1  4A           LSR   A
BFB2  90 0A        BCC   BFBE

*****************************
BFB4  A5 61        LDA   61
BFB6  F0 06        BEQ   BFBE
BFB8  A5 66        LDA   66
BFBA  49 FF        EOR   #FF
BFBC  85 66        STA   66
BFBE  60           RTS
```

```
-2 16Ø ØØØ
  216 ØØØ
  -36 ØØØ
    3 6ØØ
    - 6ØØ
      6Ø
```

az SQR sor BASIC-függvény

FAC kerekítése ARG szerint

a Ø.5 konstans mutatója

hatványozás FAC=ARG felemelve az (A/Y) hatványra

FAC=ARG felemelve a FAC hatványkitevőre

ARG kitevője= bázis
nem nulla?
kész

segédakku . - mutató
FAC a segédakkuba
FAC kitevő = hatványkitevő
egynél kisebb?
INT-függvény

segédakku . mutatója
összehasonlítás a FAC-vel

ARG a FAC-be

a LOG függvény

a segédakku . mutatója
beszorzás a FAC-vel
az EXP függvény

előjelváltás
kitevő
ha a szám nullával egyenlő, akkor kész

az előjel invertálása


```
*****************************
BFBF  81 38 AA 3B 29
BFC4  07
BFC5  71 34 58 3E 56
BFCA  74 16 7E B3 1B
BFCF  77 2F EE E3 85
BFD4  7A 1D 84 1C 2A
BFD9  7C 63 59 58 0A
BFDE  7E 75 FD E7 C6
BFE3  80 31 72 18 10
BFE8  81 00 00 00 00

*****************************
BFED  A9 BF        LDA   #BF
BFEF  A0 BF        LDY   #BF
BFF1  20 28 BA     JSR   BA28
BFF4  A5 70        LDA   70
BFF6  69 50        ADC   #50
BFF8  90 03        BCC   BFFD
BFFA  20 23 BC     JSR   BC23
BFFD  4C 00 E0     JMP   E000
E000  85 56        STA   56
E002  20 0F BC     JSR   BC0F
E005  A5 61        LDA   61
E007  C9 88        CMP   #88
E009  90 03        BCC   E00E
E00B  20 D4 BA     JSR   BAD4
E00E  20 CC BC     JSR   BCCC
E011  A5 07        LDA   07
E013  18           CLC
E014  69 81        ADC   #81
E016  F0 F3        BEQ   E00B
E018  38           SEC
E019  E9 01        SBC   #01
E01B  48           PHA
E01C  A2 05        LDX   #05
E01E  B5 69        LDA   69,X
E020  B4 61        LDY   61,X
E022  95 61        STA   61,X
E024  94 69        STY   69,X
E026  CA           DEX
E027  10 F5        BPL   E01E
E029  A5 56        LDA   56
E02B  85 70        STA   70
E02D  20 53 B8     JSR   B853
E030  20 B4 BF     JSR   BFB4
E033  A9 C4        LDA   #C4
E035  A0 BF        LDY   #BF
E037  20 59 E0     JSR   E059
E03A  A9 00        LDA   #00
E03C  85 6F        STA   6F
E03E  68           PLA
E03F  20 B9 BA     JSR   BAB9
E042  60           RTS

*****************************
```

az EXP konstansai
1.442695Ø4 = 1/LOG(2)
7 = fokcím, 8 együttható
2.14987637E-5
1.4352314E-4
1.34226348E-3
9.614Ø117Ø1E-3
.555Ø51269
.24Ø226385
.693147186
1

az EXP BASIC-függvény

mutató 1/LOG(2) konstans
beszorzása a FAC-vel

a FAC mantisszájának növelése eggyel

FAC bevitele az ARG-ba
a kitevő
a szám nagyobb 128-nál?

ha pozitív, akkor "overflow"
az INTEGER függvény

127-tel egyenlő?

FAC és az ARG felcserélése

ARG - FAC
előjelváltás

mutató a polinom együtthatókra
a polinom kiszámítása

FAC és ARG kitevőinek összeadása

Y=A1*X+A2*X↑3+A3*X↑5+... polinomszámítás

```
E043  85 71       STA  71
E045  84 72       STY  72        az együttható-mutató
E047  20 CA BB    JSR  BBCA      a FAC betöltése az akku#3-ba
E04A  A9 57       LDA  #57       mutató az akku#3-ra
E04C  20 28 BA    JSR  BA28      FAC*akku#3 /négyzetreemelés/
E04F  20 5D E0    JSR  E05D      polinomszámítás
E052  A9 57       LDA  #57
E054  A0 00       LDY  #00       mutató az akku#3-ra
E056  4C 28 BA    JMP  BA28      FAC = FAC *akku#3

*****************************    Y=A0+A1*X+A2*X↑2+A3*X↑3+...polinomszámítás
E059  85 71       STA  71
E05B  84 72       STY  72        mutató a fokszámra
E05D  20 C7 BB    JSR  BBC7      a FAC betöltése az akku#4-be
E060  B1 71       LDA  (71),Y    fokszám
E062  85 67       STA  67        számláló
E064  A4 71       LDY  71
E066  C8          INY
E067  98          TYA            a mutató növelése,
E068  D0 02       BNE  E06C      az első együtthatóra mutat
E06A  E6 72       INC  72
E06C  85 71       STA  71
E06E  A4 72       LDY  72
E070  20 28 BA    JSR  BA28      FAC=FAC*konstans /A/Y-mutató szerint/
E073  A5 71       LDA  71
E075  A4 72       LDY  72
E077  18          CLC
E078  69 05       ADC  #05       a mutató növelése 5-tel, a következő szám
E07A  90 01       BCC  E07D
E07C  C8          INY
E07D  85 71       STA  71
E07F  84 72       STY  72
E081  20 67 BB    JSR  B867      FAC=FAC+konstans /A/Y-mutató/
E084  A9 5C       LDA  #5C
E086  A0 00       LDY  #00       mutató az akku#4-re
E088  C6 67       DEC  67        számláló értékének csökkentése
E08A  D0 E4       BNE  E070
E08C  60          RTS

*****************************    az RND konstansai
E08D  98 35 44 7A 00             11879546
E092  68 28 B1 46 00             3.92 67774E-4

*****************************    az RND BASIC-funkció
E097  20 2B BC    JSR  BC2B      előjel beolvasása
E09A  30 37       BMI  E0D3      negatív?
E09C  D0 20       BNE  E0BE
E09E  20 F3 FF    JSR  FFF3      a CIA címének beolvasása
E0A1  86 22       STX  22
E0A3  84 23       STY  23        és tárolása /mutató/
E0A5  A0 04       LDY  #04
E0A7  B1 22       LDA  (22),Y    "A" timer alsó
E0A9  85 62       STA  62
E0AB  C8          INY
E0AC  B1 22       LDA  (22).Y    "A" timer felső
E0AE  85 64       STA  64
```



```
E0B0  A0 08       LDY  #08
E0B2  B1 22       LDA  (22),Y    TOD 1/10 s
E0B4  85 63       STA  63
E0B6  C8          INY
E0B7  B1 22       LDA  (22),Y    TOD s
E0B9  85 65       STA  65
E0BB  4C E3 E0    JMP  E0E3
E0BE  A9 8B       LDA  #8B
E0C0  A0 00       LDY  #00       mutató az utolsó RND értéken
E0C2  20 A2 BB    JSR  BBA2      betöltés a FAC-be
E0C5  A9 8D       LDA  #8D
E0C7  A0 E0       LDY  #E0       mutató a konstansra
E0C9  20 28 BA    JSR  BA28      FAC=FAC* konstans
E0CC  A9 92       LDA  #92
E0CE  A0 E0       LDY  #E0       mutató a konstansra
E0D0  20 67 B8    JSR  B867      FAC=FAC+konstans
E0D3  A6 65       LDX  65
E0D5  A5 62       LDA  62
E0D7  85 65       STA  65
E0D9  86 62       STX  62        a helyiértékek felcserélése a FAC-ben
E0DB  A6 63       LDX  63
E0DD  A5 64       LDA  64
E0DF  85 63       STA  63
E0E1  86 64       STX  64
E0E3  A9 00       LDA  #00
E0E5  85 66       STA  66
E0E7  A5 61       LDA  61
E0E9  85 70       STA  70
E0EB  A9 80       LDA  #80       a kitevő
E0ED  85 61       STA  61
E0EF  20 D7 B8    JSR  B8D7      balrarendezés
E0F2  A2 8B       LDX  #8B
E0F4  A0 00       LDY  #00       mutató az utolsó RND értékre
E0F6  4C D4 BB    JMP  BBD4      a FAC kerekítése és tárolása

*****************************    hibakiértékelés az I/O rutinok után
E0F9  C9 F0       CMP  #F0       RS232 OPEN vagy CLOSE?
E0FB  D0 07       BNE  E104      nem
E0FD  84 38       STY  38        BASIC-RAM végének visszaállítása
E0FF  86 37       STX  37
E101  4C 63 A6    JMP  A663      és ugrás a CLR-hez
E104  AA          TAX            az X-ben tárol sorszám
E105  D0 02       BNE  E109      nem nulla?
E107  A2 1E       LDX  #1E       egyébként a "break" sorszáma
E109  4C 37 A4    JMP  A437      hibaüzenet kiírása

*****************************    a BASIC BSOUT
E10C  20 D2 FF    JSR  FFD2      egy karakter kiírása
E10F  B0 E8       BCS  E0F9      hiba?
E111  60          RTS

*****************************    a BASIC BASIN
E112  20 CF FF    JSR  FFCF      egy karakter beolvasása
E115  B0 E2       BCS  E0F9      hiba?
E117  60          RTS

*****************************    a BASIC CROUT
```

```
E118  20 AD E4    JSR   E4AD      az output egység kiválasztása
E11B  B0 DC       BCS   E0F9      hiba?
E11D  60          RTS

******************************
E11E  20 C6 FF    JSR   FFC6      a BASIC CHKIN
E121  B0 D6       BCS   E0F9      az input egység kiválasztása
E123  60          RTS             hiba?

******************************
E124  20 E4 FF    JSR   FFE4      a BASIC GETIN
E127  B0 D0       BCS   E0F9      egy karakter beolvasása
E129  60          RTS             hiba?

******************************
E12A  20 8A AD    JSR   AD8A      a SYS utasítás
E12D  20 F7 B7    JSR   B7F7      a FRMNUM numerikus kifejezés beolvasása
                                  által.címformátumra,betölt. $14,$15 címekre
E130  A9 E1       LDA   #E1
E132  48          PHA             a visszaugrási cím a verembe
E133  A9 46       LDA   #46
E135  48          PHA
E136  AD 0F 03    LDA   030F      a státusz
E139  48          PHA
E13A  AD 0C 03    LDA   030C      az akku
E13D  AE 0D 03    LDX   030D      az X-regiszter és
E140  AC 0E 03    LDY   030E      az Y-regiszter átadása
E143  28          PLP             a státusz beállítása
E144  6C 14 00    JMP   (0014)    a rutin behívása
E147  08          PHP             a státusz tárolása
E148  8D 0C 03    STA   030C      az akku,
E14B  8E 0D 03    STX   030D      az X-regiszter,
E14E  8C 0E 03    STY   030E      az Y-regiszter és
E151  68          PLA
E152  8D 0F 03    STA   030F      státusz újratárolása
E155  60          RTS

******************************
E156  20 D4 E1    JSR   E1D4      a SAVE-utasítás
E159  A6 2D       LDX   2D        paraméterek /file-név, elsődl. másodl. cím/
E15B  A4 2E       LDY   2E        a végcím egyenlő a BASIC program végével
E15D  A9 2B       LDA   #2B       a kezdőcím egyenlő a BASIC kezdet mutatójával
E15F  20 D8 FF    JSR   FFD8      a SAVE rutin
E162  B0 95       BCS   E0F9      hiba?
E164  60          RTS

******************************
E165  A9 01       LDA   #01       a VERIFY-utasítás
E167  2C                          a verify-kapcsoló

******************************
E168  A9 00       LDA   #00       a LOAD-utasítás
E16A  85 0A       STA   0A        a load-kapcsoló
E16C  20 D4 E1    JSR   E1D4      tárolása
E16F  A5 0A       LDA   0A        paraméter beolvasása
E171  A6 2B       LDX   2B        a kapcsoló
E173  A4 2C       LDY   2C        a kezdőcím egyenlő a BASIC-starttal
```



```
E175  20 D5 FF    JSR   FFD5      a load rutin
E178  B0 57       BCS   E1D1      hiba?
E17A  A5 0A       LDA   0A        load/verify-kapcsoló
E17C  F0 17       BEQ   E195      load?
E17E  A2 1C       LDX   #1C       "verify error" - offset
E180  20 B7 FF    JSR   FFB7      státusz beolvasása
E183  29 10       AND   #10       a hiba-bit leválasztása
E185  D0 17       BNE   E19E      ha a sátuszbit magas, akkor hiba!
E187  A5 7A       LDA   7A
E189  C9 02       CMP   #02       közvetlen mód?
E18B  F0 07       BEQ   E194      ha igen, akkor kész
E18D  A9 64       LDA   #64       "ok"-mutató
E18F  A0 A3       LDY   #A3
E191  4C 1E AB    JMP   AB1E      kiírás
E194  60          RTS
E195  20 B7 FF    JSR   FFB7      a státusz beolvasása
E198  29 BF       AND   #BF       az EOF bit törlése
E19A  F0 05       BEQ   E1A1      nincs hiba
E19C  A2 1D       LDX   #1D       "load error" offset
E19E  4C 37 A4    JMP   A437      a hibaüzenet kiírása
E1A1  A5 7B       LDA   7B
E1A3  C9 02       CMP   #02       közvetlen mód?
E1A5  D0 0E       BNE   E1B5      ha nem, akkor tovább
E1A7  86 2D       STX   2D        a végcím egyenlő a program végével
E1A9  84 2E       STY   2E
E1AB  A9 76       LDA   #76       mutató a "ready"-re
E1AD  A0 A3       LDY   #A3
E1AF  20 1E AB    JSR   AB1E      a szöveg kiírása
E1B2  4C 2A A5    JMP   A52A      a programsorok újraösszekapcsolása, CLR
E1B5  20 8E A6    JSR   A68E      CHRGET -mutató a programstarton
E1B8  20 33 A5    JSR   A533      a programsorok újraláncolása
E1BB  4C 77 A6    JMP   A677      RESTORE, BASIC inicializálás

******************************
E1BE  20 19 E2    JSR   E219      az OPEN BASIC-utasítás
E1C1  20 C0 FF    JSR   FFC0      paraméterek beolvasása
E1C4  B0 0B       BCS   E1D1      az OPEN rutin
E1C6  60          RTS             hiba?

******************************
E1C7  20 19 E2    JSR   E219      a CLOSE BASIC-utasítás
E1CA  A5 49       LDA   49        paraméterek beolvasása
E1CC  20 C3 FF    JSR   FFC3      a file sorszáma
E1CF  90 C3       BCC   E194      a CLOSE rutin
E1D1  4C F9 E0    JMP   E0F9      ha nincs hiba, RTS
                                  a hibakiértékeléshez

******************************
E1D4  A9 00       LDA   #00       a LOAD és SAVE paramétereinek beolvasása
E1D6  20 BD FF    JSR   FFBD      a file-név hossza a hibás
E1D9  A2 01       LDX   #01       file-paraméter beállítása
E1DB  A0 00       LDY   #00       hibás az egységszám
E1DD  20 BA FF    JSR   FFBA      hibás másodlagos cím
E1E0  20 06 E2    JSR   E206      a file-paraméter beállítása
E1E3  20 57 E2    JSR   E257      további karakter?
E1E6  20 06 E2    JSR   E206      a file-név beállítása
E1E9  20 00 E2    JSR   E200      további karakter?
                                  a paraméter beolvasása
```

```
E1EC  A0 00     LDY  #00     a másodlagos cím
E1EE  86 49     STX  49      az elsődleges cím
E1F0  20 BA FF  JSR  FFBA    a file-paraméter beállítása
E1F3  20 06 E2  JSR  E206    további karakter?
E1F6  20 00 E2  JSR  E200    paraméter behozatala
E1F9  8A        TXA
E1FA  A8        TAY
E1FB  A6 49     LDX  49      a másodlagos cím
E1FD  4C BA FF  JMP  FFBA    az egységszám
                             a file-paraméterek beállítása

*****************************
E200  20 0E E2  JSR  E20E    vessző és a további karakter vizsgalata
E203  4C 9E B7  JMP  B79E    egy byte beolvasása X-be

*****************************
E206  20 79 00  JSR  0079    további karakterek ellenőrzése
E209  D0 02     BNE  E20D    CHRGOT rutin, az utolsó karakter
E20B  68        PLA          ha van még karakter, akkor vissza!
E20C  68        PLA          visszatérés a hívó rutinhoz
E20D  60        RTS

*****************************
E20E  20 FD AE  JSR  AEFD    a vessző keresése
E211  20 79 00  JSR  0079    CHRGOT rutin, az utolsó karakter beolvasása
E214  D0 F7     BNE  E20D    ha van még karakter, akkor vissza!
E216  4C 08 AF  JMP  AF08    SYNTAX ERROR

*****************************
E219  A9 00     LDA  #00     az OPEN paramétereinek beolvasása
E21B  20 BD FF  JSR  FFBD    a file-név hossza hibás
E21E  20 11 E2  JSR  E211    a file-név paraméter beállítása
E221  20 9E B7  JSR  B79E    további karakter?
E224  86 49     STX  49      a logikai file-szám beolvasása X-be
E226  8A        TXA
E227  A2 01     LDX  #01     logikai file-szám
E229  A0 00     LDY  #00     hibás egységszám
E22B  20 BA FF  JSR  FFBA    hibás másodlagos cím
E22E  20 06 E2  JSR  E206    a file-paraméter beállítása
E231  20 00 E2  JSR  E200    további karakter?
E234  86 4A     STX  4A      egységszám beolvasása
E236  A0 00     LDY  #00     a másodlagos cím
E238  A5 49     LDA  49      logikai file-szám
E23A  E0 03     CPX  #03     az egységszám kisebb 3-nál?
E23C  90 01     BCC  E23F    igen
E23E  88        DEY
E23F  20 BA FF  JSR  FFBA    a file-paraméter beállítása
E242  20 06 E2  JSR  E206    további karakter?
E245  20 00 E2  JSR  E200    a másodlagos cím beolvasása
E248  8A        TXA
E249  A8        TAY          a másodlagos cím
E24A  A6 4A     LDX  4A      az egységszám
E24C  A5 49     LDA  49      a logikai file-szám
E24E  20 BA FF  JSR  FFBA    a file-paraméter beállítása
E251  20 06 E2  JSR  E206    további karakter?
E254  20 0E E2  JSR  E20E    a vessző keresése
E257  20 9E AD  JSR  AD9E    a FORMUL kifejezés beolvasása
```



```
E25A  20 A3 B6  JSR  B6A3    a szöveges paraméter beolvasása FROMSTR
E25D  A6 22     LDX  22      a file-név címe
E25F  A4 23     LDY  23      a file-név paramétereinek beállítása
E261  4C BD FF  JMP  FFBD

*****************************
E264  A9 E0     LDA  #E0     a COS BASIC-függvény
E266  A0 E2     LDY  #E2     a π/2 konstans hozzáadása a FAC-hez
E268  20 67 B8  JSR  B867

*****************************
E26B  20 0C BC  JSR  BC0C    a SIN BASIC-függvény
E26E  A9 E5     LDA  #E5     a FAC kerekítése és átvitele az ARG-be
E270  A0 E2     LDY  #E2
E272  A6 6E     LDX  6E      a π x 2 konstans mutató
E274  20 07 BB  JSR  BB07    FAC osztása 2xπ-vel
E277  20 0C BC  JSR  BC0C    FAC kerekítése és átvitele ARG-be
E27A  20 CC BC  JSR  BCCC    az INT függvény
E27D  A9 00     LDA  #00
E27F  85 6F     STA  6F
E281  20 53 B8  JSR  B853    ARG minusz FAC
E284  A9 EA     LDA  #EA
E286  A0 E2     LDY  #E2     0.25 konstans mutatója
E288  20 50 B8  JSR  B850    0.25 - FAC
E28B  A5 66     LDA  66
E28D  48        PHA          az előjel a veremben
E28E  10 0D     BPL  E29D    pozitív?
E290  20 49 B8  JSR  B849    FAC + 0.5
E293  A5 66     LDA  66      az előjel
E295  30 09     BMI  E2A0    negatív?
E297  A5 12     LDA  12
E299  49 FF     EOR  #FF     a kapcsoló átváltása
E29B  85 12     STA  12
E29D  20 B4 BF  JSR  BFB4    előjelcsere
E2A0  A9 EA     LDA  #EA
E2A2  A0 E2     LDY  #E2     0.25 konstans mutatója
E2A4  20 67 B8  JSR  B867    FAC + 0.25
E2A7  68        PLA          az előjel beolvasása
E2A8  10 03     BPL  E2AD    pozitív?
E2AA  20 B4 BF  JSR  BFB4    előjelcsere
E2AD  A9 EF     LDA  #EF
E2AF  A0 E2     LDY  #E2     mutató az együtthatóra
E2B1  4C 43 E0  JMP  E043    a polinom kiszámítása

*****************************
E2B4  20 CA BB  JSR  BBCA    a TAN BASIC-függvény
E2B7  A9 00     LDA  #00     FAC az akku#3-ba
E2B9  85 12     STA  12
E2BB  20 6B E2  JSR  E26B    a kapcsoló beállítása
E2BE  A2 4E     LDX  #4E     a SIN kiszámítása
E2C0  A0 00     LDY  #00
E2C2  20 F6 E0  JSR  E0F6    a segédakku mutatója
E2C5  A9 57     LDA  #57     a FAC a segédakkuba
E2C7  A0 00     LDY  #00
E2C9  20 A2 BB  JSR  BBA2    akku#3 mutatója
E2CC  A9 00     LDA  #00     akku#3 a FAC-be
```

```
E2CE  85 66        STA   66        előjel
E2D0  A5 12        LDA   12        kapcsoló
E2D2  20 DC E2     JSR   E2DC      a COS kiszámítása
E2D5  A9 4E        LDA   #4E
E2D7  A0 00        LDY   #00       a segédakku mutatója /SIN/
E2D9  4C 0F BB     JMP   BB0F      osztása FAC-vel
E2DC  48           PHA
E2DD  4C 9D E2     JMP   E29D      a COS kiszámítása

******************************
E2E0  81 49 0F DA A2               SIN- és COS-konstansok
E2E5  83 49 0F DA A2               1.57079633   π/2
E2EA  7F 00 00 00 00               6.28318531   2*π
E2EF  05                           .25
E2F0  85 E6 1A 2D 1B               5 = fokszám, 6 együttható
E2F5  86 28 07 FB F8               -14.3813907
E2FA  87 99 68 89 01               42.0077971
E2FF  87 23 35 DF E1               -76.7041703
E304  86 A5 5D E7 28               81.6052237
E309  83 49 0F DA A2               -41.3147021
                                   6.28318531   2*π
******************************     az ATN BASIC függvény
E30E  A5 66        LDA   66        az előjel
E310  48           PHA             tárolása
E311  10 03        BPL   E316      pozitív?
E313  20 B4 BF     JSR   BFB4      előjelcsere
E316  A5 61        LDA   61        a kitevő
E318  48           PHA             tárolása
E319  C9 81        CMP   #81       a szám összehasonlítása 1-gyel
E31B  90 07        BCC   E324      kisebb?
E31D  A9 BC        LDA   #BC
E31F  A0 B9        LDY   #B9       1 konstans - mutatója
E321  20 0F BB     JSR   BB0F      1 osztása a FAC-vel /reciprok/
E324  A9 3E        LDA   #3E
E326  A0 E3        LDY   #E3       mutató az együtthatóra
E328  20 43 E0     JSR   E043      a polinom kiszámítása
E32B  68           PLA             a kitevő visszaolvasása
E32C  C9 81        CMP   #81       a szám kisebb volt 1-nél?
E32E  90 07        BCC   E337
E330  A9 E0        LDA   #E0       π/2 konstans mutatója
E332  A0 E2        LDY   #E2       π/2 minusz FAC
E334  20 50 BB     JSR   BB50      az előjel beolvasása
E337  68           PLA             pozitív?
E338  10 03        BPL   E33D      előjelcsere
E33A  4C B4 BF     JMP   BFB4
E33D  60           RTS

******************************     az ATN függv. lebegőpontos konstansai
E33E  0B                           11 = fokszám, 12 együttható
E33F  76 B3 83 BD D3               -6.84793912E-04
E344  79 1E F4 A6 F5               4.85094216E-03
E349  7B 83 FC B0 10               -.0161117015
E34E  7C 0C 1F 67 CA               .034209638
E353  7C DE 53 CB C1               -.054279133
E358  7D 14 64 70 4C               .0724571965
E35D  7D B7 EA 51 7A               -.0898019185
E362  7D 63 30 88 7E               .1102932413
```



```
E367  7E 92 44 99 3A               -.142839808
E36C  7E 4C CC 91 C7               .19999912
E371  7F AA AA AA 13               -.333333316
E376  81 00 00 00 00               1

******************************     BASIC NMI-beugrás
E37B  20 CC FF     JSR   FFCC      CLRCH
E37E  A9 00        LDA   #00
E380  85 13        STA   13        az input egység a billentyúzet
E382  20 7A A6     JSR   A67A      a BASIC inicializálása
E385  58           CLI
E386  A2 80        LDX   #80       "nincs hiba" kapcsoló
E388  6C 00 03     JMP   (0300)    a BASIC melegindítás JMP $E38B vektorra
E38B  8A           TXA             a hibaszám az akku-ban
E38C  30 03        BMI   E391      ha nincs hiba, akkor "ready."
E38E  4C 3A A4     JMP   A43A      hibaüzenet
E391  4C 74 A4     JMP   A474      ready-mód

******************************     BASIC-hidegindítás
E394  20 53 E4     JSR   E453      a BASIC-vektorok beállítása
E397  20 BF E3     JSR   E3BF      a RAM inicializálása
E39A  20 22 E4     JSR   E422      bekapcsolási üzenet kiírása
E39D  A2 FB        LDX   #FB       a verem-mutató beállítása
E39F  9A           TXS
E3A0  D0 E4        BNE   E386      a melegindításhoz

******************************     a CHRGET rutin másolata
E3A2  E6 7A        INC   7A
E3A4  D0 02        BNE   E3A8      a mutató növelése a BASIC szövegben
E3A6  E6 7B        INC   7B
E3A8  AD 60 EA     LDA   EA60
E3AB  C9 3A        CMP   #3A       ":"
E3AD  B0 0A        BCS   E3B9
E3AF  C9 20        CMP   #20       " " üres jel átolvasása
E3B1  F0 EF        BEQ   E3A2
E3B3  38           SEC
E3B4  E9 30        SBC   #30       ha számjegy-ellenőrzés, akkor C=1
E3B6  38           SEC
E3B7  E9 D0        SBC   #D0
E3B9  60           RTS

******************************     az RND-függvény kezdőértéke
E3BA  80 4F C7 52 58               .811635157

******************************     a BASIC-RAM inicializálása
E3BF  A9 4C        LDA   #4C       JMP
E3C1  85 54        STA   54        a függvényekhez
E3C3  8D 10 03     STA   0310      a USR függvényhez
E3C6  A9 48        LDA   #48       "illegal quantity" mutató
E3C8  A0 B2        LDY   #B2
E3CA  8D 11 03     STA   0311      tárolása USR vektorként
E3CD  8C 12 03     STY   0312
E3D0  A9 91        LDA   #91       $B391
E3D2  A0 B3        LDY   #B3
E3D4  85 05        STA   05        vektor a fix.ill. lebegőpontos átalakításhoz
E3D6  84 06        STY   06
```

```
E3D8  A9 AA       LDA  #AA        $BLAA
E3DA  A0 B1       LDY  #B1        vektor a lebegő-, ill.fixpontos átalakításhoz
E3DC  85 03       STA   03
E3DE  84 04       STY   04
E3E0  A2 1C       LDX  #1C
E3E2  BD A2 E3    LDA   E3A2,X    a CHRGET rutin
E3E5  95 73       STA   73,X      bemásolása a RAM-ba
E3E7  CA          DEX
E3E8  10 F8       BPL   E3E2
E3EA  A9 03       LDA  #03        a garbage collect lépésköze
E3EC  85 53       STA   53
E3EE  A9 00       LDA  #00
E3F0  85 68       STA   68        az input egység a billentyűzet
E3F2  85 13       STA   13
E3F4  85 18       STA   18
E3F6  A2 01       LDX  #01
E3F8  8E FD 01    STX   01FD      álargumentum a láncolási címhez
E3FB  8E FC 01    STX   01FC
E3FE  A2 19       LDX  #19
E400  86 16       STX   16        a szövegkezelés mutatója
E402  38          SEC
E403  20 9C FF    JSR   FF9C      a RAM-start beolvasása és
E406  86 2B       STX   2B
E408  84 2C       STY   2C        tárolása BASIC-startként
E40A  38          SEC
E40B  20 99 FF    JSR   FF99      a RAM végének beolvasása és
E40E  86 37       STX   37
E410  84 38       STY   38        tárolása a BASIC végeként
E412  86 33       STX   33
E414  84 34       STY   34
E416  A0 00       LDY  #00
E418  98          TYA
E419  91 2B       STA   (2B),Y    a BASIC-startnál $00
E41B  E6 2B       INC   2B
E41D  D0 02       BNE   E421      a BASIC-start + 1
E41F  E6 2C       INC   2C
E421  60          RTS

*****************************
E422  A5 2B       LDA   2B
E424  A4 2C       LDY   2C        BASIC-RAM start-mutató
E426  20 08 A4    JSR   A408      az elegendő tárterület ellenőrzése
E429  A9 73       LDA  #73        a bekapcsolási üzenet mutatója
E42B  A0 E4       LDY  #E4
E42D  20 1E AB    JSR   AB1E      a szöveg kiírása
E430  A5 37       LDA   37
E432  38          SEC             a BASIC vége
E433  E5 2B       SBC   2B
E435  AA          TAX             minusz a BASIC-kezdet
E436  A5 38       LDA   38
E438  E5 2C       SBC   2C        egyenlő a szabad byte-okkal
E43A  20 CD BD    JSR   BDCD      a byte-ok számának kiírása
E43D  A9 60       LDA  #60        "basic bytes free" mutató
E43F  A0 E4       LDY  #E4
E441  20 1E AB    JSR   AB1E      a szöveg kiírása
E444  4C 44 A6    JMP   A644      ugrás a NEW-utasításhoz
```



```
*****************************
E447  8B E3 83 A4 7C A5 1A A7     a BASIC-vektorok táblázata
E44F  E4 A7 86 AE

*****************************
E453  A2 0B       LDX  #0B        a BASIC-vektorok betöltése
E455  BD 47 E4    LDA   E447,X
E458  9D 00 03    STA   0300,X
E45B  CA          DEX
E45C  10 F7       BPL   E455
E45E  60          RTS

*****************************
E45F  00 20 42 41 53 49 43 20     az operációs rendszer rendszerüzenetek
E467  42 59 54 45 53 20 46 52     basic bytes free
E46F  45 45 0D 00
E473  93 0D 20 20 20 20 2A 2A     /clr/ **** Commodore 64 basic v2 ****
E47B  2A 2A 20 43 4F 4D 4D 4F     /cr/ /cr/ 64k ram system
E483  44 4F 52 45 20 36 34 20
E48B  42 41 53 49 43 20 56 32
E493  20 2A 2A 2A 2A 0D 0D 20
E49B  36 34 4B 20 52 41 4D 20
E4A3  53 59 53 54 45 4D 20 20
E4AB  00
E4AC  5C
*****************************
E4AD  48          PHA             a BASIC CHROUT rutin
E4AE  20 C9 FF    JSR   FFC9      a CHROUT output egység beállítása
E4B1  AA          TAX             a hiba sorszáma X-be
E4B2  68          PLA
E4B3  90 01       BCC   E4B6      nincs hiba?
E4B5  8A          TXA             a hiba sorszáma ismét az akkuban
E4B6  60          RTS
E4B7  AA AA AA AA AA AA AA AA
E4BF  AA AA AA AA AA AA AA AA
E4C7  AA AA AA AA AA AA AA AA
E4CF  AA AA AA AA 85 A9 A9 01
E4D7  85 AB 60

*****************************
E4DA  AD 86 02    LDA   0286      a háttérszín beállítása
E4DD  91 F3       STA   (F3),Y    szín beolvasása és
E4DF  60          RTS             beírása a színramba

*****************************
E4E0  69 02       ADC  #02        várakozás a C= billentyűre
E4E2  A4 91       LDY   91        2*256/60 = 8.5 másodperc kivárása
E4E4  C8          INY             a kapcsoló vizsgálata
E4E5  D0 04       BNE   E4EB      a billentyű le van nyomva?
E4E7  C5 A1       CMP   A1        az idő még nem telt le?
E4E9  D0 F7       BNE   E4E2
E4EB  60          RTS

*****************************
E4EC  19 26       RS232 Baud Rate timer állandói, PAL-változat
                  $2619 = 9753          50 baud
```

```
E4EE  44 19
E4F0  1A 11
E4F2  E8 0D
E4F4  70 0C
E4F6  06 06
E4F8  D1 02
E4FA  37 01
E4FC  AE 00
E4FE  69 00

*****************************
E500  A2 00       LDX  #00
E502  A0 DC       LDY  #DC
E504  60          RTS

*****************************
E505  A2 28       LDX  #28
E507  A0 19       LDY  #19
E509  60          RTS

*****************************
E50A  B0 07       BCS  E513
E50C  86 D6       STX  D6
E50E  84 D3       STY  D3
E510  20 6C E5    JSR  E56C
E513  A6 D6       LDX  D6
E515  A4 D3       LDY  D3
E517  60          RTS

*****************************
E518  20 A0 E5    JSR  E5A0
E51B  A9 00       LDA  #00
E51D  8D 91 02    STA  0291
E520  85 CF       STA  CF
E522  A9 48       LDA  #48
E524  8D 8F 02    STA  028F
E527  A9 EB       LDA  #EB
E529  8D 90 02    STA  0290
E52C  A9 0A       LDA  #0A
E52E  8D 89 02    STA  0289
E531  8D 8C 02    STA  028C
E534  A9 0E       LDA  #0E
E536  8D 86 02    STA  0286
E539  A9 04       LDA  #04
E53B  8D 8B 02    STA  028B
E53E  A9 0C       LDA  #0C
E540  85 CD       STA  CD
E542  85 CC       STA  CC

*****************************
E544  AD 88 02    LDA  0288
E547  09 80       ORA  #80
E549  A8          TAY
E54A  A9 00       LDA  #00
E54C  AA          TAX
E54D  94 D9       STY  D9,X
```

$1944 = 6468 75 baud
$111A = 4378 110 baud
$0DEB = 3560 134.5 baud
$0C70 = 3184 150 baud
$0606 = 1542 300 baud
$02D1 = 736 600 baud
$0137 = 311 1200 baud
$00AE = 174 1800 baud
$0069 = 105 2400 baud

a CIA báziscímének beolvasása

$DC00

a sorok és oszlopok számának beolvasása
40 oszlop
25 sor

a kurzor beállítása /C=0/, beolvasása /C=1/

sor
oszlop
a kurzor beállítása

a képernyő RESET-je
videovezérlő inicializálása

shift-commodore engedélyezése
a kurzor nincs villogófázisban

/$028F/ = $EB48

a billentyűzet-dekódolás címmutatója
10
a billentyűzet-puffer max. hossza
"ismétlési sebesség" számláló
világoskék
a pillanatnyi szin

az ismétlési sebesség

a kurzor villogási idő
a kurzor villogás-kapcsoló

a képernyő törlése
a képernyő RAM területe

a sorok címe


```
E54F  18          CLC
E550  69 28       ADC  #28
E552  90 01       BCC  E555
E554  C8          INY
E555  E8          INX
E556  E0 1A       CPX  #1A
E558  D0 F3       BNE  E54D
E55A  A9 FF       LDA  #FF
E55C  95 D9       STA  D9,X
E55E  A2 18       LDX  #18
E560  20 FF E9    JSR  E9FF
E563  CA          DEX
E564  10 FA       BPL  E560

*****************************
E566  A0 00       LDY  #00
E568  84 D3       STY  D3
E56A  84 D6       STY  D6

*****************************
E56C  A6 D6       LDX  D6
E56E  A5 D3       LDA  D3
E570  B4 D9       LDY  D9,X
E572  30 08       BMI  E57C
E574  18          CLC
E575  69 28       ADC  #28
E577  85 D3       STA  D3
E579  CA          DEX
E57A  10 F4       BPL  E570
E57C  20 F0 E9    JSR  E9F0
E57F  A9 27       LDA  #27
E581  E8          INX
E582  B4 D9       LDY  D9,X
E584  30 06       BMI  E58C
E586  18          CLC
E587  69 28       ADC  #28
E589  E8          INX
E58A  10 F6       BPL  E582
E58C  85 D5       STA  D5
E58E  4C 24 EA    JMP  EA24
E591  E4 C9       CPX  C9
E593  F0 03       BEQ  E598
E595  4C ED E6    JMP  E6ED
E598  60          RTS
E599  EA          NOP

*****************************
E59A  20 A0 E5    JSR  E5A0
E59D  4C 66 E5    JMP  E566

*****************************
E5A0  A9 03       LDA  #03
E5A2  85 9A       STA  9A
E5A4  A9 00       LDA  #00
E5A6  85 99       STA  99
E5A8  A2 2F       LDX  #2F
```

40 hozzáadása /egy sor/

26, összes sor?

24, a sorok száma minusz 1
a sor törlése

cursor home

a kurzor oszlopa
a kurzor sora

a kurzorpozició kiszámítása, a mut. beáll.
a kurzor sora
a kurzor oszlopa

+40

a sor kezdőcímének MSB-je

+40

a videovezérlő inicializálása
a cursor home

a videovezérlő inicializálása

kiírás a képernyőre

beolvasás a billentyűzetről
47

```
E5AA  BD B8 EC    LDA   ECB8,X      a konstansok
E5AD  9D FF CF    STA   CFFF,X      beírása a videovezérlőbe
E5B0  CA          DEX
E5B1  D0 F7       BNE   E5AA
E5B3  60          RTS

*****************************     egy karakter beolv. a billentyűzet-pufferből
E5B4  AC 77 02    LDY   0277        az első karakter behozatala
E5B7  A2 00       LDX   #00
E5B9  BD 78 02    LDA   0278,X      a puffer tömörítése
E5BC  9D 77 02    STA   0277,X
E5BF  E8          INX
E5C0  E4 C6       CPX   C6          összehasonlítás a karakterek számával
E5C2  D0 F5       BNE   E5B9
E5C4  C6 C6       DEC   C6          a karakterek számának csökkentése
E5C6  98          TYA               az akkuban lévő karakter beolvasása
E5C7  58          CLI
E5C8  18          CLC
E5C9  60          RTS

*****************************     a billentyűzési várakozóciklus
E5CA  20 16 E7    JSR   E716        a karakter kiírása a képernyőre
E5CD  A5 C6       LDA   C6          a lenyomott billentyűk száma
E5CF  85 CC       STA   CC
E5D1  8D 92 02    STA   0292
E5D4  F0 F7       BEQ   E5CD        ha nincs billentyű lenyomva, akkor várakozá
E5D6  78          SEI
E5D7  A5 CF       LDA   CF          a kurzor villogó fázisban?
E5D9  F0 0C       BEQ   E5E7        nem
E5DB  A5 CE       LDA   CE          a karakter a kurzor alatt
E5DD  AE 87 02    LDX   0287        szín a kurzor alatt
E5E0  A0 00       LDY   #00
E5E2  84 CF       STY   CF          a kurzor nincs villogó fázisban
E5E4  20 13 EA    JSR   EA13        a karakter és a szín beállítása
E5E7  20 B4 E5    JSR   E5B4        a karakter beolv. a billentyűzet-pufferből
E5EA  C9 83       CMP   #83        "shift run" kód?
E5EC  D0 10       BNE   E5FE
E5EE  A2 09       LDX   #09        9 karakter
E5F0  78          SEI
E5F1  86 C6       STX   C6          a karakterek számának tárolása
E5F3  BD E6 EC    LDA   ECE6,X      "load /cr/ run /cr/"
E5F6  9D 76 02    STA   0276,X      bevitel a billentyűzet-pufferba
E5F9  CA          DEX
E5FA  D0 F7       BNE   E5F3
E5FC  F0 CF       BEQ   E5CD        és kiértékelés
E5FE  C9 0D       CMP   #0D         "CR"
E600  D0 C8       BNE   E5CA        ha nem, vissza a várakozási ciklushoz
E602  A4 D5       LDY   D5          a sorok hossza
E604  84 D0       STY   D0          CR-kapcsoló beállítása
E606  B1 D1       LDA   (D1),Y      a karakterek beolv. a képernyőről
E608  C9 20       CMP   #20         az üresjelek kiszűrése a sor végén
E60A  D0 03       BNE   E60F
E60C  88          DEY
E60D  D0 F7       BNE   E606
E60F  C8          INY
E610  84 C8       STY   C8          a pozició tárolása
```



```
E612  A0 00       LDY   #00
E614  8C 92 02    STY   0292        az oszlop nullával egyenlő
E617  84 D3       STY   D3
E619  84 D4       STY   D4          "egyszeres idézőjel" kapcsoló törlése
E61B  A5 C9       LDA   C9
E61D  30 1B       BMI   E63A
E61F  A6 D6       LDX   D6
E621  20 91 E5    JSR   E591
E624  E4 C9       CPX   C9
E626  D0 12       BNE   E63A
E628  A5 CA       LDA   CA          az utolsó oszlop
E62A  85 D3       STA   D3          bevitele az oszlop-mutatóba
E62C  C5 C8       CMP   C8          összehasonlítás az indexszel
E62E  90 0A       BCC   E63A
E630  B0 2B       BCS   E65D

*****************************     egy karakter beolvasása a képernyőről
E632  98          TYA
E633  48          PHA               a regiszter mentése
E634  8A          TXA
E635  48          PHA
E636  A5 D0       LDA   D0          a CR-kapcsoló
E638  F0 93       BEQ   E5CD        ha nem, akkor a várakozóciklushoz!
E63A  A4 D3       LDY   D3          oszlop
E63C  B1 D1       LDA   (D1),Y      a karakter beolv. a képernyőről
E63E  85 D7       STA   D7
E640  29 3F       AND   #3F
E642  06 D7       ASL   D7          és konvertálása ASCII-kódra
E644  24 D7       BIT   D7
E646  10 02       BPL   E64A
E648  09 80       ORA   #80
E64A  90 04       BCC   E650
E64C  A6 D4       LDX   D4
E64E  D0 04       BNE   E654
E650  70 02       BVS   E654
E652  09 40       ORA   #40
E654  E6 D3       INC   D3          a kurzor léptetése egy pozicióval
E656  20 84 E6    JSR   E684        "egyszeres idézőjel" vizsgálat
E659  C4 C8       CPY   C8          a kurzor az utolsó oszlopban?
E65B  D0 17       BNE   E674
E65D  A9 00       LDA   #00
E65F  85 D0       STA   D0          a "CR"-kapcsoló
E661  A9 0D       LDA   #0D
E663  A6 99       LDX   99
E665  E0 03       CPX   #03         beírás a képernyőről?
E667  F0 06       BEQ   E66F        igen
E669  A6 9A       LDX   9A          kiírás a képernyőre
E66B  E0 03       CPX   #03         igen
E66D  F0 03       BEQ   E672
E66F  20 16 E7    JSR   E716        sorbeírás a képernyőre
E672  A9 0D       LDA   #0D
E674  85 D7       STA   D7
E676  68          PLA
E677  AA          TAX               a regiszter visszaolvasása
E678  68          PLA
E679  A8          TAY
```

```
E67A  A5 D7       LDA   D7        a képernyőkód
E67C  C9 DE       CMP   #DE       összehasonlítása a π kóddal
E67E  D0 02       BNE   E682
E680  A9 FF       LDA   #FF       igen, helyettesítés BASIC-kóddal
E682  18          CLC
E683  60          RTS

*****************************
E684  C9 22       CMP   #22       "egyszeres idézőjel" vizsgálata
E686  D0 08       BNE   E690      "(")"?
E688  A5 D4       LDA   D4        ha nem, akkor kész
E68A  49 01       EOR   #01       az "egyszeres idézőjel" kapcsoló konvertálá
E68C  85 D4       STA   D4
E68E  A9 22       LDA   #22       az "egyszeres idézőjel" kód visszaállítása
E690  60          RTS

*****************************
E691  09 40       ORA   #40       a karakter kiírása a képernyőre
E693  A6 C7       LDX   C7        RVS?
E695  F0 02       BEQ   E699      átalakítás képernyőkódra
E697  09 80       ORA   #80       ha igen, akkor a 7.bit beállítása
E699  A6 D8       LDX   D8
E69B  F0 02       BEQ   E69F
E69D  C6 D8       DEC   D8
E69F  AE 86 02    LDX   0286      a színkód
E6A2  20 13 EA    JSR   EA13      a karakter beírása a képernyőram-ba
E6A5  20 B6 E6    JSR   E6B6      a "sorkezdet" táblázat aktualizálása
E6A8  68          PLA
E6A9  A8          TAY
E6AA  A5 D8       LDA   D8
E6AC  F0 02       BEQ   E6B0
E6AE  46 D4       LSR   D4        "egyszeres idézőjel" kód törlése
E6B0  68          PLA
E6B1  AA          TAX
E6B2  68          PLA
E6B3  18          CLC
E6B4  58          CLI
E6B5  60          RTS

*****************************
E6B6  20 B3 E8    JSR   E8B3      a sor-kezdet MSB-jének újraszámítása
E6B9  E6 D3       INC   D3
E6BB  A5 D5       LDA   D5
E6BD  C5 D3       CMP   D3
E6BF  B0 3F       BCS   E700
E6C1  C9 4F       CMP   #4F       79 karakter /kettős sor/?
E6C3  F0 32       BEQ   E6F7
E6C5  AD 92 02    LDA   0292
E6C8  F0 03       BEQ   E6CD
E6CA  4C 67 E9    JMP   E967
E6CD  A6 D6       LDX   D6        sor
E6CF  E0 19       CPX   #19       25?
E6D1  90 07       BCC   E6DA
E6D3  20 EA E8    JSR   E8EA
E6D6  C6 D6       DEC   D6
E6D8  A6 D6       LDX   D6
```



```
E6DA  16 D9       ASL   D9,X
E6DC  56 D9       LSR   D9,X
E6DE  E8          INX
E6DF  B5 D9       LDA   D9,X
E6E1  09 80       ORA   #80
E6E3  95 D9       STA   D9,X
E6E5  CA          DEX
E6E6  A5 D5       LDA   D5
E6E8  18          CLC
E6E9  69 28       ADC   #28       40 hozzáadása
E6EB  85 D5       STA   D5
E6ED  B5 D9       LDA   D9,X
E6EF  30 03       BMI   E6F4
E6F1  CA          DEX
E6F2  D0 F9       BNE   E6ED
E6F4  4C F0 E9    JMP   E9F0      az X.sor mutatója a színramban
E6F7  C6 D6       DEC   D6        sorok csökkentése
E6F9  20 7C E8    JSR   E87C
E6FC  A9 00       LDA   #00
E6FE  85 D3       STA   D3        oszlop = 0
E700  60          RTS

*****************************
E701  A6 D6       LDX   D6        visszalépés az előző sorra
E703  D0 06       BNE   E70B      a kurzor sora
E705  86 D3       STX   D3        nulla?
E707  68          PLA             a kurzor oszlopa
E708  68          PLA
E709  D0 9D       BNE   E6A8
E70B  CA          DEX
E70C  86 D6       STX   D6        a sorszám csökkentése
E70E  20 6C E5    JSR   E56C      a kurzorpozició kiszámítása
E711  A4 D5       LDY   D5
E713  84 D3       STY   D3
E715  60          RTS

*****************************
E716  48          PHA             kiírás a képernyőre
E717  85 D7       STA   D7        a karakter tárolása
E719  8A          TXA
E71A  48          PHA             a regiszter mentése
E71B  98          TYA
E71C  48          PHA
E71D  A9 00       LDA   #00
E71F  85 D0       STA   D0
E721  A4 D3       LDY   D3
E723  A5 D7       LDA   D7
E725  10 03       BPL   E72A
E727  4C D4 E7    JMP   E7D4      07F-nél nagyobb karakter lekezelése
E72A  C9 0D       CMP   #0D       "carriage return"?
E72C  D0 03       BNE   E731
E72E  4C 91 E8    JMP   E891      a RETURN kiírása
E731  C9 20       CMP   #20       " "
E733  90 10       BCC   E745      a kinyomtatandó karakter?
E735  C9 60       CMP   #60
E737  90 04       BCC   E73D
```

```
E739  29 DF       AND  #DF
E73B  D0 02       BNE  E73F
E73D  29 3F       AND  #3F
E73F  20 84 E6    JSR  E684
E742  4C 93 E6    JMP  E693
E745  A6 D8       LDX  D8
E747  F0 03       BEQ  E74C
E749  4C 97 E6    JMP  E697
E74C  C9 14       CMP  #14
E74E  D0 2E       BNE  E77E
E750  98          TYA
E751  D0 06       BNE  E759
E753  20 01 E7    JSR  E701
E756  4C 73 E7    JMP  E773
E759  20 A1 E8    JSR  E8A1
E75C  88          DEY
E75D  84 D3       STY  D3
E75F  20 24 EA    JSR  EA24
E762  C8          INY
E763  B1 D1       LDA  (D1),Y
E765  88          DEY
E766  91 D1       STA  (D1),Y
E768  C8          INY
E769  B1 F3       LDA  (F3),Y
E76B  88          DEY
E76C  91 F3       STA  (F3),Y
E76E  C8          INY
E76F  C4 D5       CPY  D5
E771  D0 EF       BNE  E762
E773  A9 20       LDA  #20
E775  91 D1       STA  (D1),Y
E777  AD 86 02    LDA  0286
E77A  91 F3       STA  (F3),Y
E77C  10 4D       BPL  E7CB
E77E  A6 D4       LDX  D4
E780  F0 03       BEQ  E785
E782  4C 97 E6    JMP  E697
E785  C9 12       CMP  #12
E787  D0 02       BNE  E78B
E789  85 C7       STA  C7
E78B  C9 13       CMP  #13
E78D  D0 03       BNE  E792
E78F  20 66 E5    JSR  E566
E792  C9 1D       CMP  #1D
E794  D0 17       BNE  E7AD
E796  C8          INY
E797  20 B3 E8    JSR  E8B3
E79A  84 D3       STY  D3
E79C  88          DEY
E79D  C4 D5       CPY  D5
E79F  90 09       BCC  E7AA
E7A1  C6 D6       DEC  D6
E7A3  20 7C E8    JSR  E87C
E7A6  A0 00       LDY  #00
E7A8  84 D3       STY  D3
E7AA  4C A8 E6    JMP  E6A8
```

"egyszeres idézőjel" teszt
az ASCII-kód átvált. képernyőkódra kiíráshoz

"DEL"?

vissza az előző sorba

a szinram mutatójának kiszámítása

egy karakter a képernyőről

balraléptetés eggyel

szín

balraléptetés eggyel

üresjel beszúrása

a színkód beállítása
kész
"egyszeres idézőjel" mód?
nem
fordítás karakter kiírása
"RVS ON"?
nem
RVS-kapcsoló beállítása
"HOME"?
nem
igen, kurzor home
"cursor right"?
nem

az oszlop nullával egyenlő
kész


```
E7AD  C9 11       CMP  #11
E7AF  D0 1D       BNE  E7CE
E7B1  18          CLC
E7B2  98          TYA
E7B3  69 28       ADC  #28
E7B5  A8          TAY
E7B6  E6 D6       INC  D6
E7B8  C5 D5       CMP  D5
E7BA  90 EC       BCC  E7A8
E7BC  F0 EA       BEQ  E7A8
E7BE  C6 D6       DEC  D6
E7C0  E9 28       SBC  #28
E7C2  90 04       BCC  E7C8
E7C4  85 D3       STA  D3
E7C6  D0 F8       BNE  E7C0
E7C8  20 7C E8    JSR  E87C
E7CB  4C A8 E6    JMP  E6A8
E7CE  20 CB E8    JSR  E8CB
E7D1  4C 44 EC    JMP  EC44

*****************************
E7D4  29 7F       AND  #7F
E7D6  C9 7F       CMP  #7F
E7D8  D0 02       BNE  E7DC
E7DA  A9 5E       LDA  #5E
E7DC  C9 20       CMP  #20
E7DE  90 03       BCC  E7E3
E7E0  4C 91 E6    JMP  E691
E7E3  C9 0D       CMP  #0D
E7E5  D0 03       BNE  E7EA
E7E7  4C 91 E8    JMP  E891
E7EA  A6 D4       LDX  D4
E7EC  D0 3F       BNE  E82D
E7EE  C9 14       CMP  #14
E7F0  D0 37       BNE  E829
E7F2  A4 D5       LDY  D5
E7F4  B1 D1       LDA  (D1),Y
E7F6  C9 20       CMP  #20
E7F8  D0 04       BNE  E7FE
E7FA  C4 D3       CPY  D3
E7FC  D0 07       BNE  E805
E7FE  C0 4F       CPY  #4F
E800  F0 24       BEQ  E826
E802  20 65 E9    JSR  E965
E805  A4 D5       LDY  D5
E807  20 24 EA    JSR  EA24
E80A  88          DEY
E80B  B1 D1       LDA  (D1),Y
E80D  C8          INY
E80E  91 D1       STA  (D1),Y
E810  88          DEY
E811  B1 F3       LDA  (F3),Y
E813  C8          INY
E814  91 F3       STA  (F3),Y
E816  88          DEY
E817  C4 D3       CPY  D3
```

"Cursor down"?
nem

plusz 40, egy sorral lejjebb

40 levonása

kész
a színkód ellenőrzése
további speciális karakterek tesztelése

karakter $127-nél nagyobb
kód 127-nél nagyobb, a 7. bit törlése
"π"?

π képernyőkódja
vezérlőkarakter?
igen
a karakter kiírása
"shift return"?

új sor
"egyszeres idézőjel" mód?
igen, vezérlőkarakter shiftelt kiírása
"INS"?

a sor hossza
az utolsó karakter a sorban
üres jel?

a kurzor az utolsó oszlopban?

79? maximális sorhossz
ha az utolsó oszlop, akkor nincs művelet
üres sor beszúrása
a sor hossza
a színram-mutató kiszámítása

a karakter a képernyőről

jobbraléptetés eggyel

és a szín

eltolása

felzárkóztatás az aktuális pozícióig

```
E819  D0 EF        BNE   E80A
E81B  A9 20        LDA   #20        üres karakter beírása
E81D  91 D1        STA   (D1),Y     a pillanatnyi pozicióra
E81F  AD 86 02     LDA   0286       a szín
E822  91 F3        STA   (F3),Y     beállítása
E824  E6 D8        INC   D8         a beszúrások számának növelése
E826  4C A8 E6     JMP   E6A8
E829  A6 D8        LDX   D8
E82B  F0 05        BEQ   E832
E82D  09 40        ORA   #40
E82F  4C 97 E6     JMP   E697

*****************************
E832  C9 11        CMP   #11        cursor up
E834  D0 16        BNE   E84C
E836  A6 D6        LDX   D6         a sor
E838  F0 37        BEQ   E871       ha nulla, akkor kész
E83A  C6 D6        DEC   D6         a sorszám csökkentése eggyel
E83C  A5 D3        LDA   D3         az oszlop
E83E  38           SEC
E83F  E9 28        SBC   #28        40 levonása
E841  90 04        BCC   E847
E843  85 D3        STA   D3         a kurzor oszlopa
E845  10 2A        BPL   E871       pozitív, ok
E847  20 6C E5     JSR   E56C       a képernyő-mutató újrabeállítása
E84A  D0 25        BNE   E871
E84C  C9 12        CMP   #12        "RVS OFF"
E84E  D0 04        BNE   E854
E850  A9 00        LDA   #00
E852  85 C7        STA   C7         az RVS-kapcsoló törlése
E854  C9 1D        CMP   #1D        "cursor left"?
E856  D0 12        BNE   E86A
E858  98           TYA
E859  F0 09        BEQ   E864
E85B  20 A1 E8     JSR   E8A1
E85E  88           DEY
E85F  84 D3        STY   D3         a kurzor oszlopa
E861  4C A8 E6     JMP   E6A8       kész
E864  20 01 E7     JSR   E701       visszalépés az előző sorba
E867  4C A8 E6     JMP   E6A8       kész
E86A  C9 13        CMP   #13        "CLR SCREEN"?
E86C  D0 06        BNE   E874
E86E  20 44 E5     JSR   E544       a képernyő törlése
E871  4C A8 E6     JMP   E6A8       kész
E874  09 80        ORA   #80        a 7. bit helyreállítása
E876  20 CB E8     JSR   E8CB       a színkód ellenőrzése
E879  4C 4F EC     JMP   EC4F       szöveg/grafika konvertálás ellenőrzése
E87C  46 C9        LSR   C9
E87E  A6 D6        LDX   D6
E880  E8           INX
E881  E0 19        CPX   #19        25, utolsó sor
E883  D0 03        BNE   E888
E885  20 EA E8     JSR   E8EA       a képernyő görgetése
E888  B5 D9        LDA   D9,X
E88A  10 F4        BPL   E880
E88C  86 D6        STX   D6
```



```
E88E  4C 6C E5     JMP   E56C       a kurzorpozició kiszámítása
E891  A2 00        LDX   #00
E893  86 D8        STX   D8
E895  86 C7        STX   C7         a kapcsolók törlése
E897  86 D4        STX   D4
E899  86 D3        STX   D3
E89B  20 7C E8     JSR   E87C
E89E  4C A8 E6     JMP   E6A8       kész
E8A1  A2 02        LDX   #02
E8A3  A9 00        LDA   #00
E8A5  C5 D3        CMP   D3
E8A7  F0 07        BEQ   E8B0
E8A9  18           CLC
E8AA  69 28        ADC   #28        40 hozzáadása, egy sor
E8AC  CA           DEX
E8AD  D0 F6        BNE   E8A5
E8AF  60           RTS
E8B0  C6 D6        DEC   D6
E8B2  60           RTS
E8B3  A2 02        LDX   #02
E8B5  A9 27        LDA   #27        39, az utolsó oszlop
E8B7  C5 D3        CMP   D3
E8B9  F0 07        BEQ   E8C2
E8BB  18           CLC
E8BC  69 28        ADC   #28        40 hozzáadása
E8BE  CA           DEX
E8BF  D0 F6        BNE   E8B7
E8C1  60           RTS
E8C2  A6 D6        LDX   D6
E8C4  E0 19        CPX   #19        25
E8C6  F0 02        BEQ   E8CA
E8C8  E6 D6        INC   D6
E8CA  60           RTS

*****************************       a színkód vizsgálata
E8CB  A2 0F        LDX   #0F        a kódok keresése
E8CD  DD DA E8     CMP   E8DA,X     a táblázatban
E8D0  F0 04        BEQ   E8D6       sikerült megtalálni
E8D2  CA           DEX
E8D3  10 F8        BPL   E8CD
E8D5  60           RTS
E8D6  8E 86 02     STX   0286       a színkód beállítása
E8D9  60           RTS

*****************************       a színkódok táblázata
E8DA  90 05 1C 9F 9C 1E 1F 9E
E8E2  81 95 96 97 98 99 9A 9B

*****************************       a képernyő görgetése
E8EA  A5 AC        LDA   AC
E8EC  48           PHA
E8ED  A5 AD        LDA   AD
E8EF  48           PHA              a mutató tárolása
E8F0  A5 AE        LDA   AE
E8F2  48           PHA
E8F3  A5 AF        LDA   AF
```

```
E8F5  48          PHA
E8F6  A2 FF       LDX  #FF        kezdés a nulla sortól
E8F8  C6 D6       DEC  D6         a sorszám csökkentése
E8FA  C6 C9       DEC  C9
E8FC  CE A5 02    DEC  02A5
E8FF  E8          INX             a sorszám növelése
E900  20 F0 E9    JSR  E9F0       az X. sor mutatója a színramban
E903  E0 18       CPX  #18        24
E905  B0 0C       BCS  E913       már az összes sor?
E907  BD F1 EC    LDA  ECF1,X     az LSB beolvasása
E90A  85 AC       STA  AC
E90C  B5 DA       LDA  DA,X       MSB
E90E  20 C8 E9    JSR  E9C8       a sor görgetése felfelé
E911  30 EC       BMI  E8FF       következő sor
E913  20 FF E9    JSR  E9FF       a legalsó sor törlése
E916  A2 00       LDX  #00
E918  B5 D9       LDA  D9,X
E91A  29 7F       AND  #7F
E91C  B4 DA       LDY  DA,X
E91E  10 02       BPL  E922
E920  09 80       ORA  #80
E922  95 D9       STA  D9,X
E924  E8          INX
E925  E0 18       CPX  #18        24
E927  D0 EF       BNE  E918
E929  A5 F1       LDA  F1
E92B  09 80       ORA  #80
E92D  85 F1       STA  F1
E92F  A5 D9       LDA  D9
E931  10 C3       BPL  E8F6
E933  E6 D6       INC  D6
E935  EE A5 02    INC  02A5
E938  A9 7F       LDA  #7F
E93A  8D 00 DC    STA  DC00
E93D  AD 01 DC    LDA  DC01
E940  C9 FB       CMP  #FB        a CTRL-billentyű le van nyomva?
E942  08          PHP
E943  A9 7F       LDA  #7F
E945  8D 00 DC    STA  DC00
E948  28          PLP
E949  D0 0B       BNE  E956       nincs lenyomva
E94B  A0 00       LDY  #00
E94D  EA          NOP
E94E  CA          DEX
E94F  D0 FC       BNE  E94D       a késleltető ciklus
E951  88          DEY
E952  D0 F9       BNE  E94D
E954  84 C6       STY  C6         a lenyomott billentyűk száma egyenlő nullával
E956  A6 D6       LDX  D6
E958  68          PLA
E959  85 AF       STA  AF
E95B  68          PLA
E95C  85 AE       STA  AE         a mutató visszaolvasása
E95E  68          PLA
E95F  85 AD       STA  AD
E961  68          PLA
```



```
E962  85 AC       STA  AC
E964  60          RTS

*****************************     egy sor beszúrása
E965  A6 D6       LDX  D6         sorszám
E967  E8          INX
E968  B5 D9       LDA  D9,X
E96A  10 FB       BPL  E967
E96C  8E A5 02    STX  02A5
E96F  E0 18       CPX  #18        24
E971  F0 0E       BEQ  E981
E973  90 0C       BCC  E981
E975  20 EA E8    JSR  E8EA       a képernyő görgetése
E978  AE A5 02    LDX  02A5
E97B  CA          DEX
E97C  C6 D6       DEC  D6         a sorszám csökkentése
E97E  4C DA E6    JMP  E6DA

*****************************
E981  A5 AC       LDA  AC
E983  48          PHA
E984  A5 AD       LDA  AD
E986  48          PHA
E987  A5 AE       LDA  AE
E989  48          PHA
E98A  A5 AF       LDA  AF
E98C  48          PHA
E98D  A2 19       LDX  #19        25
E98F  CA          DEX             a sorszám
E990  20 F0 E9    JSR  E9F0       a színram-mutató kiszámítása
E993  EC A5 02    CPX  02A5
E996  90 0E       BCC  E9A6
E998  F0 0C       BEQ  E9A6
E99A  BD EF EC    LDA  ECEF,X     a sorkezdet LSB-jének beállítása
E99D  85 AC       STA  AC
E99F  B5 D8       LDA  D8,X       az LSB beállítása
E9A1  20 C8 E9    JSR  E9C8       a sor görgetése
E9A4  30 E9       BMI  E98F
E9A6  20 FF E9    JSR  E9FF       a sor törlése
E9A9  A2 17       LDX  #17        23
E9AB  EC A5 02    CPX  02A5
E9AE  90 0F       BCC  E9BF
E9B0  B5 DA       LDA  DA,X
E9B2  29 7F       AND  #7F
E9B4  B4 D9       LDY  D9,X
E9B6  10 02       BPL  E9BA
E9B8  09 80       ORA  #80
E9BA  95 DA       STA  DA,X
E9BC  CA          DEX
E9BD  D0 EC       BNE  E9AB
E9BF  AE A5 02    LDX  02A5
E9C2  20 DA E6    JSR  E6DA       az MSB újraszámítása
E9C5  4C 58 E9    JMP  E958       regiszter visszaolvasása,RTS

*****************************     a sor görgetése felfelé
E9C8  29 03       AND  #03
```

```
E9CA  0D 88 02     ORA   0288        az új sor mutatója
E9CD  85 AD        STA   AD
E9CF  20 E0 E9     JSR   E9E0        az új sor mutatójának kiszámítása
E9D2  A0 27        LDY   #27         39 karakter
E9D4  B1 AC        LDA   (AC),Y
E9D6  91 D1        STA   (D1),Y      a karakter
E9D8  B1 AE        LDA   (AE),Y
E9DA  91 F3        STA   (F3),Y      és a szín átvitele
E9DC  88           DEY
E9DD  10 F5        BPL   E9D4        az összes oszlop?
E9DF  60           RTS

*****************************
E9E0  20 24 EA     JSR   EA24        a görgetendő sor kiszámítása
E9E3  A5 AC        LDA   AC          a színram mutatójának kiszámítása
E9E5  85 AE        STA   AE
E9E7  A5 AD        LDA   AD
E9E9  29 03        AND   #03         a mutató a $AE/$AF-en
E9EB  09 D8        ORA   #D8
E9ED  85 AF        STA   AF
E9EF  60           RTS

*****************************
E9F0  BD F0 EC     LDA   ECF0,X      az X. sor videoram mutatója
E9F3  85 D1        STA   D1          LSB behozatala
E9F5  B5 D9        LDA   D9,X        MSB
E9F7  29 03        AND   #03
E9F9  0D 88 02     ORA   0288        a videoram felső byte-ja
E9FC  85 D2        STA   D2
E9FE  60           RTS

*****************************
E9FF  A0 27        LDY   #27         az X. sor törlése
EA01  20 F0 E9     JSR   E9F0        39 oszlop
EA04  20 24 EA     JSR   EA24        a videoram mutatójának beállítása
EA07  20 DA E4     JSR   E4DA        a színram mutatójának beállítása
EA0A  A9 20        LDA   #20         az üres karakter
EA0C  91 D1        STA   (D1),Y      beállítása
EA0E  88           DEY               a háttérszín beállítása
EA0F  10 F6        BPL   EA07
EA11  60           RTS               már a 0. oszlop?
EA12  EA           NOP

*****************************
EA13  A8           TAY
EA14  A9 02        LDA   #02
EA16  85 CD        STA   CD          ism.funkciónál villogás-számláló beállítása
EA18  20 24 EA     JSR   EA24        színram mutatójának kiszámítása
EA1B  98           TYA

*****************************
EA1C  A4 D3        LDY   D3          a karakter és a szín beáll. a képernyőn
EA1E  91 D1        STA   (D1),Y      az oszlop pozíció
EA20  8A           TXA               az X-beli színkód
EA21  91 F3        STA   (F3),Y      beírása a SZÍNRAM-ba
EA23  60           RTS
```



```
*****************************
EA24  A5 D1        LDA   D1          a színram mutatójának kiszámítása
EA26  85 F3        STA   F3          $D1/$D2="videoram pozíció" mutató
EA28  A5 D2        LDA   D2
EA2A  29 03        AND   #03
EA2C  09 D8        ORA   #D8         a felső byte =$D8
EA2E  85 F4        STA   F4          $F3/$F4="színram pozíció" mutató
EA30  60           RTS

*****************************
EA31  20 EA FF     JSR   FFEA        az interrupt rutin /megszakítás/
EA34  A5 CC        LDA   CC          a stop billentyű, az idő növelése
EA36  D0 29        BNE   EA61        a kurzorvillogás-kapcsolója
EA38  C6 CD        DEC   CD          ha nem villog, akkor tovább
EA3A  D0 25        BNE   EA61        a villogásszámláló állásának csökkentése
EA3C  A9 14        LDA   #14         ha nem nulla, akkor tovább
EA3E  85 CD        STA   CD          a villogásszámláló visszaállítása 20-ra
EA40  A4 D3        LDY   D3          a kurzor oszlopa
EA42  46 CF        LSR   CF          ha a villogáskapcsoló nulla, akkor C=1
EA44  AE 87 02     LDX   0287        a kurzor alatti szín
EA47  B1 D1        LDA   (D1),Y      a karakterkód beállítása
EA49  B0 11        BCS   EA5C        ha a villogáskapcs. egy volt, akkor tovább
EA4B  E6 CF        INC   CF          a villogáskapcsoló be
EA4D  85 CE        STA   CE          a kurzor alatti karakter tárolása
EA4F  20 24 EA     JSR   EA24        a színram mutató kiszámítása
EA52  B1 F3        LDA   (F3),Y      a színkód beolvasása
EA54  8D 87 02     STA   0287        és tárolása
EA57  AE 86 02     LDX   0286        a kurzor alatti színkód
EA5A  A5 CE        LDA   CE          a kurzor alatti karakter
EA5C  49 80        EOR   #80         az RVS-bit megfordítása
EA5E  20 1C EA     JSR   EA1C        a karakter és a szín beállítása
EA61  A5 01        LDA   01
EA63  29 10        AND   #10         a rekord-billentyű ellenőrzése
EA65  F0 0A        BEQ   EA71        le van nyomva?
EA67  A0 00        LDY   #00         a rekordkapcsoló beállítása
EA69  84 C0        STY   C0
EA6B  A5 01        LDA   01          a motor ki
EA6D  09 20        ORA   #20
EA6F  D0 08        BNE   EA79        a rekordkapcsoló
EA71  A5 C0        LDA   C0
EA73  D0 06        BNE   EA7B
EA75  A5 01        LDA   01          a motor be
EA77  29 1F        AND   #1F
EA79  85 01        STA   01
EA7B  20 87 EA     JSR   EA87        a billentyűzet lekérdezése
EA7E  AD 0D DC     LDA   DC0D        az IRQ-kapcsoló törlése
EA81  68           PLA
EA82  A8           TAY               a regiszter visszaállítása
EA83  68           PLA
EA84  AA           TAX
EA85  68           PLA               és visszatérés a megszakításból
EA86  40           RTI

*****************************
EA87  A9 00        LDA   #00         a billentyűzet lekérdezése
```

```
EA89  8D 8D 02     STA   028D      shift/CTRL-kapcsoló visszaállítása
EA8C  A0 40        LDY  #40        $40= nincs billentyű lenyomva
EA8E  84 CB        STY   CB        a lenyomott billentyű kódja
EA90  8D 00 DC     STA   DC00      a mátrixsorok vizsgálata
EA93  AE 01 DC     LDX   DC01
EA96  E0 FF        CPX  #FF        nincs billentyű lenyomva?
EA98  F0 61        BEQ   EAFB      ha nincs, akkor kész
EA9A  A8           TAY
EA9B  A9 81        LDA  #81
EA9D  85 F5        STA   F5
EA9F  A9 EB        LDA  #EB
EAA1  85 F6        STA   F6        az 1. táblázat mutatója $EB81
EAA3  A9 FE        LDA  #FE
EAA5  8D 00 DC     STA   DC00
EAA8  A2 08        LDX  #08        8 mátrixsor
EAAA  48           PHA
EAAB  AD 01 DC     LDA   DC01
EAAE  CD 01 DC     CMP   DC01      a billentyűzet kioldása
EAB1  D0 F8        BNE   EAAB
EAB3  4A           LSR   A         a bitek egymásutáni betolása a carry-be
EAB4  B0 16        BCS   EACC      ha "1", akkor lenyomva
EAB6  48           PHA
EAB7  B1 F5        LDA  (F5),Y     az ASCII-kód beolvasása a táblázatból
EAB9  C9 05        CMP  #05        5-tel egyenlő vagy ennél nagyobb?
EABB  B0 0C        BCS   EAC9
EABD  C9 03        CMP  #03
EABF  F0 08        BEQ   EAC9      "STOP"-kód?
EAC1  0D 8D 02     ORA   028D
EAC4  8D 8D 02     STA   028D      kapcsoló beállítása
EAC7  10 02        BPL   EACB
EAC9  84 CB        STY   CB        billentyű sorszámának tárolása
EACB  68           PLA
EACC  C8           INY
EACD  C0 41        CPY  #41
EACF  B0 0B        BCS   EADC      nagyobb $40-nél?
EAD1  CA           DEX
EAD2  D0 DF        BNE   EAB3      a következő matrix oszlop
EAD4  38           SEC
EAD5  68           PLA
EAD6  2A           ROL   A
EAD7  8D 00 DC     STA   DC00
EADA  D0 CC        BNE   EAA8      a következő mátrixsor
EADC  68           PLA
EADD  6C 8F 02     JMP  (028F)     a JMP $EB48 a mutatót a táblázatra állítja
EAE0  A4 CB        LDY   CB        a billentyű sorszáma
EAE2  B1 F5        LDA  (F5),Y     az ASCII-kód beolvasása a táblázatból
EAE4  AA           TAX
EAE5  C4 C5        CPY   C5        összehasonlítás az utolsó billentyűvel
EAE7  F0 07        BEQ   EAF0
EAE9  A0 10        LDY  #10
EAEB  8C 8C 02     STY   028C      repeat-késleltetés számláló
EAEE  D0 36        BNE   EB26
EAF0  29 7F        AND  #7F        a 7. bit törlése
EAF2  2C 8A 02     BIT   028A      az összes billentyű repeat-funkciója?
EAF5  30 16        BMI   EB0D      ha a 7.bit magas, az összes billentyű ismétl.
EAF7  70 49        BVS   EB42      ha a 6.bit magas, figyelmen kívül hagyni
```



```
EAF9  C9 7F        CMP  #7F        csak a következő billentyűk ismétlése
EAFB  F0 29        BEQ   EB26
EAFD  C9 14        CMP  #14        "DEL", "INST" kód
EAFF  F0 0C        BEQ   EB0D
EB01  C9 20        CMP  #20        az üres karakter
EB03  F0 08        BEQ   EB0D
EB05  C9 1D        CMP  #1D        cursor right, left
EB07  F0 04        BEQ   EB0D
EB09  C9 11        CMP  #11        cursor down, up
EB0B  D0 35        BNE   EB42
EB0D  AC 8C 02     LDY   028C      a repeat-késleltetés számláló
EB10  F0 05        BEQ   EB17
EB12  CE 8C 02     DEC   028C      visszaszámlálás
EB15  D0 2B        BNE   EB42
EB17  CE 8B 02     DEC   028B      a repeat-sebesség számláló
EB1A  D0 26        BNE   EB42
EB1C  A0 04        LDY  #04
EB1E  8C 8B 02     STY   028B      a számláló visszaállítása
EB21  A4 C6        LDY   C6        a karakterek száma a billentyűzet-pufferban
EB23  88           DEY
EB24  10 1C        BPL   EB42      ha 1-nél több karakter, figyelmen kívül hagy.
EB26  A4 CB        LDY   CB
EB28  84 C5        STY   C5
EB2A  AC 8D 02     LDY   028D
EB2D  8C 8E 02     STY   028E
EB30  E0 FF        CPX  #FF        érvénytelen a billentyűzet-kód?
EB32  F0 0E        BEQ   EB42      ha igen, figyelmen kívül hagyni
EB34  8A           TXA
EB35  A6 C6        LDX   C6        a karakterek száma a billentyűzet-pufferban
EB37  EC 89 02     CPX   0289      a karakterek számának összehas. max.számmal
EB3A  B0 06        BCS   EB42      ha a puffer megtelt,karaktereket figy.kívül
EB3C  9D 77 02     STA   0277,X    karakterek beírása a billentyűzet-pufferba
EB3F  E8           INX
EB40  86 C6        STX   C6        karakterek számának növelése
EB42  A9 7F        LDA  #7F        bill.-mátrix lekérdezése, hibavizsgálat
EB44  8D 00 DC     STA   DC00
EB47  60           RTS

*****************************
EB48  AD 8D 02     LDA   028D      shift-, CTRL- és Commodore-billentyűk vizsg.
EB4B  C9 03        CMP  #03        a shift/CTRL-kapcsoló
EB4D  D0 15        BNE   EB64      dekód.-táblázat mutatójának kiszámítása
EB4F  CD 8E 02     CMP   028E
EB52  F0 EE        BEQ   EB42
EB54  AD 91 02     LDA   0291      engedélyezett a shift-commodore?
EB57  30 1D        BMI   EB76      ha nem,vissza a dekódoláshoz
EB59  AD 18 D0     LDA   D018      shift/Commodore
EB5C  49 02        EOR  #02        kisbetűs/nagybetűs átváltás
EB5E  8D 18 D0     STA   D018
EB61  4C 76 EB     JMP   EB76      kész
EB64  0A           ASL   A
EB65  C9 08        CMP  #08
EB67  90 02        BCC   EB6B
EB69  A9 06        LDA  #06
EB6B  AA           TAX
EB6C  BD 79 EB     LDA   EB79,X
```

```
EB6F  85 F5        STA   F5          bill.dekódolási táblázatának mutatója
EB71  BD 7A EB     LDA   EB7A,X
EB74  85 F6        STA   F6
EB76  4C E0 EA     JMP   EAE0        vissza a dekódoláshoz

*****************************
EB79  81 EB C2 EB 03 EC 78 EC       bill. dekódolási táblázatának mutatója

*****************************
EB81  14 0D 1D 88 85 86 87 11       bill. 1.dekód.tábl. shiftelés nélkül
EB89  33 57 41 34 5A 53 45 01
EB91  35 52 44 36 43 46 54 58
EB99  37 59 47 38 42 48 55 56
EBA1  39 49 4A 30 4D 4B 4F 4E
EBA9  2B 50 4C 2D 2E 3A 40 2C
EBB1  5C 2A 3B 13 01 3D 5E 2F
EBB9  31 5F 04 32 20 02 51 03
EBC1  FF

*****************************
EBC2  94 8D 9D 8C 89 8A 8B 91       bill. 2.dekód.táblázata, shiftelve
EBCA  23 D7 C1 24 DA D3 C5 01
EBD2  25 D2 C4 26 C3 C6 D4 D8
EBDA  27 D9 C7 28 C2 C8 D5 D6
EBE2  29 C9 CA 30 CD CB CF CE
EBEA  DB D0 CC DD 3E 5B BA 3C
EBF2  A9 C0 5D 93 01 3D DE 3F
EBFA  21 5F 04 22 A0 02 D1 83
EC02  FF

*****************************
EC03  94 8D 9D 8C 89 8A 8B 91       bill. 3.dekód.tábl.,"C="-billentyűvel
EC0B  96 B3 B0 97 AD AE B1 01
EC13  98 B2 AC 99 BC BB A3 BD
EC1B  9A B7 A5 9B BF B4 B8 BE
EC23  29 A2 B5 30 A7 A1 B9 AA
EC2B  A6 AF B6 DC 3E 5B A4 3C
EC33  A8 DF 5D 93 01 3D DE 3F
EC3B  81 5F 04 95 A0 02 AB 83
EC43  FF

*****************************
EC44  C9 0E        CMP   #0E          a vezérlőkarakterek  vizsgálata
EC46  D0 07        BNE   EC4F         chr$(14)
EC48  AD 18 D0     LDA   D018
EC4B  09 02        ORA   #02          a karaktergenerátor
EC4D  D0 09        BNE   EC58         nagybetűs mód
EC4F  C9 8E        CMP   #8E          chr$(142)
EC51  D0 0B        BNE   EC5E
EC53  AD 18 D0     LDA   D018         kisbetűs mód
EC56  29 FD        AND   #FD          beállítása
EC58  8D 18 D0     STA   D018
EC5B  4C A8 E6     JMP   E6A8         chr$(8)
EC5E  C9 08        CMP   #08
EC60  D0 07        BNE   EC69         shift-Commodore letiltása
EC62  A9 80        LDA   #80
EC64  0D 91 02     ORA   0291         chr$(9)
EC67  30 09        BMI   EC72
EC69  C9 09        CMP   #09
```



```
EC6B  D0 EE        BNE   EC5B
EC6D  A9 7F        LDA   #7F
EC6F  2D 91 02     AND   0291         shift-Commodore engedélyezése
EC72  8D 91 02     STA   0291
EC75  4C A8 E6     JMP   E6A8

*****************************
EC78  FF FF FF FF FF FF FF FF       a bill.4.dekód.tábl. CTRL-billentyűvel
EC80  1C 17 01 9F 1A 13 05 FF
EC88  9C 12 04 1E 03 06 14 18
EC90  1F 19 07 9E 02 08 15 16
EC98  12 09 0A 92 0D 0B 0F 0E
ECA0  FF 10 0C FF FF 1B 00 FF
ECA8  1C FF 1D FF FF 1F 1E FF
ECB0  90 06 FF 05 FF FF 11 FF

*****************************
ECB9  00 00 00 00 00 00 00 00       a viedovezérlő konstansai
ECC1  00 00 00 00 00 00 00 00
ECC9  00 9B 37 00 00 00 08 00
ECD1  14 0F 00 00 00 00 00 00
ECD9  0E 06 01 02 03 04 00 01
ECE1  02 03 04 05 06 07

*****************************
ECE7  4C 4F 41 44 0D               szöveg a SHIFT+RUN/STOP lenyomása után
ECEC  52 55 4E 0D                  "load /cr/ run /cr"

*****************************
ECF0  00 28 50 78 A0 C8 F0 18       a képernyősor-kezdet LSB-táblázata
ECF8  40 68 90 B8 E0 08 30 58
ED00  80 A8 D0 F8 20 48 70 98
ED08  C0

*****************************
ED09  09 40        ORA   #40       az IEC-busz rutinok, TALK küldése
ED0B  2C                           a TALK bit beállítása

*****************************
ED0C  09 20        ORA   #20       LISTEN küldése
ED0E  20 A4 F0     JSR   F0A4      a LISTEN bit beállítása
ED11  48           PHA             az IEC timer "time-out" beállítása
ED12  24 94        BIT   94
ED14  10 0A        BPL   ED20      még egy byte kiírása?
ED16  38           SEC             nem
ED17  66 A3        ROR   A3
ED19  20 40 ED     JSR   ED40      a byte továbbítása az IEC-buszra
ED1C  46 94        LSR   94
ED1E  46 A3        LSR   A3
ED20  68           PLA
ED21  85 95        STA   95        a kiírandó byte
ED23  78           SEI
ED24  20 97 EE     JSR   EE97      DATA hI
ED27  C9 3F        CMP   #3F
ED29  D0 03        BNE   ED2E
```

```
ED2B  20 85 EE    JSR   EE85        CLOCK hI
ED2E  AD 00 DD    LDA   DD00
ED31  09 08       ORA   #08         az ATN LO beállítása
ED33  8D 00 DD    STA   DD00
ED36  78          SEI
ED37  20 8E EE    JSR   EE8E        CLOCk LO
ED3A  20 97 EE    JSR   EE97        DATA hI
ED3D  20 B3 EE    JSR   EEB3        várakozás 1  ms.-ig

******************************    egy byte továbbítása az IEC-buszra
ED40  78          SEI
ED41  20 97 EE    JSR   EE97        DATA hI
ED44  20 A9 EE    JSR   EEA9        DATA a carry-be
ED47  B0 64       BCS   EDAD        ha DATA LO, "device not present"
ED49  20 85 EE    JSR   EE85        CLOCk hI
ED4C  24 A3       BIT   A3
ED4E  10 0A       BPL   ED5A
ED50  20 A9 EE    JSR   EEA9        DATA a carry-be
ED53  90 FB       BCC   ED50        várakozás a DATA LO-ra
ED55  20 A9 EE    JSR   EEA9        DATA a carry-be
ED58  B0 FB       BCS   ED55        várakozás a DATA hI-re
ED5A  20 A9 EE    JSR   EEA9        DATA a carry-be
ED5D  90 FB       BCC   ED5A        várakozás a DATA LO-ra
ED5F  20 8E EE    JSR   EE8E        CLOCK hI
ED62  A9 08       LDA   #08
ED64  85 A5       STA   A5          a soros átv.bit-számlálójának beállítása
ED66  AD 00 DD    LDA   DD00
ED69  CD 00 DD    CMP   DD00
ED6C  D0 F8       BNE   ED66
ED6E  0A          ASL   A           DATA a carry-be
ED6F  90 3F       BCC   EDB0        ha DATA hI, akkor "time out"
ED71  66 95       ROR   95          a következő bit előkészítése
ED73  B0 05       BCS   ED7A        a bit magas !
ED75  20 A0 EE    JSR   EEA0        nem, DATA LO kiírása
ED78  D0 03       BNE   ED7D
ED7A  20 97 EE    JSR   EE97        DATA hI kiírása
ED7D  20 85 EE    JSR   EE85        CLOCK HI
ED80  EA          NOP
ED81  EA          NOP
ED82  EA          NOP
ED83  EA          NOP
ED84  AD 00 DD    LDA   DD00
ED87  29 DF       AND   #DF         DATA hI
ED89  09 10       ORA   #10         és CLOCK LO
ED8B  8D 00 DD    STA   DD00        kiírás
ED8E  C6 A5       DEC   A5          a következő bit
ED90  D0 D4       BNE   ED66
ED92  A9 04       LDA   #04
ED94  8D 07 DC    STA   DC07        a timer hi, kb. 1 ms-ig
ED97  A9 19       LDA   #19
ED99  8D 0F DC    STA   DC0F        a timer indítása
ED9C  AD 0D DC    LDA   DC0D
ED9F  AD 0D DC    LDA   DC0D
EDA2  29 02       AND   #02         a timer lefutott?
EDA4  D0 0A       BNE   EDB0        ha igen, "time out"
EDA6  20 A9 EE    JSR   EEA9        DATA a carry-be
```



```
EDA9  B0 F4       BCS   ED9F        várakozás a DATA hI-re
EDAB  58          CLI
EDAC  60          RTS

******************************
EDAD  A9 80       LDA   #80         "device not present"
EDAF  2C
EDB0  A9 03       LDA   #03         "time out"
EDB2  20 1C FE    JSR   FE1C        a státusz beállítása
EDB5  58          CLI
EDB6  18          CLC
EDB7  90 4A       BCC   EE03

******************************
EDB9  85 95       STA   95          a másodlagos cím küldése LISTEN
EDBB  20 36 ED    JSR   ED36        a másodlagos cím tárolása
EDBE  AD 00 DD    LDA   DD00        kiírás ATN LO-vel
EDC1  29 F7       AND   #F7         az ATN visszaállítása, hI
EDC3  8D 00 DD    STA   DD00
EDC6  60          RTS

******************************
EDC7  85 95       STA   95          a másodlagos cím továbbítása TALK
EDC9  20 36 ED    JSR   ED36        a másodlagos cím tárolása
EDCC  78          SEI               kiírás az ATN-nel
EDCD  20 A0 EE    JSR   EEA0        DATA LO
EDD0  20 BE ED    JSR   EDBE        az ATN visszaállítása, hI
EDD3  20 85 EE    JSR   EE85        CLOCK hI
EDD6  20 A9 EE    JSR   EEA9        az adatbit beolvasása
EDD9  30 FB       BMI   EDD6        várakozás a DATA hI-re
EDDB  58          CLI
EDDC  60          RTS

******************************
EDDD  24 94       BIT   94          IECOUT - egy byte tov. az IEC-buszhoz
EDDF  30 05       BMI   EDE6        van még kiírandó byte?
EDE1  38          SEC               igen
EDE2  66 94       ROR   94          a kapcsoló beállítása
EDE4  D0 05       BNE   EDEB        feltétlen ugrás
EDE6  48          PHA               a byte tárolása
EDE7  20 40 ED    JSR   ED40        a byte továbbítása a buszhoz
EDEA  68          PLA               a byte visszaolvasása
EDEB  85 95       STA   95          és betöltése a kiviteli reg.-be
EDED  18          CLC
EDEE  60          RTS

******************************
EDEF  78          SEI               UNTALK küldése
EDF0  20 8E EE    JSR   EE8E        CLOCK LO
EDF3  AD 00 DD    LDA   DD00
EDF6  09 08       ORA   #08         az ATN LO beállítása
EDF8  8D 00 DD    STA   DD00
EDFB  A9 5F       LDA   #5F
EDFD  2C

******************************    UNLISTEN küldése
```

```
EDFE  A9 3F       LDA  #3F
EE00  20 11 ED    JSR  ED11      kiírás
EE03  20 BE ED    JSR  EDBE      az ATN visszaállítása, HI
EE06  8A          TXA
EE07  A2 0A       LDX  #0A
EE09  CA          DEX            kb. 40 . ms.   . kivárása
EE0A  D0 FD       BNE  EE09
EE0C  AA          TAX
EE0D  20 85 EE    JSR  EE85      CLOCK HI
EE10  4C 97 EE    JMP  EE97      DATA HI

*****************************    IECIN - egy karakter beolvasása IEC buszról
EE13  78          SEI
EE14  A9 00       LDA  #00
EE16  85 A5       STA  A5
EE18  20 85 EE    JSR  EE85      CLOCK HI
EE1B  20 A9 EE    JSR  EEA9      az adat beolvasása
EE1E  10 FB       BPL  EE1B      várakozás a DATA HI-re
EE20  A9 01       LDA  #01
EE22  8D 07 DC    STA  DC07      timer hi
EE25  A9 19       LDA  #19
EE27  8D 0F DC    STA  DC0F      a timer indítása
EE2A  20 97 EE    JSR  EE97      DATA HI
EE2D  AD 0D DC    LDA  DC0D
EE30  AD 0D DC    LDA  DC0D      timer
EE33  29 02       AND  #02       lefutott?
EE35  D0 07       BNE  EE3E      igen, "time out"
EE37  20 A9 EE    JSR  EEA9      egy adatbit beolvasása
EE3A  30 F4       BMI  EE30      várakozás a DATA LO-ra
EE3C  10 18       BPL  EE56
EE3E  A5 A5       LDA  A5
EE40  F0 05       BEQ  EE47
EE42  A9 02       LDA  #02       "time out"
EE44  4C B2 ED    JMP  EDB2      a státusz beállítása
EE47  20 A0 EE    JSR  EEA0      DATA LO
EE4A  20 85 EE    JSR  EE85      CLOCK HI
EE4D  A9 40       LDA  #40       "EOF"
EE4F  20 1C FE    JSR  FE1C      a státusz beállítása
EE52  E6 A5       INC  A5
EE54  D0 CA       BNE  EE20
EE56  A9 08       LDA  #08
EE58  85 A5       STA  A5        a bitszámláló beállítása
EE5A  AD 00 DD    LDA  DD00      az adatbit beolvasása
EE5D  CD 00 DD    CMP  DD00
EE60  D0 F8       BNE  EE5A
EE62  0A          ASL  A         áttolás a carry-be
EE63  10 F5       BPL  EE5A      a bit "0"
EE65  66 A4       ROR  A4        nem
EE67  AD 00 DD    LDA  DD00
EE6A  CD 00 DD    CMP  DD00      adatok beolvasása
EE6D  D0 F8       BNE  EE67
EE6F  0A          ASL  A
EE70  30 F5       BMI  EE67
EE72  C6 A5       DEC  A5        a következő bit
EE74  D0 E4       BNE  EE5A
EE76  20 A0 EE    JSR  EEA0      DATA LO
```



```
EE79  24 90       BIT  90        a státusz
EE7B  50 03       BVC  EE80      nem EOF?
EE7D  20 06 EE    JSR  EE06      várakozás és az "1" bit küldése
EE80  A5 A4       LDA  A4
EE82  58          CLI
EE83  18          CLC
EE84  60          RTS

*****************************
EE85  AD 00 DD    LDA  DD00      CLOCK HI
EE88  29 EF       AND  #EF       a 4. bit törlése
EE8A  8D 00 DD    STA  DD00
EE8D  60          RTS

*****************************
EE8E  AD 00 DD    LDA  DD00      CLOCK LO
EE91  09 10       ORA  #10       a 4. bit beállítása
EE93  8D 00 DD    STA  DD00
EE96  60          RTS

*****************************
EE97  AD 00 DD    LDA  DD00      DATA HI
EE9A  29 DF       AND  #DF       az 5. bit törlése
EE9C  8D 00 DD    STA  DD00
EE9F  60          RTS

*****************************
EEA0  AD 00 DD    LDA  DD00      DATA LO
EEA3  09 20       ORA  #20       az 5. bit beállítása
EEA5  8D 00 DD    STA  DD00
EEA8  60          RTS

*****************************
EEA9  AD 00 DD    LDA  DD00      bit beolv.IEC buszról a carry kapcsolóba
EEAC  CD 00 DD    CMP  DD00      adatregiszter
EEAF  D0 F8       BNE  EEA9      az adatbit léptetése a carry-be
EEB1  0A          ASL  A
EEB2  60          RTS

*****************************
EEB3  8A          TXA            1 ms.   . késleltetése
EEB4  A2 B8       LDX  #B8       184
EEB6  CA          DEX
EEB7  D0 FD       BNE  EEB6
EEB9  AA          TAX
EEBA  60          RTS

*****************************
EEBB  A5 B4       LDA  B4        RS232 output
EEBD  F0 47       BEQ  EF06      a kiírandó bitek száma
EEBF  30 3F       BMI  EF00      a byte továbbítva?
EEC1  46 B6       LSR  B6        stopbit?
EEC3  A2 00       LDX  #00       a köv. bit a carry-be
EEC5  90 01       BCC  EEC8
EEC7  CA          DEX            a bit törülve?
EEC8  8A          TXA            ha nem, akkor $FF
```

```
EEC9  45 BD      EOR  BD      összekapcsolás a paritással
EECB  85 BD      STA  BD      és tárolás
EECD  C6 B4      DEC  B4      a bitszámláló csökkentése
EECF  F0 06      BEQ  EED7    minden bit továbbítva?
EED1  8A         TXA
EED2  29 04      AND  #04     a 2. bit leválasztása
EED4  85 B5      STA  B5      és tárolása a kiviteli reg-ben
EED6  60         RTS
-------------------------------------------------------------
EED7  A9 20      LDA  #20     5. bit /paritás/
EED9  2C 94 02   BIT  0294    az RS232 utasításregisztere
EEDC  F0 14      BEQ  EEF2    paritás nélkül?
EEDE  30 1C      BMI  EEFC    rögzített paritás?
EEE0  70 14      BVS  EEF6    a paritás páratlan?
EEE2  A5 BD      LDA  BD      a paritás 1?
EEE4  D0 01      BNE  EEE7    igen
EEE6  CA         DEX          a paritás $FF
EEE7  C6 B4      DEC  B4      a bitszámláló $FF
EEE9  AD 93 02   LDA  0293    RS232 vezérlőregiszter
EEEC  10 E3      BPL  EED1    két stopbit?
EEEE  C6 B4      DEC  B4      a bitszámláló $FE
EEF0  D0 DF      BNE  EED1    a stopbitek számításához!
EEF2  E6 B4      INC  B4      a bitszámláló növelése, paritás nélkül
EEF4  D0 F0      BNE  EEE6    a stopbitek számításához
EEF6  A5 BD      LDA  BD      paritás
EEF8  F0 ED      BEQ  EEE7    ha $0, akkor a $0. bit kiírása
EEFA  D0 EA      BNE  EEE6    egyébként az 1. bit kiírása
EEFC  70 E9      BVS  EEE7    a $0. bit kiírása
EEFE  50 E6      BVC  EEE6    egyébként az 1. bit kiírása /rögz. paritás/
-------------------------------------------------------------
EF00  E6 B4      INC  B4      a bitszámláló növelése
EF02  A2 FF      LDX  #FF     stopbit
EF04  D0 CB      BNE  EED1    feltétlen ugrás
EF06  AD 94 02   LDA  0294    az RS232 utasításregiszter
EF09  4A         LSR  A       $0. bit a carry-be
EF0A  90 07      BCC  EF13    a handshake lekérdezése
EF0C  2C 01 DD   BIT  DD01
EF0F  10 1D      BPL  EF2E    hiányzik a DSR?
EF11  50 1E      BVC  EF31    hiányzik a CTS?
EF13  A9 00      LDA  #00
EF15  85 BD      STA  BD      a paritásbit törlése
EF17  85 B5      STA  B5      a küldendő bitek regisztere
EF19  AE 98 02   LDX  0298    a bitek száma
EF1C  86 B4      STX  B4      bitszámláló
EF1E  AC 9D 02   LDY  029D    összes byte továbbítva?
EF21  CC 9E 02   CPY  029E
EF24  F0 13      BEQ  EF39    az átvitel vége, lezárás
EF26  B1 F9      LDA  (F9),Y  adat-byte az RS232-pufferből
EF28  85 B6      STA  B6      továbbítás
EF2A  EE 9D 02   INC  029D    a puffer-mutató növelése
EF2D  60         RTS
-------------------------------------------------------------
EF2E  A9 40      LDA  #40     nincs DSR /data set ready/
EF30  2C
EF31  A9 10      LDA  #10     nincs CTS /clear to send/
EF33  0D 97 02   ORA  0297    a státusz beállítása
EF36  8D 97 02   STA  0297
EF39  A9 01      LDA  #01
EF3B  8D 0D DD   STA  DD0D    az A timer NMI-jének törlése
```



```
EF3E  4D A1 02   EOR  02A1
EF41  09 80      ORA  #80     RS232 kapcsoló megfordítása
EF43  8D A1 02   STA  02A1
EF46  8D 0D DD   STA  DD0D    a maradék NMI-k elhagyva
EF49  60         RTS

*****************************
EF4A  A2 09      LDX  #09     RS232 - adatbitek számának meghatározása
EF4C  A9 20      LDA  #20
EF4E  2C 93 02   BIT  0293    az ellenőrző szó
EF51  F0 01      BEQ  EF54    az 5. bit törölve?
EF53  CA         DEX
EF54  50 02      BVC  EF58    a 6. bit törölve
EF56  CA         DEX
EF57  CA         DEX          X=adatbitek száma
EF58  60         RTS
EF59  A6 A9      LDX  A9      a fogadott bit feldolgozása , startbit?
EF5B  D0 33      BNE  EF90    igen
EF5D  C6 A8      DEC  A8      a bitszámláló csökkentése
EF5F  F0 36      BEQ  EF97    minden bit feldolgozva?
EF61  30 0D      BMI  EF70    várakozás a stopbitre
EF63  A5 A7      LDA  A7      a fogadott bit
EF65  45 AB      EOR  AB      a paritás kiszámítása
EF67  85 AB      STA  AB
EF69  46 A7      LSR  A7      az érkező bit a carry-be
EF6B  66 AA      ROR  AA      és a fogadó regiszterbe
EF6D  60         RTS
-------------------------------------------------------------
EF6E  C6 A8      DEC  A8      a bitszámláló csökkentése
EF70  A5 A7      LDA  A7      stopbit
EF72  F0 67      BEQ  EFDB    egyenlő nullával?
EF74  AD 93 02   LDA  0293    ellenőrző szó
EF77  0A         ASL  A       a 7. bit a carry-be /stopbitek száma/
EF78  A9 01      LDA  #01
EF7A  65 A8      ADC  A8      bitek száma és a stopbitek összeadása
EF7C  D0 EF      BNE  EF6D    van még stopbit?
EF7E  A9 90      LDA  #90     RDA fogadása a "Flag"-en
EF80  8D 0D DD   STA  DD0D    az NMI felszabadítása
EF83  0D A1 02   ORA  02A1
EF86  8D A1 02   STA  02A1    az RS232 NMI kapcsolója
EF89  85 A9      STA  A9      a startbit kapcsolója
EF8B  A9 02      LDA  #02
EF8D  4C 3B EF   JMP  EF3B    "B" timer NMI törlése
-------------------------------------------------------------
EF90  A5 A7      LDA  A7      startbit
EF92  D0 EA      BNE  EF7E    nem egyenlő nullával?
EF94  4C D3 E4   JMP  E4D3    a startbit kapcsoló visszaállítása
-------------------------------------------------------------
EF97  AC 9B 02   LDY  029B    puffer-mutató növelése
EF9A  C8         INY
EF9B  CC 9C 02   CPY  029C    a fogadó puffer megtelt?
EF9E  F0 2A      BEQ  EFCA    ha igen, a státusz beállítása
EFA0  8C 9B 02   STY  029B    a puffer-mutató tárolása
EFA3  88         DEY
EFA4  A5 AA      LDA  AA      a fogadott byte
EFA6  AE 98 02   LDX  0298    az adatbitek száma
EFA9  E0 09      CPX  #09     8 bit és egy stopbit?
EFAB  F0 04      BEQ  EFB1    igen, ok
EFAD  4A         LSR  A       egyébként eltolás
```

```
EFAE  E8          INX
EFAF  D0 F8       BNE   EFA9
EFB1  91 F7       STA   (F7),Y      a byte beírása az RS232 pufferba
EFB3  A9 20       LDA   #20
EFB5  2C 94 02    BIT   0294        az 5. bit az utasításszóban
EFB8  F0 B4       BEQ   EF6E        átvitel paritás nélkül?
EFBA  30 B1       BMI   EF6D        rögzített bit?
EFBC  A5 A7       LDA   A7          fogadó paritásbit
EFBE  45 AB       EOR   AB          összehasonlítása a számítottal
EFC0  F0 03       BEQ   EFC5        egyenlő?
EFC2  70 A9       BVS   EF6D        páros paritás, ok
EFC4  2C 50 A6    BIT   A650        páratlan paritás, ok
EFC7  A9 01       LDA   #01         paritás-hiba
EFC9  2C A9 04    BIT   04A9        a fogadó-puffer megtelt
EFCC  2C A9 80    BIT   80A9        break-utasítás fogadása
EFCF  2C A9 02    BIT   02A9        kerethiba
EFD2  0D 97 02    ORA   0297
EFD5  8D 97 02    STA   0297        a státusz beállítása
EFD8  4C 7E EF    JMP   EF7E        a következő byte fogadásához!
EFDB  A5 AA       LDA   AA          a fogadott byte
EFDD  D0 F1       BNE   EFD0        kerethiba
EFDF  F0 EC       BEQ   EFCD        a break-utasítás fogadása

******************************
EFE1  85 9A       STA   9A          RS232 CHOUT, kiírás az RS232-esre
EFE3  AD 94 02    LDA   0294        az egységszám tárolása
EFE6  4A          LSR   A           RS232 utasításszó
EFE7  90 29       BCC   F012        Ø. bit
EFE9  A9 02       LDA   #02         3-as handshake vonal?
EFEB  2C 01 DD    BIT   DD01
EFEE  10 1D       BPL   F00D        DSR lekérdezése
EFF0  D0 20       BNE   F012        ha nem, akkor hiba
EFF2  AD A1 02    LDA   02A1        nem RTS?
EFF5  29 02       AND   #02         az NMI-kapcsoló
EFF7  D0 F9       BNE   EFF2        az adatfogadó aktív?
EFF9  2C 01 DD    BIT   DD01        várakozás a fogadás végéig
EFFC  70 FB       BVS   EFF9
EFFE  AD 01 DD    LDA   DD01        várakozás a CTS-re
F001  09 02       ORA   #02
F003  8D 01 DD    STA   DD01        a RTS beállítása
F006  2C 01 DD    BIT   DD01
F009  70 07       BVS   F012        várakozás a CTS-re
F00B  30 F9       BMI   F006        a DSR lekérdezése
F00D  A9 40       LDA   #40
F00F  8D 97 02    STA   0297        nincs DSR jel, a státusz beállítása
F012  18          CLC
F013  60          RTS

******************************
F014  20 28 F0    JSR   F028        kiírás az RS232 pufferba
F017  AC 9E 02    LDY   029E        az átvitel indítása
F01A  C8          INY               a kiviteli puffer írásmutató
F01B  CC 9D 02    CPY   029D        növelése
F01E  F0 F4       BEQ   F014        összehasonlítása az olvasómutatóval
F020  8C 9E 02    STY   029E        ha a puffer megtelt, várakozás
F023  88          DEY               az írásmutató új értékének tárolása
```



```
F024  A5 9E       LDA   9E          a kiírandó byte
F026  91 F9       STA   (F9),Y      a pufferba
F028  AD A1 02    LDA   02A1        az RS232 NMI-kapcsoló
F02B  4A          LSR   A           a Ø. bit vizsgálata
F02C  B0 1E       BCS   F04C        az adatátvitel vége?
F02E  A9 10       LDA   #10         az A timer
F030  8D 0E DD    STA   DD0E        indítása
F033  AD 99 02    LDA   0299
F036  8D 04 DD    STA   DD04
F039  AD 9A 02    LDA   029A        "baud-rate" timer újrabeállítása
F03C  8D 05 DD    STA   DD05
F03F  A9 81       LDA   #81
F041  20 3B EF    JSR   EF3B        megszakítás az A timer túlcsordulásakor
F044  20 06 EF    JSR   EF06        a CTS és DSR vizsgálata
F047  A9 11       LDA   #11
F049  8D 0E DD    STA   DD0E        az A timer indítása
F04C  60          RTS

******************************
F04D  85 99       STA   99          RS232 CHKIN, az RS232 input beállítása
F04F  AD 94 02    LDA   0294        egységszám
F052  4A          LSR   A           RS232 utasításszó
F053  90 28       BCC   F07D        Ø.bit
F055  29 08       AND   #08         a 3-as handshake-vonal?
F057  F0 24       BEQ   F07D
F059  A9 02       LDA   #02
F05B  2C 01 DD    BIT   DD01        DSR lekérdezése
F05E  10 AD       BPL   F00D        nem
F060  F0 22       BEQ   F084        RTS lekérdezése
F062  AD A1 02    LDA   02A1        RS232 NMI-kapcsoló
F065  4A          LSR   A           a továbbító aktív?
F066  B0 FA       BCS   F062        ha igen, várakozás
F068  AD 01 DD    LDA   DD01
F06B  29 FD       AND   #FD         RTS
F06D  8D 01 DD    STA   DD01
F070  AD 01 DD    LDA   DD01
F073  29 04       AND   #04         data terminal ready
F075  F0 F9       BEQ   F070        nem, várni
F077  A9 90       LDA   #90         a "flag" NMI-maszkja
F079  18          CLC
F07A  4C 3B EF    JMP   EF3B        NMI törlése
F07D  AD A1 02    LDA   02A1        az RS232 NMI-kapcsoló
F080  29 12       AND   #12         az RS232 nem aktív?
F082  F0 F3       BEQ   F077        akkor start!
F084  18          CLC
F085  60          RTS

******************************
F086  AD 97 02    LDA   0297        GET a RS232-ről
F089  AC 9C 02    LDY   029C        a státusz
F08C  CC 9B 02    CPY   029B        a puffer-mutató
F08F  F0 0B       BEQ   F09C        egyenlő
F091  29 F7       AND   #F7         a puffer üres?
F093  8D 97 02    STA   0297        3. bit /üres puffer/
F096  B1 F7       LDA   (F7),Y      törlése a státuszban
F098  EE 9C 02    INC   029C        a byte beolvasása a pufferből
                                    a puffer-mutató növelése
```

```
F09B  60          RTS
F09C  09 08       ORA  #08        a puffer üres
F09E  8D 97 02    STA   0297      a státusz beállítása
F0A1  A9 00       LDA  #00        nulla átadása
F0A3  60          RTS

*****************************     RS232 várakozás az átvitel végére
F0A4  48          PHA
F0A5  AD A1 02    LDA   02A1      az RS232 NMI flag magas?
F0A8  F0 11       BEQ   F0BB      ha nem, akkor rendben
F0AA  AD A1 02    LDA   02A1
F0AD  29 03       AND  #03        0.bit /küldés/ és 1.bit /fogadás/
F0AF  D0 F9       BNE   F0AA      várakozás
F0B1  A9 10       LDA  #10
F0B3  8D 0D DD    STA   DD0D      megszakítás a "flag"-vonalon
F0B6  A9 00       LDA  #00
F0B8  8D A1 02    STA   02A1      a kapcsoló visszaállítása
F0BB  68          PLA
F0BC  60          RTS

*****************************     rendszerüzenetek
F0BD  0D 49 2F 4F 20 45 52 52     I/O error#
F0C5  4F 52 20 A3
F0C9  0D 53 45 41 52 43 48 49     searching
F0D1  4E 47 A0
F0D4  46 4F 52 A0                 for
F0D8  0D 50 52 45 53 53 20 50     press play on tape
F0E0  4C 41 59 20 4F 4E 20 54
F0E8  41 50 C5
F0EB  50 52 45 53 53 20 52 45     press record & play on tape
F0F3  43 4F 52 44 20 26 20 50
F0FB  4C 41 59 20 4F 4E 20 54
F103  41 50 C5
F106  0D 4C 4F 41 44 49 4E C7     loading
F10E  0D 53 41 56 49 4E 47 A0     saving
F116  0D 56 45 52 49 46 59 49     verifying
F11E  4E C7
F120  0D 46 4F 55 4E 44 A0 0D     found
F128  4F 4B 8D                    ok

*****************************     rendszerüzenetek kiírása
F12B  24 9D       BIT   9D        közvetlen mód kapcsoló
F12D  10 0D       BPL   F13C      ha program, átugrani
F12F  B9 BD F0    LDA   F0BD,Y    az üzenet offsetje az Y-ban
F132  08          PHP
F133  29 7F       AND  #7F        a 7 bit törlése
F135  20 D2 FF    JSR   FFD2      kiírás
F138  C8          INY
F139  28          PLP
F13A  10 F3       BPL   F12F      van még betü?
F13C  18          CLC
F13D  60          RTS

*****************************     GETIN
F13E  A5 99       LDA   99        a beneneti egység
F140  D0 08       BNE   F14A
```



```
F142  A5 C6       LDA   C6        a karakterek száma a bill.pufferban
F144  F0 0F       BEQ   F155      nincs karakter?
F146  78          SEI
F147  4C B4 E5    JMP   E5B4      karakterek beolvasása a pufferból
F14A  C9 02       CMP  #02
F14C  D0 18       BNE   F166      ha nem RS232, akkor a BASIN-rutinhoz
F14E  84 97       STY   97
F150  20 86 F0    JSR   F086      GET az RS232-ből
F153  A4 97       LDY   97
F155  18          CLC
F156  60          RTS

*****************************     BASIN - egy karakter beolvasása
F157  A5 99       LDA   99        az egységszám
F159  D0 0B       BNE   F166      nem billentyüzet?
F15B  A5 D3       LDA   D3        kurzorpozició
F15D  85 CA       STA   CA
F15F  A5 D6       LDA   D6        beolvasas a billentyüzetről
F161  85 C9       STA   C9
F163  4C 32 E6    JMP   E632      bevitel a képernyőről
F166  C9 03       CMP  #03
F168  D0 09       BNE   F173

*****************************     a képernyőről
F16A  85 D0       STA   D0
F16C  A5 D5       LDA   D5
F16E  85 C8       STA   C8
F170  4C 32 E6    JMP   E632
F173  B0 38       BCS   F1AD      az IEC-busztól
F175  C9 02       CMP  #02
F177  F0 3F       BEQ   F1B8

*****************************     a szalagról
F179  86 97       STX   97        az X regiszter mentése
F17B  20 99 F1    JSR   F199      egy karakter beolvasása szalagról
F17E  B0 16       BCS   F196
F180  48          PHA
F181  20 99 F1    JSR   F199      egy karakter beolvasása szalagról
F184  B0 0D       BCS   F193      hiba?
F186  D0 05       BNE   F18D      utolsó karakter
F188  A9 40       LDA  #40        EOF
F18A  20 1C FE    JSR   FE1C      a státusz beállítása
F18D  C6 A6       DEC   A6        szalag -puffer mutató csökkentése
F18F  A6 97       LDX   97        az X-regiszter visszatöltése
F191  68          PLA
F192  60          RTS
F193  AA          TAX             a hiba sorszáma az X-ben
F194  68          PLA             a karakter visszaolvasása
F195  8A          TXA             a hiba sorszáma az akku-ban
F196  A6 97       LDX   97        az X-regiszter visszaolvasása
F198  60          RTS

*****************************     egy karakter beolvasása szalagról
F199  20 0D F8    JSR   F80D      szalag -puffer mutató növelése
F19C  D0 0B       BNE   F1A9      van még karakter a pufferban?
F19E  20 41 F8    JSR   F841      ha nincs, következő blokk beolvasása
```

```
F1A1  B0 11       BCS   F1B4      STOP-billentyu esetén megszakítás
F1A3  A9 00       LDA   #00
F1A5  85 A6       STA   A6        puffer-mutató=nulla, egy karakter beolv.
F1A7  F0 F0       BEQ   F199
F1A9  B1 B2       LDA   (B2),Y    karakter beolvasása a pufferből
F1AB  18          CLC
F1AC  60          RTS
F1AD  A5 90       LDA   90        a státusz vizsgálata
F1AF  F0 04       BEQ   F1B5      ok
F1B1  A9 0D       LDA   #0D       "CR" -kód kiírása
F1B3  18          CLC
F1B4  60          RTS

*****************************
F1B5  4C 13 EE    JMP   EE13      IEC-input
                                  egy byte beolvasása az IEC buszról

*****************************
F1B8  20 4E F1    JSR   F14E      RS232 input
F1BB  B0 F7       BCS   F1B4      egy byte beolvasása az RS232-ről
F1BD  C9 00       CMP   #00       hiba?
F1BF  D0 F2       BNE   F1B3      nulla byte?
F1C1  AD 97 02    LDA   0297      ha nem, akkor ok
F1C4  29 60       AND   #60       a státusz
F1C6  D0 E9       BNE   F1B1      hiányzik a DSR?
F1C8  F0 EE       BEQ   F1B8      igen, "CR" visszaadása
                                  ha nem, új kisérlet

*****************************
F1CA  48          PHA             BSOUT - egy karakter kiírása
F1CB  A5 9A       LDA   9A
F1CD  C9 03       CMP   #03       egységszám
F1CF  D0 04       BNE   F1D5      képernyő?
F1D1  68          PLA             nem
F1D2  4C 16 E7    JMP   E716
F1D5  90 04       BCC   F1DB      egy karakter kiírása a képernyőre
F1D7  68          PLA
F1D8  4C DD ED    JMP   EDDD
F1DB  4A          LSR   A         egy byte kiírása az IEC buszra
F1DC  68          PLA
F1DD  85 9E       STA   9E        a karakter tárolása
F1DF  8A          TXA
F1E0  48          PHA
F1E1  98          TYA
F1E2  48          PHA
F1E3  90 23       BCC   F208      RS232 output
F1E5  20 0D F8    JSR   F80D      a szalag-puffer mutató növelése
F1E8  D0 0E       BNE   F1F8      a puffer megtelt?
F1EA  20 64 F8    JSR   F864      a puffer kiírása szalagra
F1ED  B0 0E       BCS   F1FD      STOP esetén megszakítás
F1EF  A9 02       LDA   #02       az adatblokk ellenőrzőbyte-ja
F1F1  A0 00       LDY   #00
F1F3  91 B2       STA   (B2),Y    a pufferba
F1F5  C8          INY
F1F6  84 A6       STY   A6        a puffer-mutató növelése és tárolása
F1F8  A5 9E       LDA   9E
F1FA  91 B2       STA   (B2),Y    a karakter beírása a pufferba
F1FC  18          CLC
```



```
F1FD  68          PLA             a karakter visszaolvasása
F1FE  A8          TAY             ok?
F1FF  68          PLA             "STOP-billentyű lenyomva" kapcsoló
F200  AA          TAX
F201  A5 9E       LDA   9E
F203  90 02       BCC   F207
F205  A9 00       LDA   #00
F207  60          RTS

*****************************
F208  20 17 F0    JSR   F017      RS232 output
F20B  4C FC F1    JMP   F1FC      egy karakter beírása az RS232-pufferbe

*****************************
F20E  20 0F F3    JSR   F30F      CHKIN - beviteli egység beállítása
F211  F0 03       BEQ   F216      logikai file-szám keresés
F213  4C 01 F7    JMP   F701      megvan?
F216  20 1F F3    JSR   F31F      "file not open"
F219  A5 BA       LDA   BA        a file-paraméter beállítása
F21B  F0 16       BEQ   F233      egységszám
F21D  C9 03       CMP   #03       0, billentyúzet
F21F  F0 12       BEQ   F233
F221  B0 14       BCS   F237      3, képernyő
F223  C9 02       CMP   #02       IEC-busz
F225  D0 03       BNE   F22A      RS232?
F227  4C 4D F0    JMP   F04D      ha nem, akkor szalag
                                  igen

*****************************
F22A  A6 B9       LDX   B9        bemeneti egység a kazetta
F22C  E0 60       CPX   #60       másodlagos cím
F22E  F0 03       BEQ   F233      nulla?
F230  4C 0A F7    JMP   F70A
F233  85 99       STA   99        "not input file"
F235  18          CLC             a kimeneti egység száma
F236  60          RTS
F237  AA          TAX
F238  20 09 ED    JSR   ED09      TALK küldése
F23B  A5 B9       LDA   B9        a másodlagos cím
F23D  10 06       BPL   F245
F23F  20 CC ED    JSR   EDCC      várakozás a rendszerütemre
F242  4C 48 F2    JMP   F248
F245  20 C7 ED    JSR   EDC7      a TALK másodlagos címe
F248  8A          TXA
F249  24 90       BIT   90        a státusz
F24B  10 E6       BPL   F233      ok?
F24D  4C 07 F7    JMP   F707      "devise not present"

*****************************
F250  20 0F F3    JSR   F30F      CHKOUT - a kimeneti egység beállítása
F253  F0 03       BEQ   F258      logikai file-szám keresése
F255  4C 01 F7    JMP   F701      megvan
F258  20 1F F3    JSR   F31F      "file not open"
F25B  A5 BA       LDA   BA        a file-paraméter beállítása
F25D  D0 03       BNE   F262      egységszám
F25F  4C 0D F7    JMP   F70D      nem egyenlő nullával?
F262  C9 03       CMP   #03       "not input file"
                                  képernyő?
```

```
F264  F0 0F       BEQ   F275        igen
F266  B0 11       BCS   F279        IEC-busz
F268  C9 02       CMP   #02         RS232?
F26A  D0 03       BNE   F26F
F26C  4C E1 EF    JMP   EFE1        igen

*****************************     kimeneti egység a kazetta
F26F  A6 B9       LDX   B9          másodlagos cím
F271  E0 60       CPX   #60         nulla?
F273  F0 EA       BEQ   F25F        az olvasandó file,"not output file"
F275  85 9A       STA   9A          a kimeneti egység száma
F277  18          CLC
F278  60          RTS

*****************************     kimenet az IEC-buszra
F279  AA          TAX
F27A  20 0C ED    JSR   ED0C        LISTEN küldése
F27D  A5 B9       LDA   B9          a másodlagos cím
F27F  10 05       BPL   F286
F281  20 BE ED    JSR   EDBE        az ATN visszaállítása
F284  D0 03       BNE   F289
F286  20 B9 ED    JSR   EDB9        LISTEN másodlagos cím
F289  8A          TXA
F28A  24 90       BIT   90          a státusz
F28C  10 E7       BPL   F275        ok
F28E  4C 07 F7    JMP   F707        "device not present"

*****************************     CLOSE - logikai file-szám az "A"-ba
F291  20 14 F3    JSR   F314        logikai file-szám keresése
F294  F0 02       BEQ   F298
F296  18          CLC               ha nincs ilyen file, akkor kész
F297  60          RTS
F298  20 1F F3    JSR   F31F        a file-paraméter beállítása
F29B  8A          TXA
F29C  48          PHA
F29D  A5 BA       LDA   BA          egységszám
F29F  F0 50       BEQ   F2F1        billentyűzet?
F2A1  C9 03       CMP   #03
F2A3  F0 4C       BEQ   F2F1        képernyő?
F2A5  B0 47       BCS   F2EE        IEC-busz?
F2A7  C9 02       CMP   #02         RS232?
F2A9  D0 1D       BNE   F2C8        szalag

*****************************     az RS232 file lezárása
F2AB  68          PLA
F2AC  20 F2 F2    JSR   F2F2        a táblázati file-bejegyzés törlése
F2AF  20 83 F4    JSR   F483        I/O CIA-inak visszaállítása
F2B2  20 27 FE    JSR   FE27        RAM- top betöltése
F2B5  A5 F8       LDA   F8          input-puffer
F2B7  F0 01       BEQ   F2BA
F2B9  C8          INY
F2BA  A5 FA       LDA   FA          output-puffer
F2BC  F0 01       BEQ   F2BF
F2BE  C8          INY
F2BF  A9 00       LDA   #00
F2C1  85 F8       STA   F8          puffer felszabadítása
```



```
F2C3  85 FA       STA   FA
F2C5  4C 7D F4    JMP   F47D        a RAM-top visszaállítása

*****************************     a szalag-file lezárása
F2C8  A5 B9       LDA   B9          másodlagos cím
F2CA  29 0F       AND   #0F
F2CC  F0 23       BEQ   F2F1        olvasási file?
F2CE  20 D0 F7    JSR   F7D0        a szalag-puffer kezdőcímének betöltése
F2D1  A9 00       LDA   #00
F2D3  38          SEC
F2D4  20 DD F1    JSR   F1DD
F2D7  20 64 F8    JSR   F864        puffer az íráshoz
F2DA  90 04       BCC   F2E0
F2DC  68          PLA
F2DD  A9 00       LDA   #00
F2DF  60          RTS
F2E0  A5 B9       LDA   B9          a másodlagos cím
F2E2  C9 62       CMP   #62         2-vel egyenlő
F2E4  D0 0B       BNE   F2F1
F2E6  A9 05       LDA   #05         EOT-fej ellenőrzőbyte-ja
F2E8  20 6A F7    JSR   F76A        a blokk felírása a szalagra
F2EB  4C F1 F2    JMP   F2F1
F2EE  20 42 F6    JSR   F642        az IEC file lezárása
F2F1  68          PLA
F2F2  AA          TAX
F2F3  C6 98       DEC   98          a megnyitott file-ok számának csökkentése
F2F5  E4 98       CPX   98
F2F7  F0 14       BEQ   F30D        ha nullával egyenlő, akkor kész
F2F9  A4 98       LDY   98
F2FB  B9 59 02    LDA   0259,Y      a file-paraméter táblázat megnyitása
F2FE  9D 59 02    STA   0259,X
F301  B9 63 02    LDA   0263,Y
F304  9D 63 02    STA   0263,X
F307  B9 6D 02    LDA   026D,Y
F30A  9D 6D 02    STA   026D,X
F30D  18          CLC
F30E  60          RTS

*****************************     logikai file-szám keresése /az X-ben/
F30F  A9 00       LDA   #00
F311  85 90       STA   90          a státusz törlése
F313  8A          TXA
F314  A6 98       LDX   98          a megnyitott file-ok száma
F316  CA          DEX
F317  30 15       BMI   F32E
F319  DD 59 02    CMP   0259,X      táblázati bejegyzés keresése
F31C  D0 F8       BNE   F316
F31E  60          RTS

*****************************     a file-paraméter/ek/ beállítása
F31F  BD 59 02    LDA   0259,X
F322  85 B8       STA   B8          a logikai file-szám
F324  BD 63 02    LDA   0263,X
F327  85 BA       STA   BA          az egységszám
F329  BD 6D 02    LDA   026D,X
F32C  85 B9       STA   B9          a másodlagos cím
```

```
F32E  60          RTS

*****************************
F32F  A9 00       LDA  #00
F331  85 98       STA  98

*****************************
F333  A2 03       LDX  #03
F335  E4 9A       CPX  9A
F337  B0 03       BCS  F33C
F339  20 FE ED    JSR  EDFE
F33C  E4 99       CPX  99
F33E  B0 03       BCS  F343
F340  20 EF ED    JSR  EDEF
F343  86 9A       STX  9A
F345  A9 00       LDA  #00
F347  85 99       STA  99
F349  60          RTS

*****************************
F34A  A6 B8       LDX  B8
F34C  D0 03       BNE  F351
F34E  4C 0A F7    JMP  F70A
F351  20 0F F3    JSR  F30F
F354  D0 03       BNE  F359
F356  4C FE F6    JMP  F6FE
F359  A6 98       LDX  98
F35B  E0 0A       CPX  #0A
F35D  90 03       BCC  F362
F35F  4C FB F6    JMP  F6FB
F362  E6 98       INC  98
F364  A5 B8       LDA  B8
F366  9D 59 02    STA  0259,X
F369  A5 B9       LDA  B9
F36B  09 60       ORA  #60
F36D  85 B9       STA  B9
F36F  9D 6D 02    STA  026D,X
F372  A5 BA       LDA  BA
F374  9D 63 02    STA  0263,X
F377  F0 5A       BEQ  F3D3
F379  C9 03       CMP  #03
F37B  F0 56       BEQ  F3D3
F37D  90 05       BCC  F384
F37F  20 D5 F3    JSR  F3D5
F382  90 4F       BCC  F3D3
F384  C9 02       CMP  #02
F386  D0 03       BNE  F38B
F388  4C 09 F4    JMP  F409
F38B  20 D0 F7    JSR  F7D0
F38E  B0 03       BCS  F393
F390  4C 13 F7    JMP  F713
F393  A5 B9       LDA  B9
F395  29 0F       AND  #0F
F397  D0 1F       BNE  F3B8
F399  20 17 F8    JSR  F817
F39C  B0 36       BCS  F3D4
```

CLALL - lezárja az input/output csat.

a megnyitott file-ok száma = nullával

CLRCH - lezárja az aktív I/O-csatornát

a kimeneti egység száma
3-nál kisebb
IEC, UNLISTEN küldése
bemeneti egység száma
3-nál kisebb
IEC, UNTALK küldése
kivitel ismét a képernyőre

beolvasás a billentyűről

OPEN
a file-szám
nem egyenlő nullával
"not input file"(??)
a logikai file-szám keresése
ha nincs, új sorszám beállítása
"file open"
a megnyitott file-ok számának
összehasonlítása 10-zel

"too many files"
a nyitott file-ok számának növelése
a logikai file-szám

a másodlagos cím

az egységszám
beírása a megfelelő táblázatokba
billentyúzet?

képernyő

file megnyitása az IEC-buszon

szalag?
nem
RS232 open
a szalag-puffer kezdőcímének beolvasása

"illegal device number"
a másodlagos cím

nem egyenlő nullával, akkor írás
várakozás a Play billentyúre
a STOP-billentyú le van nyomva?


```
F39E  20 AF F5    JSR  F5AF
F3A1  A5 B7       LDA  B7
F3A3  F0 0A       BEQ  F3AF
F3A5  20 EA F7    JSR  F7EA
F3A8  90 18       BCC  F3C2
F3AA  F0 28       BEQ  F3D4
F3AC  4C 04 F7    JMP  F704
F3AF  20 2C F7    JSR  F72C
F3B2  F0 20       BEQ  F3D4
F3B4  90 0C       BCC  F3C2
F3B6  B0 F4       BCS  F3AC
F3B8  20 38 F8    JSR  F838
F3BB  B0 17       BCS  F3D4
F3BD  A9 04       LDA  #04
F3BF  20 6A F7    JSR  F76A
F3C2  A9 BF       LDA  #BF
F3C4  A4 B9       LDY  B9
F3C6  C0 60       CPY  #60
F3C8  F0 07       BEQ  F3D1
F3CA  A0 00       LDY  #00
F3CC  A9 02       LDA  #02
F3CE  91 B2       STA  (B2),Y
F3D0  98          TYA
F3D1  85 A6       STA  A6
F3D3  18          CLC
F3D4  60          RTS

*****************************
F3D5  A5 B9       LDA  B9
F3D7  30 FA       BMI  F3D3
F3D9  A4 B7       LDY  B7
F3DB  F0 F6       BEQ  F3D3
F3DD  A9 00       LDA  #00
F3DF  85 90       STA  90
F3E1  A5 BA       LDA  BA
F3E3  20 0C ED    JSR  ED0C
F3E6  A5 B9       LDA  B9
F3E8  09 F0       ORA  #F0
F3EA  20 B9 ED    JSR  EDB9
F3ED  A5 90       LDA  90
F3EF  10 05       BPL  F3F6
F3F1  68          PLA
F3F2  68          PLA
F3F3  4C 07 F7    JMP  F707
F3F6  A5 B7       LDA  B7
F3F8  F0 0C       BEQ  F406
F3FA  A0 00       LDY  #00
F3FC  B1 BB       LDA  (BB),Y
F3FE  20 DD ED    JSR  EDDD
F401  C8          INY
F402  C4 B7       CPY  B7
F404  D0 F6       BNE  F3FC
F406  4C 54 F6    JMP  F654

*****************************
F409  20 83 F4    JSR  F483
```

"searching" /"for name"/ kiírása
a file-név hossza
ha nincs file-név, akkor tovább
a kívánt szalag-fejet keresi
ha megvan
ok
EOT, "file not found" kiírása
következő szalag-fej keresése
EOT, hiba
megvan
PRG-file, továbbkeresni
a record & Play billentyúre vár
ha STOP billentyú, akkor megszakítás
a fejléc ellenőrzőbyte-ja
a fejléc felírása a szalagra
a szalag-puffer végének mutatója
a másodlagos cím

ha egyenlő nullával, akkor tovább

adatblokk ellenőrzőbyte-jának beírása
a szalag-pufferba

mutató a szalag-pufferre

egy file megnyitása az IEC buszon
a másodlagos cím

a file-név hossza
ha nullával egyenlő, akkor kész

a státusz törlése
az egységszám
LISTEN
a másodlagos cím

küldése
a státusz tesztelése
ok

"device not present"
a file-név hossza

a file-név
kiírása az IEC-buszra

UNLISTEN, return

RS232 OPEN
a CIA-k beállítása

```
F40C  8C 97 02     STY   0297        az RS232 státusz törlése
F40F  C4 B7        CPY   B7          a file-név hossza
F411  F0 0A        BEQ   F41D
F413  B1 BB        LDA   (BB),Y      az első 4 karakter tárolása
F415  99 93 02     STA   0293,Y
F418  C8           INY
F419  C0 04        CPY   #04
F41B  D0 F2        BNE   F40F
F41D  20 4A EF     JSR   EF4A        az adatbitek számának meghatározása
F420  8E 98 02     STX   0298        és tárolása
F423  AD 93 02     LDA   0293        az ellenőrző regiszter
F426  29 0F        AND   #0F         "baud-rate" bitek leválasztása
F428  F0 1C        BEQ   F446
F42A  0A           ASL   A           * 2 a táblázathoz
F42B  AA           TAX
F42C  AD A6 02     LDA   02A6        NTSC-változat?
F42F  D0 09        BNE   F43A        nem
F431  BC C1 FE     LDY   FEC1,X      baud-rate, az NTSC felső byte
F434  BD C0 FE     LDA   FEC0,X      baud-rate, alsó byte
F437  4C 40 F4     JMP   F440
F43A  BC EB E4     LDY   E4EB,X      baud-rate, PAL időzítő, alsó
F43D  BD EA E4     LDA   E4EA,X      baud-rate, alsó byte
F440  8C 96 02     STY   0296
F443  8D 95 02     STA   0295        tárolás
F446  AD 95 02     LDA   0295
F449  0A           ASL   A
F44A  20 2E FF     JSR   FF2E        baud-rate kódjának kiszámítása
F44D  AD 94 02     LDA   0294
F450  4A           LSR   A
F451  90 09        BCC   F45C
F453  AD 01 DD     LDA   DD01        hiányzik a DSR?
F456  0A           ASL   A
F457  B0 03        BCS   F45C        nem
F459  20 0D F0     JSR   F00D        a DSR státuszának beállítása
F45C  AD 9B 02     LDA   029B
F45F  8D 9C 02     STA   029C        az RS232 input- output puffer-mutató beáll.
F462  AD 9E 02     LDA   029E
F465  8D 9D 02     STA   029D
F468  20 27 FE     JSR   FE27        a tártető betöltése
F46B  A5 F8        LDA   F8
F46D  D0 05        BNE   F474        input-puffer kész?
F46F  88           DEY
F470  84 F8        STY   F8          az RS232 input-puffer mutatója
F472  86 F7        STX   F7
F474  A5 FA        LDA   FA
F476  D0 05        BNE   F47D        az output-puffer már kész?
F478  88           DEY
F479  84 FA        STY   FA          az RS232 output-puffer mutatója
F47B  86 F9        STX   F9
F47D  38           SEC
F47E  A9 F0        LDA   #F0         a kapcsoló beállítása
F480  4C 2D FE     JMP   FE2D        a tártető visszaállítása

*****************************       a CIA-k visszaállítása az RS232-be
F483  A9 7F        LDA   #7F
F485  8D 0D DD     STA   DD0D        IRQ-k visszaállítása
```



```
F488  A9 06        LDA   #06         az 1-es és 2-es bit kimenet
F48A  8D 03 DD     STA   DD03        a B port iránya
F48D  8D 01 DD     STA   DD01        az A port iránya
F490  A9 04        LDA   #04
F492  0D 00 DD     ORA   DD00        a 2.bit=TAD
F495  8D 00 DD     STA   DD00
F498  A0 00        LDY   #00
F49A  8C A1 02     STY   02A1        az NMI-vonal törlése
F49D  60           RTS

*****************************
F49E  86 C3        STX   C3          a LOAD rutin
F4A0  84 C4        STY   C4          a kezdőcím tárolása
F4A2  6C 30 03     JMP   (0330)      JMP $F4A5 LOAD-vektor
F4A5  85 93        STA   93          load/verify kapcsoló
F4A7  A9 00        LDA   #00         a státusz törlése
F4A9  85 90        STA   90          ha az egységszám
F4AB  A5 BA        LDA   BA          nem egyenlő nullával, akkor tovább
F4AD  D0 03        BNE   F4B2        "illegal device number"
F4AF  4C 13 F7     JMP   F713        képernyő?
F4B2  C9 03        CMP   #03         ha igen, akkor hiba
F4B4  F0 F9        BEQ   F4AF        ha 3-nál kisebb, betöltés szalagról
F4B6  90 7B        BCC   F533

*****************************
F4B8  A4 B7        LDY   B7          IEC-load
F4BA  D0 03        BNE   F4BF        a file-név hossza
F4BC  4C 10 F7     JMP   F710        ha nem egyenlő nullával, akkor ok
F4BF  A6 B9        LDX   B9          "missing filename"
F4C1  20 AF F5     JSR   F5AF        a másodlagos cím
F4C4  A9 60        LDA   #60         "searching for filename"
F4C6  85 B9        STA   B9          a másodlagos cím nulla
F4C8  20 D5 F3     JSR   F3D5
F4CB  A5 BA        LDA   BA          a file megnyitása az IEC buszon
F4CD  20 09 ED     JSR   ED09        az egységszám
F4D0  A5 B9        LDA   B9          TALK küldése
F4D2  20 C7 ED     JSR   EDC7
F4D5  20 13 EE     JSR   EE13        másodlagos cím küldése
F4D8  85 AE        STA   AE          egy byte beolvasása az IEC-buszról
F4DA  A5 90        LDA   90          kezdőcím alsó byte
F4DC  4A           LSR   A           státusz
F4DD  4A           LSR   A
F4DE  B0 50        BCS   F530        ha time out, akkor hiba
F4E0  20 13 EE     JSR   EE13        a kezdőcím felső byte beolvasása
F4E3  85 AF        STA   AF
F4E5  8A           TXA
F4E6  D0 08        BNE   F4F0        másodlagos cím nem egyenlő nullával?
F4E8  A5 C3        LDA   C3          ha nem, előzetes címtől kezdődő betöltés
F4EA  85 AE        STA   AE
F4EC  A5 C4        LDA   C4
F4EE  85 AF        STA   AF
F4F0  20 D2 F5     JSR   F5D2        "loading"/"verifying" kiírása
F4F3  A9 FD        LDA   #FD
F4F5  25 90        AND   90          a time-out bit törlése
F4F7  85 90        STA   90
F4F9  20 E1 FF     JSR   FFE1        a stop billentyű lekérdezése
```

```
F4FC  D0 03      BNE   F501      ha nincs lenyomva, akkor tovább
F4FE  4C 33 F6   JMP   F633      a file lezárása
F501  20 13 EE   JSR   EE13      a program-byte beolvasása a buszról
F504  AA         TAX
F505  A5 90      LDA   90        a státusz vizsgálata
F507  4A         LSR   A
F508  4A         LSR   A
F509  B0 E8      BCS   F4F3      ha hiba, akkor megszakítás
F50B  8A         TXA
F50C  A4 93      LDY   93        load/verify kapcsoló vizsgálata
F50E  F0 0C      BEQ   F51C      ha nullával egyenlő, akkor LOAD
F510  A0 00      LDY   #00
F512  D1 AE      CMP   (AE),Y    verify, összehasonlítás
F514  F0 0B      BEQ   F51E
F516  A9 10      LDA   #10       ha nem egyenlő, státusz beállítása
F518  20 1C FE   JSR   FE1C      a státusz beállítása
F51B  2C
F51C  91 AE      STA   (AE),Y    a byte tárolása
F51E  E6 AE      INC   AE
F520  D0 02      BNE   F524      a cím növelése
F522  E6 AF      INC   AF
F524  24 90      BIT   90        státusz
F526  50 CB      BVC   F4F3      még nem EOF?
F528  20 EF ED   JSR   EDEF      UNTALK küldése
F52B  20 42 F6   JSR   F642      a file lezárása
F52E  90 79      BCC   F5A9      nincs hiba?
F530  4C 04 F7   JMP   F704      "file not found"

*****************************
F533  4A         LSR   A         egységszám
F534  B0 03      BCS   F539      ha egyes /szalag/, akkor tovább
F536  4C 13 F7   JMP   F713      RS232, "illegal device number"
F539  20 D0 F7   JSR   F7D0      szalag-puffer kezdőcímének beolvasása
F53C  B0 03      BCS   F541
F53E  4C 13 F7   JMP   F713      "illegal device number"
F541  20 17 F8   JSR   F817      várakozás a play-re
F544  B0 68      BCS   F5AE      ha stop-billentyű, akkor megszakítás
F546  20 AF F5   JSR   F5AF      "searching" /"for name"/ kiírása
F549  A5 B7      LDA   B7        a file-név hossza
F54B  F0 09      BEQ   F556      ha Ø-val egyenlő, akkor tovább
F54D  20 EA F7   JSR   F7EA      kívánt fejléc keresése
F550  90 0B      BCC   F55D      megvan
F552  F0 5A      BEQ   F5AE      STOP-billentyű, megszakítás
F554  B0 DA      BCS   F530      ha EOT, akkor "file not found"
F556  20 2C F7   JSR   F72C      következő fejléc keresése
F559  F0 53      BEQ   F5AE      STOP-billentyű, megszakítás
F55B  B0 D3      BCS   F530      Ha "EOT", akkor"file not found"
F55D  A5 90      LDA   90        a státusz beolvasása
F55F  29 10      AND   #10       az EOF bit leválasztása
F561  38         SEC
F562  D0 4A      BNE   F5AE      más hiba?
F564  E0 01      CPX   #01       1. fejtipus=BASIC-program /betöltés/
F566  F0 11      BEQ   F579
F568  E0 03      CPX   #03       3=gépi program /abszolut/
F56A  D0 DD      BNE   F549
F56C  A0 01      LDY   #01
```



```
F56E  B1 B2      LDA   (B2),Y    kezdőcím alsó
F570  85 C3      STA   C3
F572  C8         INY
F573  B1 B2      LDA   (B2),Y    kezdőcím felső
F575  85 C4      STA   C4
F577  B0 04      BCS   F57D
F579  A5 B9      LDA   B9        a másodlagos cím
F57B  D0 EF      BNE   F56C      ha nem nulla, a betöltés nem eltolható
F57D  A0 03      LDY   #03
F57F  B1 B2      LDA   (B2),Y
F581  A0 01      LDY   #01       a végcím minusz
F583  F1 B2      SBC   (B2),Y
F585  AA         TAX
F586  A0 04      LDY   #04
F588  B1 B2      LDA   (B2),Y
F58A  A0 02      LDY   #02       a kezdőcím
F58C  F1 B2      SBC   (B2),Y
F58E  A8         TAY
F58F  18         CLC             egyenlő a programhosszúsággal
F590  8A         TXA
F591  65 C3      ADC   C3
F593  85 AE      STA   AE        programhossz + kezdőcím
F595  98         TYA
F596  65 C4      ADC   C4        egyenlő a végcímmel
F598  85 AF      STA   AF
F59A  A5 C3      LDA   C3
F59C  85 C1      STA   C1        kezdőcím a $C1/$C2-re
F59E  A5 C4      LDA   C4
F5A0  85 C2      STA   C2
F5A2  20 D2 F5   JSR   F5D2      "loading"/"verifying" kiírása
F5A5  20 4A F8   JSR   F84A      program betöltése a szalagról
F5A8  24
F5A9  18         CLC
F5AA  A6 AE      LDX   AE        végcím az X/Y-ba
F5AC  A4 AF      LDY   AF
F5AE  60         RTS

*****************************   "searching" /for filename/ kiírása
F5AF  A5 9D      LDA   9D        közvetlen mód?
F5B1  10 1E      BPL   F5D1      ha nem, akkor kihagyni
F5B3  A0 0C      LDY   #0C       a "searching"- offset
F5B5  20 2F F1   JSR   F12F      üzenet kiírása
F5B8  A5 B7      LDA   B7        a file-név hossza
F5BA  F0 15      BEQ   F5D1      ha nulla, akkor kész
F5BC  A0 17      LDY   #17       "for"-offset
F5BE  20 2F F1   JSR   F12F      az üzenet kiírása
F5C1  A4 B7      LDY   B7        a file-név hossza
F5C3  F0 0C      BEQ   F5D1      ha nulla, akkor kész
F5C5  A0 00      LDY   #00
F5C7  B1 BB      LDA   (BB),Y    a file-név beolvasása
F5C9  20 D2 FF   JSR   FFD2      és kiírása
F5CC  C8         INY
F5CD  C4 B7      CPY   B7
F5CF  D0 F6      BNE   F5C7
F5D1  60         RTS

*****************************   "loading"/"verifying" kiírása
```

```
F5D2  A0 49       LDY  #49      "loading"-offset
F5D4  A5 93       LDA  93       load/verify kapcs. vizsgálata
F5D6  F0 02       BEQ  F5DA     ha Load, akkor kiírás
F5D8  A0 59       LDY  #59      "verifying"-offset
F5DA  4C 2B F1    JMP  F12B     üzenet kiírása

*****************************   a SAVE rutin
F5DD  86 AE       STX  AE       végcím alsó
F5DF  84 AF       STY  AF       végcím felső
F5E1  AA          TAX
F5E2  B5 00       LDA  00,X
F5E4  85 C1       STA  C1       kezdőcím alsó
F5E6  B5 01       LDA  01,X
F5E8  85 C2       STA  C2       kezdőcím felső
F5EA  6C 32 03    JMP  (0332)   a SAVE vektor, JMP $F5ED
F5ED  A5 BA       LDA  BA       egységszám
F5EF  D0 03       BNE  F5F4
F5F1  4C 13 F7    JMP  F713     "illegal device number"
F5F4  C9 03       CMP  #03
F5F6  F0 F9       BEQ  F5F1     képernyő, hiba
F5F8  90 5F       BCC  F659     szalag

*****************************   tárolás az IEC buszon
F5FA  A9 61       LDA  #61      1 másodlagos cím
F5FC  85 B9       STA  B9       beállítása
F5FE  A4 B7       LDY  B7       a file-név hossza
F600  D0 03       BNE  F605     ha nem egyenlő nulla, akkor ok
F602  4C 10 F7    JMP  F710     "missing filename"
F605  20 D5 F3    JSR  F3D5     a file-név az IEC buszra
F608  20 8F F6    JSR  F68F     a "saving" kiírása
F60B  A5 BA       LDA  BA       az egységszám
F60D  20 0C ED    JSR  ED0C     LISTEN küldése
F610  A5 B9       LDA  B9       a LISTEN másodlagos címének
F612  20 B9 ED    JSR  EDB9     küldése
F615  A0 00       LDY  #00
F617  20 8E FB    JSR  FB8E     a kezdőcím a $AC/$AD-re
F61A  A5 AC       LDA  AC       kezdőcím alsó
F61C  20 DD ED    JSR  EDDD     küldése
F61F  A5 AD       LDA  AD       és a kezd cím felső
F621  20 DD ED    JSR  EDDD     küldése
F624  20 D1 FC    JSR  FCD1     már a végcím?
F627  B0 16       BCS  F63F     ha igen, akkor kész
F629  B1 AC       LDA  (AC),Y   egy program-byte
F62B  20 DD ED    JSR  EDDD     kiírása az IEC-buszra
F62E  20 E1 FF    JSR  FFE1     a stop billentyű lekérdezése
F631  D0 07       BNE  F63A     ha nincs lenyomva, folytatás
F633  20 42 F6    JSR  F642     az IEC busz csatorna lezárása
F636  A9 00       LDA  #00
F638  38          SEC           "break" output kapcsoló beállítása
F639  60          RTS
F63A  20 DB FC    JSR  FCDB     a cím növelése
F63D  D0 E5       BNE  F624
F63F  20 FE ED    JSR  EDFE     UNLISTEN küldése
F642  24 B9       BIT  B9       másodlagos cím
F644  30 11       BMI  F657     nem másodlagos cím?
F646  A5 BA       LDA  BA       egységszám
```



```
F648  20 0C ED    JSR  ED0C     LISTEN küldése
F64B  A5 B9       LDA  B9       másodlagos cím
F64D  29 EF       AND  #EF
F64F  09 E0       ORA  #E0      a CLOSE-hoz
F651  20 B9 ED    JSR  EDB9     a másodlagos cím kiírása
F654  20 FE ED    JSR  EDFE     UNLISTEN küldése
F657  18          CLC
F658  60          RTS
F659  4A          LSR  A        egységszám/2
F65A  B0 03       BCS  F65F     szalag
F65C  4C 13 F7    JMP  F713     RS232, "illegal device number"
F65F  20 D0 F7    JSR  F7D0     a szalag-puffer kezdőcímének betöltése
F662  90 8D       BCC  F5F1     "illegal device number"
F664  20 38 F8    JSR  F838     record-play billentyűre vár
F667  B0 25       BCS  F68E     ha stop, akkor megszakítás
F669  20 8F F6    JSR  F68F     "saving name" kiírása
F66C  A2 03       LDX  #03      ha a fejtipus 3, gépiprogram /abszolut/
F66E  A5 B9       LDA  B9       másodlagos cím
F670  29 01       AND  #01      0.bit beállítva /1 vagy 3/
F672  D0 02       BNE  F676     ha igen, akkor gépiprogram
F674  A2 01       LDX  #01      1 fejtipus = BASIC-program /eltolható/
F676  8A          TXA
F677  20 6A F7    JSR  F76A     a fej felírása a szalagra
F67A  B0 12       BCS  F68E     kiugrás a stop-billentyűnél
F67C  20 67 F8    JSR  F867     program felírása a szalagra
F67F  B0 0D       BCS  F68E     kiugrás a stop-billentyűnél
F681  A5 B9       LDA  B9       másodlagos cím
F683  29 02       AND  #02      az 1.bit beállítva /2 vagy 3/
F685  F0 06       BEQ  F68D     ha nem, akkor kész
F687  A9 05       LDA  #05      EOT ellenőrző-byte
F689  20 6A F7    JSR  F76A     a blokk felírása a szalagra
F68C  24
F68D  18          CLC
F68E  60          RTS

*****************************   a "saving" kiírása
F68F  A5 9D       LDA  9D       közvetlen mód?
F691  10 FB       BPL  F68E     ha nem, akkor kész
F693  A0 51       LDY  #51      a "saving"-offset
F695  20 2F F1    JSR  F12F     üzenet kiírása
F698  4C C1 F5    JMP  F5C1     a file-név kiírása

*****************************   UDTIM - a time növelése
F69B  A2 00       LDX  #00
F69D  E6 A2       INC  A2
F69F  D0 06       BNE  F6A7     az idő növelése
F6A1  E6 A1       INC  A1
F6A3  D0 02       BNE  F6A7
F6A5  E6 A0       INC  A0
F6A7  38          SEC
F6A8  A5 A2       LDA  A2
F6AA  E9 01       SBC  #01
F6AC  A5 A1       LDA  A1       összehasonlítás a 24 órás értékkel
F6AE  E9 1A       SBC  #1A
F6B0  A5 A0       LDA  A0
F6B2  E9 4F       SBC  #4F
```

```
F6B4  90 06       BCC   F6BC      ha kisebb, akkor ok
F6B6  86 A0       STX   A0
F6B8  86 A1       STX   A1        az idő nullára állítása
F6BA  86 A2       STX   A2
F6BC  AD 01 DC    LDA   DC01
F6BF  CD 01 DC    CMP   DC01      van lenyomott billentyű
F6C2  D0 F8       BNE   F6BC
F6C4  AA          TAX
F6C5  30 13       BMI   F6DA      nincs
F6C7  A2 BD       LDX   #BD
F6C9  8E 00 DC    STX   DC00
F6CC  AE 01 DC    LDX   DC01
F6CF  EC 01 DC    CPX   DC01      a stop-billentyű vizsgálata
F6D2  D0 F8       BNE   F6CC
F6D4  8D 00 DC    STA   DC00
F6D7  E8          INX
F6D8  D0 02       BNE   F6DC
F6DA  85 91       STA   91        a stop-billentyű kapcsoló
F6DC  60          RTS

*****************************     a TIME beolvasása
F6DD  78          SEI
F6DE  A5 A2       LDA   A2
F6E0  A6 A1       LDX   A1
F6E2  A4 A0       LDY   A0

*****************************     a TIME beállítása
F6E4  78          SEI
F6E5  85 A2       STA   A2
F6E7  86 A1       STX   A1
F6E9  84 A0       STY   A0
F6EB  58          CLI
F6EC  60          RTS

*****************************     a stop-billentyű lekérdezése
F6ED  A5 91       LDA   91        kapcsoló
F6EF  C9 7F       CMP   #7F       a stop-kód vizsgálata
F6F1  D0 07       BNE   F6FA
F6F3  08          PHP
F6F4  20 CC FF    JSR   FFCC      CLRCH
F6F7  85 C6       STA   C6        a lenyomott billentyűk száma
F6F9  28          PLP
F6FA  60          RTS

*****************************     az op.rendszer üzeneteinek kiírása
F6FB  A9 01       LDA   #01       "too many files"
F6FD  2C
F6FE  A9 02       LDA   #02       "file open"
F700  2C
F701  A9 03       LDA   #03       "file not open"
F703  2C
F704  A9 04       LDA   #04       "file not found"
F706  2C
F707  A9 05       LDA   #05       "device not present"
F709  2C
F70A  A9 06       LDA   #06       "not input file"
```



```
F70C  2C
F70D  A9 07       LDA   #07       "not output file"
F70F  2C
F710  A9 08       LDA   #08       "missing filename"
F712  2C
F713  A9 09       LDA   #09       "illegal device number"
F715  48          PHA             a hiba sorszámának tárolása
F716  20 CC FF    JSR   FFCC      CLRCH
F719  A0 00       LDY   #00
F71B  24 9D       BIT   9D        a hibajelző kapcsoló vizsgálata
F71D  50 0A       BVC   F729      ha nincs beállíta, kiugrás
F71F  20 2F F1    JSR   F12F      "I/O ERROR# " kiadása
F722  68          PLA
F723  48          PHA             a sorszám beolvasása
F724  09 30       ORA   #30       konvertálása ASCII-re
F726  20 D2 FF    JSR   FFD2      és kiírása
F729  68          PLA
F72A  38          SEC
F72B  60          RTS

*****************************     a program fejléc beolv. szalagról
F72C  A5 93       LDA   93        load/verify-kapcsoló mentése
F72E  48          PHA
F72F  20 41 F8    JSR   F841      a blokk olvasása szalagról
F732  68          PLA
F733  85 93       STA   93        load/verify-kapcsoló visszahozatala
F735  B0 32       BCS   F769      ha hiba, akkor kész
F737  A0 00       LDY   #00
F739  B1 B2       LDA   (B2),Y    a fejtipus vizsgálata
F73B  C9 05       CMP   #05       EOT?
F73D  F0 2A       BEQ   F769
F73F  C9 01       CMP   #01       BASIC program?
F741  F0 08       BEQ   F74B
F743  C9 03       CMP   #03       gépi program?
F745  F0 04       BEQ   F74B
F747  C9 04       CMP   #04       adat?
F749  D0 E1       BNE   F72C      nem
F74B  AA          TAX
F74C  24 9D       BIT   9D        közvetlen mód?
F74E  10 17       BPL   F767      ha nem, akkor tovább
F750  A0 63       LDY   #63
F752  20 2F F1    JSR   F12F      a "found" kiírása
F755  A0 05       LDY   #05       a file-név offset
F757  B1 B2       LDA   (B2),Y
F759  20 D2 FF    JSR   FFD2      a file-név kiírása
F75C  C8          INY
F75D  C0 15       CPY   #15
F75F  D0 F6       BNE   F757
F761  A5 A1       LDA   A1        középért. time-byte betölt. az akku-ba
F763  20 E0 E4    JSR   E4E0      várakozás a C= bill. vagy az időciklusra
F766  EA          NOP
F767  18          CLC             hiba esetén C=1
F768  88          DEY
F769  60          RTS

*****************************     a fejléc generálása és felírása szalagra
```

```
F76A  85 9E        STA   9E          a fejtipus
F76C  20 D0 F7     JSR   F7D0        szalag-puffer címének beolvasása
F76F  90 5E        BCC   F7CF
F771  A5 C2        LDA   C2
F773  48           PHA               a kezdőcím
F774  A5 C1        LDA   C1
F776  48           PHA               és
F777  A5 AF        LDA   AF
F779  48           PHA               a végcím tárolása
F77A  A5 AE        LDA   AE
F77C  48           PHA
F77D  A0 BF        LDY   #BF         puffer hossza - 1
F77F  A9 20        LDA   #20
F781  91 B2        STA   (B2),Y      a szalag-puffer törlése
F783  88           DEY
F784  D0 FB        BNE   F781
F786  A5 9E        LDA   9E
F788  91 B2        STA   (B2),Y      a fejtipus
F78A  C8           INY
F78B  A5 C1        LDA   C1
F78D  91 B2        STA   (B2),Y      a kezdőcím
F78F  C8           INY
F790  A5 C2        LDA   C2
F792  91 B2        STA   (B2),Y
F794  C8           INY
F795  A5 AE        LDA   AE
F797  91 B2        STA   (B2),Y      a végcím
F799  C8           INY
F79A  A5 AF        LDA   AF
F79C  91 B2        STA   (B2),Y
F79E  C8           INY
F79F  84 9F        STY   9F
F7A1  A0 00        LDY   #00
F7A3  84 9E        STY   9E          a"file-név-hossz" számláló
F7A5  A4 9E        LDY   9E          állásának
F7A7  C4 B7        CPY   B7          összehasonlítása a hosszal
F7A9  F0 0C        BEQ   F7B7        ha nincs több betű, akkor tovább
F7AB  B1 BB        LDA   (BB),Y      a file-név beolvasása
F7AD  A4 9F        LDY   9F
F7AF  91 B2        STA   (B2),Y      beírása a fejlécbe
F7B1  E6 9E        INC   9E
F7B3  E6 9F        INC   9F
F7B5  D0 EE        BNE   F7A5
F7B7  20 D7 F7     JSR   F7D7        kezdő- és végcím a szalag-pufferba
F7BA  A9 69        LDA   #69
F7BC  85 AB        STA   AB          a fej,ill.adatblokk eő. összege = $69
F7BE  20 6B F8     JSR   F86B        a blokk felírása a szalagra
F7C1  A8           TAY
F7C2  68           PLA
F7C3  85 AE        STA   AE
F7C5  68           PLA               a végcím
F7C6  85 AF        STA   AF
F7C8  68           PLA               és
F7C9  85 C1        STA   C1
F7CB  68           PLA               a kezdőcím visszaolvasása
F7CC  85 C2        STA   C2
```



```
F7CE  98           TYA
F7CF  60           RTS

*****************************
F7D0  A6 B2        LDX   B2          a szalag-puffer kezdőcímének beolvasása
F7D2  A4 B3        LDY   B3
F7D4  C0 02        CPY   #02         a cím kisebb $2$$-nál?
F7D6  60           RTS

*****************************
F7D7  20 D0 F7     JSR   F7D0        a szalag-puffer címének beolvasása
F7DA  8A           TXA
F7DB  85 C1        STA   C1          kezdőcím=szalag-puffer kezdete
F7DD  18           CLC
F7DE  69 C0        ADC   #C0
F7E0  85 AE        STA   AE
F7E2  98           TYA               végcím=kezdőcím + hossz /192/
F7E3  85 C2        STA   C2
F7E5  69 00        ADC   #00
F7E7  85 AF        STA   AF
F7E9  60           RTS

*****************************
F7EA  20 2C F7     JSR   F72C        a szalag fejlécének keresése
F7ED  B0 1D        BCS   F80C        a köv. fejléc keresése
F7EF  A0 05        LDY   #05         ha EOT, akkor kész
F7F1  84 9F        STY   9F          a file-név offset
F7F3  A0 00        LDY   #00
F7F5  84 9E        STY   9E          "file-név-hossz" számláló
F7F7  C4 B7        CPY   B7          állásának összehas. keresett név hosszával
F7F9  F0 10        BEQ   F80B        ha azonos, megvan
F7FB  B1 BB        LDA   (BB),Y      a file-név betűinek
F7FD  A4 9F        LDY   9F
F7FF  D1 B2        CMP   (B2),Y      összehas. fejlécben levő file-névvel
F801  D0 E7        BNE   F7EA        ha nem azonos, következő fejre
F803  E6 9E        INC   9E
F805  E6 9F        INC   9F          a számláló állásának növelése
F807  A4 9E        LDY   9E
F809  D0 EC        BNE   F7F7        a további betűk összehasonlítása
F80B  18           CLC
F80C  60           RTS

*****************************
F80D  20 D0 F7     JSR   F7D0        a szalag-puffer mutató növelése
F810  E6 A6        INC   A6          szalag-puffer címének beolvasása
F812  A4 A6        LDY   A6          a mutató növelése
F814  C0 C0        CPY   #C0
F816  60           RTS               összehas. a maximális értékkel /192/

*****************************
F817  20 2E F8     JSR   F82E        várakozás a kazetta-billentyűre
F81A  F0 1A        BEQ   F836        a kazetta-billentyű lekérdezése
F81C  A0 1B        LDY   #1B         ha le van nyomva, akkor kész
F81E  20 2F F1     JSR   F12F        "press play on tape" - offset
F821  20 D0 F8     JSR   F8D0        kiírása
F824  20 2E F8     JSR   F82E        a stop-billentyű vizsgálata
                                     a kazetta-billentyű lekérdezése
```

```
F827  D0 F8        BNE   F821
F829  A0 6A        LDY   #6A
F82B  4C 2F F1     JMP   F12F

*******************************
F82E  A9 10        LDA   #10
F830  24 01        BIT   01
F832  D0 02        BNE   F836
F834  24 01        BIT   01
F836  18           CLC
F837  60           RTS

*******************************
F838  20 2E F8     JSR   F82E
F83B  F0 F9        BEQ   F836
F83D  A0 2E        LDY   #2E
F83F  D0 DD        BNE   F81E

*******************************
F841  A9 00        LDA   #00
F843  85 90        STA   90
F845  85 93        STA   93
F847  20 D7 F7     JSR   F7D7

*******************************
F84A  20 17 F8     JSR   F817
F84D  B0 1F        BCS   F86E
F84f  78           SEI
F850  A9 00        LDA   #00
F852  85 AA        STA   AA
F854  85 B4        STA   B4
F856  85 B0        STA   B0
F858  85 9E        STA   9E
F85A  85 9F        STA   9F
F85C  85 9C        STA   9C
F85E  A9 90        LDA   #90
F860  A2 0E        LDX   #0E
F862  D0 11        BNE   F875

*******************************
F864  20 D7 F7     JSR   F7D7
F867  A9 14        LDA   #14
F869  85 AB        STA   AB

*******************************
F86B  20 38 F8     JSR   F838
F86E  B0 6C        BCS   F8DC
F870  78           SEI
F871  A9 82        LDA   #82
F873  A2 08        LDX   #08
F875  A0 7F        LDY   #7F
F877  8C 0D DC     STY   DC0D
F87A  8D 0D DC     STA   DC0D
F87D  AD 0E DC     LDA   DC0E
F880  09 19        ORA   #19
F882  8D 0F DC     STA   DC0F
```

"ok"- offset
kiírása

a kazetta-billentyű lenyomva?
a 4.bit vizsgálata
a billentyű lenyomva?
nem
újrakérdezés
ha igen, akkor Z=1, egyébként Z=∅

várakozás írásra
a kazetta-billentyű lekérdezése
ha le van nyomva, akkor kész
"press record - play on tape"- offset
tovább, a fentiek szerint

a blokk olvasása a szalagról

a státusz és
a verify-kapcsoló törlése
a szalag-puffer címének beolvasása

a program betöltése szalagról
várakozás a play-re
le van nyomva a stop-billentyű?

munkaterület az IRQ rutin törlésére

IRQ a "flag" lábon
az IRQ vektor sorszáma, ∅F92C

a puffer felírása a szalagra
a szalag-puffer címének beolvasása
adatblokkfej WRITE-hoz

a blokk vagy program felírása szalagra
várakozás a Record - Play-billentyűre
a stop-billentyű le van nyomva?

IRQ a B timer aláfutásakor
az IRQ-vektor sorszáma, ∅FC6A

az IRQ-maszk törlése
és újrabeállítása
CRA
a B timer betöltése, egy ütem, a
CRB, és IRQ indítása


```
F8B5  29 91        AND   #91
F8B7  8D A2 02     STA   02A2
F8BA  20 A4 F0     JSR   F0A4
F8BD  AD 11 D0     LDA   D011
F8B0  29 EF        AND   #EF
F8B2  8D 11 D0     STA   D011
F8B5  AD 14 03     LDA   0314
F8B8  8D 9F 02     STA   029F
F8BB  AD 15 03     LDA   0315
F8BE  8D A0 02     STA   02A0
F8A1  20 BD FC     JSR   FCBD
F8A4  A9 02        LDA   #02
F8A6  85 BE        STA   BE
F8A8  20 97 FB     JSR   FB97
F8AB  A5 01        LDA   01
F8AD  29 1F        AND   #1F
F8AF  85 01        STA   01
F8B1  85 C0        STA   C0
F8B3  A2 FF        LDX   #FF
F8B5  A0 FF        LDY   #FF
F8B7  88           DEY
F8B8  D0 FD        BNE   F8B7
F8BA  CA           DEX
F8BB  D0 F8        BNE   F8B5
F8BD  58           CLI

*******************************
F8BE  AD A0 02     LDA   02A0
F8C1  CD 15 03     CMP   0315
F8C4  18           CLC
F8C5  F0 15        BEQ   F8DC
F8C7  20 D0 F8     JSR   F8D0
F8CA  20 BC F6     JSR   F6BC
F8CD  4C BE F8     JMP   F8BE

*******************************
F8D0  20 E1 FF     JSR   FFE1
F8D3  18           CLC
F8D4  D0 0B        BNE   F8E1
F8D6  20 93 FC     JSR   FC93
F8D9  38           SEC
F8DA  68           PLA
F8DB  68           PLA
F8DC  A9 00        LDA   #00
F8DE  8D A0 02     STA   02A0
F8E1  60           RTS

*******************************
F8E2  86 B1        STX   B1
F8E4  A5 B0        LDA   B0
F8E6  0A           ASL   A
F8E7  0A           ASL   A
F8E8  18           CLC
F8E9  65 B0        ADC   B0
F8EB  18           CLC
F8EC  65 B1        ADC   B1
```

A timer ellenőrző kapcs.
várakozás az RS232 átvitel végére

a képernyő sötét

az IRQ-vektor

tárolása a ∅29F/∅2A∅-ra

I/O szalag IRQ-vekt. beállítása /X-index./

blokkok számának olvasása
a soros olv. előkész. bitszámláló beáll.

a motor bekapcsolása

a motor kapcsoló beállítása

a felfutási idő késleltetési ciklusa

I/O szalag megszak. felszabadítása

I/O lezárás kivárása
az IRQ vektor ismét alapállapotban?

ha igen, akkor kész
a stop-billentyű vizsgálata
ha le van nyomva, kapcsoló beállítása
várakozás tovább

a stop-billentyű vizsgálata
a stop-billentyű lekérdezése

ha nem, akkor visszatérés
a motor ki , IRQ alapértéke
a megszakítás jele

a visszaugrási cím törlése

a normál IRQ jelzése

a szalag előkészítése olvasásra

```
F8EE  85 B1      STA  B1
F8F0  A9 00      LDA  #00
F8F2  24 B0      BIT  B0
F8F4  30 01      BMI  F8F7
F8F6  2A         ROL  A
F8F7  06 B1      ASL  B1
F8F9  2A         ROL  A
F8FA  06 B1      ASL  B1
F8FC  2A         ROL  A
F8FD  AA         TAX
F8FE  AD 06 DC   LDA  DC06      B timer LO
F901  C9 16      CMP  #16
F903  90 F9      BCC  F8FE
F905  65 B1      ADC  B1
F907  8D 04 DC   STA  DC04      A timer LO
F90A  8A         TXA
F90B  6D 07 DC   ADC  DC07      B timer HI
F90E  8D 05 DC   STA  DC05      A timer HI
F911  AD A2 02   LDA  02A2
F914  8D 0E DC   STA  DC0E
F917  8D A4 02   STA  02A4
F91A  AD 0D DC   LDA  DC0D      input a kazettás egységről?
F91D  29 10      AND  #10       a bit leválasztása
F91F  F0 09      BEQ  F92A
F921  A9 F9      LDA  #F9
F923  48         PHA            a verem visszaugrási címe
F924  A9 2A      LDA  #2A
F926  48         PHA
F927  4C 43 FF   JMP  FF43      ugrás megszakításhoz
F92A  58         CLI
F92B  60         RTS

*****************************  a szalagolvasás megszakítása
F92C  AE 07 DC   LDX  DC07      B timer HI
F92F  A0 FF      LDY  #FF
F931  98         TYA
F932  ED 06 DC   SBC  DC06      B timer LO
F935  EC 07 DC   CPX  DC07      B timer hI
F938  D0 F2      BNE  F92C
F93A  86 B1      STX  B1
F93C  AA         TAX
F93D  8C 06 DC   STY  DC06      B timer LO
F940  8C 07 DC   STY  DC07      B timer hI
F943  A9 19      LDA  #19
F945  8D 0F DC   STA  DC0F      a B timer IRQ-ja
F948  AD 0D DC   LDA  DC0D      input szalagról, a "flag" láb
F94B  8D A3 02   STA  02A3
F94E  98         TYA
F94F  E5 B1      SBC  B1
F951  86 B1      STX  B1
F953  4A         LSR  A
F954  66 B1      ROR  B1
F956  4A         LSR  A
F957  66 B1      ROR  B1
F959  A5 B0      LDA  B0
F95B  18         CLC
```



```
F95C  69 3C      ADC  #3C
F95E  C5 B1      CMP  B1
F960  B0 4A      BCS  F9AC
F962  A6 9C      LDX  9C
F964  F0 03      BEQ  F969
F966  4C 60 FA   JMP  FA60
F969  A6 A3      LDX  A3
F96B  30 1B      BMI  F988
F96D  A2 00      LDX  #00
F96F  69 30      ADC  #30
F971  65 B0      ADC  B0
F973  C5 B1      CMP  B1
F975  B0 1C      BCS  F993
F977  E8         INX
F978  69 26      ADC  #26
F97A  65 B0      ADC  B0
F97C  C5 B1      CMP  B1
F97E  B0 17      BCS  F997
F980  69 2C      ADC  #2C
F982  65 B0      ADC  B0
F984  C5 B1      CMP  B1
F986  90 03      BCC  F98B
F988  4C 10 FA   JMP  FA10
F98B  A5 B4      LDA  B4
F98D  F0 1D      BEQ  F9AC
F98F  85 A8      STA  A8
F991  D0 19      BNE  F9AC
F993  E6 A9      INC  A9
F995  B0 02      BCS  F999
F997  C6 A9      DEC  A9
F999  38         SEC
F99A  E9 13      SBC  #13
F99C  E5 B1      SBC  B1
F99E  65 92      ADC  92
F9A0  85 92      STA  92
F9A2  A5 A4      LDA  A4
F9A4  49 01      EOR  #01
F9A6  85 A4      STA  A4
F9A8  F0 2B      BEQ  F9D5
F9AA  86 D7      STX  D7
F9AC  A5 B4      LDA  B4
F9AE  F0 22      BEQ  F9D2
F9B0  AD A3 02   LDA  02A3
F9B3  29 01      AND  #01
F9B5  D0 05      BNE  F9BC
F9B7  AD A4 02   LDA  02A4
F9BA  D0 16      BNE  F9D2
F9BC  A9 00      LDA  #00
F9BE  85 A4      STA  A4
F9C0  8D A4 02   STA  02A4
F9C3  A5 A3      LDA  A3
F9C5  10 30      BPL  F9F7
F9C7  30 BF      BMI  F988
F9C9  A2 A6      LDX  #A6
F9CB  20 E2 F8   JSR  F8E2
F9CE  A5 9B      LDA  9B
```

```
F9D0  D0 B9      BNE  F98B
F9D2  4C BC FE   JMP  FEBC     visszatérés a megszakításból
F9D5  A5 92      LDA  92
F9D7  F0 07      BEQ  F9E0
F9D9  30 03      BMI  F9DE
F9DB  C6 B0      DEC  B0
F9DD  2C
F9DE  E6 B0      INC  B0
F9E0  A9 00      LDA  #00
F9E2  85 92      STA  92
F9E4  E4 D7      CPX  D7
F9E6  D0 0F      BNE  F9F7
F9E8  8A         TXA
F9E9  D0 A0      BNE  F98B
F9EB  A5 A9      LDA  A9
F9ED  30 BD      BMI  F9AC
F9EF  C9 10      CMP  #10
F9F1  90 B9      BCC  F9AC
F9F3  85 96      STA  96
F9F5  B0 B5      BCS  F9AC
F9F7  8A         TXA
F9F8  45 9B      EOR  9B
F9FA  85 9B      STA  9B
F9FC  A5 B4      LDA  B4
F9FE  F0 D2      BEQ  F9D2
FA00  C6 A3      DEC  A3
FA02  30 C5      BMI  F9C9
FA04  46 D7      LSR  D7
FA06  66 BF      ROR  BF
FA08  A2 DA      LDX  #DA
FA0A  20 E2 F8   JSR  F8E2
FA0D  4C BC FE   JMP  FEBC     visszatérés a megszakításból
FA10  A5 96      LDA  96
FA12  F0 04      BEQ  FA18
FA14  A5 B4      LDA  B4
FA16  F0 07      BEQ  FA1F
FA18  A5 A3      LDA  A3
FA1A  30 03      BMI  FA1F
FA1C  4C 97 F9   JMP  F997
FA1F  46 B1      LSR  B1
FA21  A9 93      LDA  #93
FA23  38         SEC
FA24  E5 B1      SBC  B1
FA26  65 B0      ADC  B0
FA28  0A         ASL  A
FA29  AA         TAX
FA2A  20 E2 F8   JSR  F8E2
FA2D  E6 9C      INC  9C
FA2F  A5 B4      LDA  B4
FA31  D0 11      BNE  FA44
FA33  A5 96      LDA  96
FA35  F0 26      BEQ  FA5D
FA37  85 A8      STA  A8
FA39  A9 00      LDA  #00
FA3B  85 96      STA  96
FA3D  A9 81      LDA  #81
```



```
FA3F  8D 0D DC   STA  DC0D     IRQ az A timer aláfutásanál
FA42  85 B4      STA  B4
FA44  A5 96      LDA  96
FA46  85 B5      STA  B5
FA48  F0 09      BEQ  FA53
FA4A  A9 00      LDA  #00
FA4C  85 B4      STA  B4
FA4E  A9 01      LDA  #01
FA50  8D 0D DC   STA  DC0D     az IRQ-kapcsoló újratöltése
FA53  A5 BF      LDA  BF
FA55  85 BD      STA  BD
FA57  A5 A8      LDA  A8
FA59  05 A9      ORA  A9
FA5B  85 B6      STA  B6
FA5D  4C BC FE   JMP  FEBC     visszatérés a megszakításból
FA60  20 97 FB   JSR  FB97     soros kihoz. bitszámlálójának beáll.
FA63  85 9C      STA  9C
FA65  A2 DA      LDX  #DA
FA67  20 E2 F8   JSR  F8E2
FA6A  A5 BE      LDA  BE
FA6C  F0 02      BEQ  FA70
FA6E  85 A7      STA  A7
FA70  A9 0F      LDA  #0F
FA72  24 AA      BIT  AA
FA74  10 17      BPL  FA8D
FA76  A5 B5      LDA  B5
FA78  D0 0C      BNE  FA86
FA7A  A6 BE      LDX  BE
FA7C  CA         DEX
FA7D  D0 0B      BNE  FA8A
FA7F  A9 08      LDA  #08      "long block" error
FA81  20 1C FE   JSR  FE1C     a státusz beállítása
FA84  D0 04      BNE  FA8A
FA86  A9 00      LDA  #00
FA88  85 AA      STA  AA
FA8A  4C BC FE   JMP  FEBC     visszatérés a megszakításból
FA8D  70 31      BVS  FAC0
FA8F  D0 18      BNE  FAA9
FA91  A5 B5      LDA  B5
FA93  D0 F5      BNE  FA8A
FA95  A5 B6      LDA  B6
FA97  D0 F1      BNE  FA8A
FA99  A5 A7      LDA  A7
FA9B  4A         LSR  A
FA9C  A5 BD      LDA  BD
FA9E  30 03      BMI  FAA3
FAA0  90 18      BCC  FABA
FAA2  18         CLC
FAA3  B0 15      BCS  FABA
FAA5  29 0F      AND  #0F
FAA7  85 AA      STA  AA
FAA9  C6 AA      DEC  AA
FAAB  D0 DD      BNE  FA8A
FAAD  A9 40      LDA  #40
FAAF  85 AA      STA  AA
FAB1  20 8E FB   JSR  FB8E
```

```
FAB4  A9 00       LDA  #00
FAB6  85 AB       STA  AB
FAB8  F0 D0       BEQ  FA8A
FABA  A9 80       LDA  #80
FABC  85 AA       STA  AA
FABE  D0 CA       BNE  FA8A
FAC0  A5 B5       LDA  B5
FAC2  F0 0A       BEQ  FACE
FAC4  A9 04       LDA  #04
FAC6  20 1C FE    JSR  FE1C
FAC9  A9 00       LDA  #00
FACB  4C 4A FB    JMP  FB4A
FACE  20 D1 FC    JSR  FCD1
FAD1  90 03       BCC  FAD6
FAD3  4C 48 FB    JMP  FB48
FAD6  A6 A7       LDX  A7
FAD8  CA          DEX
FAD9  F0 2D       BEQ  FB08
FADB  A5 93       LDA  93
FADD  F0 0C       BEQ  FAEB
FADF  A0 00       LDY  #00
FAE1  A5 BD       LDA  BD
FAE3  D1 AC       CMP  (AC),Y
FAE5  F0 04       BEQ  FAEB
FAE7  A9 01       LDA  #01
FAE9  85 B6       STA  B6
FAEB  A5 B6       LDA  B6
FAED  F0 4B       BEQ  FB3A
FAEF  A2 3D       LDX  #3D
FAF1  E4 9E       CPX  9E
FAF3  90 3E       BCC  FB33
FAF5  A6 9E       LDX  9E
FAF7  A5 AD       LDA  AD
FAF9  9D 01 01    STA  0101,X
FAFC  A5 AC       LDA  AC
FAFE  9D 00 01    STA  0100,X
FB01  E8          INX
FB02  E8          INX
FB03  86 9E       STX  9E
FB05  4C 3A FB    JMP  FB3A
FB08  A6 9F       LDX  9F
FB0A  E4 9E       CPX  9E
FB0C  F0 35       BEQ  FB43
FB0E  A5 AC       LDA  AC
FB10  DD 00 01    CMP  0100,X
FB13  D0 2E       BNE  FB43
FB15  A5 AD       LDA  AD
FB17  DD 01 01    CMP  0101,X
FB1A  D0 27       BNE  FB43
FB1C  E6 9F       INC  9F
FB1E  E6 9F       INC  9F
FB20  A5 93       LDA  93
FB22  F0 0B       BEQ  FB2F
FB24  A5 BD       LDA  BD
FB26  A0 00       LDY  #00
FB28  D1 AC       CMP  (AC),Y
```

"short block" error
a sátusz beállítása

már a végcím?
nem

load/verify-kapcsoló
load?

az olvasott byte
összehasonlítása
egyezik?

a kapcsoló beállítása

a byte rendben van?

adatok a második menethez

hibajavítás a 2. menetben

a javítás-számláló növelése 2-vel
Verify?
nem
az olvasott byte

összehasonlítása a tártartalommal


```
FB2A  F0 17       BEQ  FB43
FB2C  C8          INY
FB2D  84 B6       STY  B6
FB2F  A5 B6       LDA  B6
FB31  F0 07       BEQ  FB3A
FB33  A9 10       LDA  #10
FB35  20 1C FE    JSR  FE1C
FB38  D0 09       BNE  FB43
FB3A  A5 93       LDA  93
FB3C  D0 05       BNE  FB43
FB3E  A8          TAY
FB3F  A5 BD       LDA  BD
FB41  91 AC       STA  (AC),Y
FB43  20 DB FC    JSR  FCDB
FB46  D0 43       BNE  FB8B
FB48  A9 80       LDA  #80
FB4A  85 AA       STA  AA
FB4C  78          SEI
FB4D  A2 01       LDX  #01
FB4F  8E 0D DC    STX  DC0D
FB52  AE 0D DC    LDX  DC0D
FB55  A6 BE       LDX  BE
FB57  CA          DEX
FB58  30 02       BMI  FB5C
FB5A  86 BE       STX  BE
FB5C  C6 A7       DEC  A7
FB5E  F0 08       BEQ  FB68
FB60  A5 9E       LDA  9E
FB62  D0 27       BNE  FB8B
FB64  85 BE       STA  BE
FB66  F0 23       BEQ  FB8B
FB68  20 93 FC    JSR  FC93
FB6B  20 8E FB    JSR  FB8E
FB6E  A0 00       LDY  #00
FB70  84 AB       STY  AB
FB72  B1 AC       LDA  (AC),Y
FB74  45 AB       EOR  AB
FB76  85 AB       STA  AB
FB78  20 DB FC    JSR  FCDB
FB7B  20 D1 FC    JSR  FCD1
FB7E  90 F2       BCC  FB72
FB80  A5 AB       LDA  AB
FB82  45 BD       EOR  BD
FB84  F0 05       BEQ  FB8B
FB86  A9 20       LDA  #20
FB88  20 1C FE    JSR  FE1C
FB8B  4C BC FE    JMP  FEBC
FB8E  A5 C2       LDA  C2
FB90  85 AD       STA  AD
FB92  A5 C1       LDA  C1
FB94  85 AC       STA  AC
FB96  60          RTS

*****************************
FB97  A9 08       LDA  #08
FB99  85 A3       STA  A3
```

ha egyenlő, akkor tovább

a hibakapcsoló beállítása
a hibakapcsoló vizsgálata
nem volt hiba?
"second pass" error
a státusz beállítása
a következő byte
verify?
igen

az olvasott byte
tárolása
a címregiszter növelése
visszatérés a megszakításból

kapcsoló az olvasás végére

az A timer IRQ-jának letiltása
az IRQ kapcsoló törlése
a menetszámláló
csökkentése

tárolása
volt hiba a menetben?

volt hiba az 1. menetben?
igen, visszatérés a megszakításból
nincs több adatblokk
visszatérés a megszakításból
egy menet végetér
a cím ismét a programkezdetre

a program-ellenőrző összeg kiszámítása

és tárolása
a cím mutató növelése
már a végcím?
nem, folytatni az összehasonlítást
a kiszámított ellenőrzési összeg
összehasonlítása a szalagról olv. eő.össz.-el
ellenőrzési összeg rendben?
"checksum" error
a státusz beállítása
visszatérés a megszakításból

kezdőcím: $C1/$C2-ből
a $AC/$AD címekre

a bitszáml. beáll. soros kivitelhez

8 bit

```
FB9B  A9 00      LDA  #00
FB9D  85 A4      STA  A4
FB9F  85 A8      STA  A8
FBA1  85 9B      STA  9B
FBA3  85 A9      STA  A9
FBA5  60         RTS

*****************************
FBA6  A5 BD      LDA  BD
FBA8  4A         LSR  A
FBA9  A9 60      LDA  #60
FBAB  90 02      BCC  FBAF
FBAD  A9 B0      LDA  #B0
FBAF  A2 00      LDX  #00
FBB1  8D 06 DC   STA  DC06
FBB4  8E 07 DC   STX  DC07
FBB7  AD 0D DC   LDA  DC0D
FBBA  A9 19      LDA  #19
FBBC  8D 0F DC   STA  DC0F
FBBF  A5 01      LDA  01
FBC1  49 08      EOR  #08
FBC3  85 01      STA  01
FBC5  29 08      AND  #08
FBC7  60         RTS
FBC8  38         SEC
FBC9  66 B6      ROR  B6
FBCB  30 3C      BMI  FC09

*****************************
FBCD  A5 A8      LDA  A8
FBCF  D0 12      BNE  FBE3
FBD1  A9 10      LDA  #10
FBD3  A2 01      LDX  #01
FBD5  20 B1 FB   JSR  FBB1
FBD8  D0 2F      BNE  FC09
FBDA  E6 A8      INC  A8
FBDC  A5 B6      LDA  B6
FBDE  10 29      BPL  FC09
FBE0  4C 57 FC   JMP  FC57
FBE3  A5 A9      LDA  A9
FBE5  D0 09      BNE  FBF0
FBE7  20 AD FB   JSR  FBAD
FBEA  D0 1D      BNE  FC09
FBEC  E6 A9      INC  A9
FBEE  D0 19      BNE  FC09
FBF0  20 A6 FB   JSR  FBA6
FBF3  D0 14      BNE  FC09
FBF5  A5 A4      LDA  A4
FBF7  49 01      EOR  #01
FBF9  85 A4      STA  A4
FBFB  F0 0F      BEQ  FC0C
FBFD  A5 BD      LDA  BD
FBFF  49 01      EOR  #01
FC01  85 BD      STA  BD
FC03  29 01      AND  #01
FC05  45 9B      EOR  9B
```

a munkaterület törlése

1 bit felírása a szalagra
a bit a $BD-ben
a Ø. bit a carry-be
a "Ø" bit ideje

az "1" bit ideje

B timer alsó
B timer felső
a megszakítás kapcsoló törlése

a timer indítása

az adatbit invertálása

az aktuális jelszint tárolása

visszatérés a megszakításból

megszakítás szalagra íráshoz

a timer $11Ø /272/

az ütem felírása a szalagra
visszatérés a megszakításból

visszatérés a megszakításból
a második blokk felírása

az "1" bit felírása
visszatérés a megszakításból

visszatérés a megszakításból
a bit felírása a szalagra
visszatérés a megszakításból

a kiviteli bit megfordítása


```
FC07  85 9B      STA  9B
FC09  4C BC FE   JMP  FEBC
FC0C  46 BD      LSR  BD
FC0E  C6 A3      DEC  A3
FC10  A5 A3      LDA  A3
FC12  F0 3A      BEQ  FC4E
FC14  10 F3      BPL  FC09
FC16  20 97 FB   JSR  FB97
FC19  58         CLI
FC1A  A5 A5      LDA  A5
FC1C  F0 12      BEQ  FC30
FC1E  A2 00      LDX  #00
FC20  86 D7      STX  D7
FC22  C6 A5      DEC  A5
FC24  A6 BE      LDX  BE
FC26  E0 02      CPX  #02
FC28  D0 02      BNE  FC2C
FC2A  09 80      ORA  #80
FC2C  85 BD      STA  BD
FC2E  D0 D9      BNE  FC09
FC30  20 D1 FC   JSR  FCD1
FC33  90 0A      BCC  FC3F
FC35  D0 91      BNE  FBC8
FC37  E6 AD      INC  AD
FC39  A5 D7      LDA  D7
FC3B  85 BD      STA  BD
FC3D  B0 CA      BCS  FC09
FC3F  A0 00      LDY  #00
FC41  B1 AC      LDA  (AC),Y
FC43  85 BD      STA  BD
FC45  45 D7      EOR  D7
FC47  85 D7      STA  D7
FC49  20 DB FC   JSR  FCDB
FC4C  D0 BB      BNE  FC09
FC4E  A5 9B      LDA  9B
FC50  49 01      EOR  #01
FC52  85 BD      STA  BD
FC54  4C BC FE   JMP  FEBC
FC57  C6 BE      DEC  BE
FC59  D0 03      BNE  FC5E
FC5B  20 CA FC   JSR  FCCA
FC5E  A9 50      LDA  #50
FC60  85 A7      STA  A7
FC62  A2 08      LDX  #08
FC64  78         SEI
FC65  20 BD FC   JSR  FCBD
FC68  D0 EA      BNE  FC54

*****************************
FC6A  A9 78      LDA  #78
FC6C  20 AF FB   JSR  FBAF
FC6F  D0 E3      BNE  FC54
FC71  C6 A7      DEC  A7
FC73  D0 DF      BNE  FC54
FC75  20 97 FB   JSR  FB97
FC78  C6 AB      DEC  AB
```

visszatérés a megszakításból
következő bit a Ø. pozicióra
a bitszámláló csökkentése

következő bit kiírása
visszatérés a megszakításból
a bitszámláló visszaállítása 8-ra

visszatérés a megszakításból
már a végcím?

visszatérés a megszakításból

a kiírandó byte

a címmutató növelése
visszatérés a megszakításból

visszatérés a megszakításból
a blokkszámláló csökkentése
még egy blokk?
nem, a motor kikapcsolása
8Ø

Irás a $FC6A-ra
visszatérés a megszakításból

megszakítás a szalagra íráshoz
12Ø
a bit felírása a szalagra
visszatérés a megszakításból
a számláló csökkentése, ha nem Ø
visszatérés a megszakításból
soros kiírás számlálójának beállítása

```
FC7A  10 D8       BPL   FC54      visszatérés a megszakításból
FC7C  A2 0A       LDX   #0A
FC7E  20 BD FC    JSR   FCBD      IRQ a $FBCD-re
FC81  58          CLI
FC82  E6 AB       INC   AB
FC84  A5 BE       LDA   BE        a blokkok száma
FC86  F0 30       BEQ   FCB8      már minden blokk fel van írva?
FC88  20 8E FB    JSR   FB8E      cím visszaállítása a kezdetre
FC8B  A2 09       LDX   #09
FC8D  86 A5       STX   A5        számláló a szinkronjelhez
FC8F  86 B6       STX   B6        flag a blokk írására
FC91  D0 83       BNE   FC16
FC93  08          PHP             a státusz tárolása
FC94  78          SEI
FC95  AD 11 D0    LDA   D011
FC98  09 10       ORA   #10       a képernyő újrabekapcsolása
FC9A  8D 11 D0    STA   D011
FC9D  20 CA FC    JSR   FCCA      a motor kikapcsolása
FCA0  A9 7F       LDA   #7F
FCA2  8D 0D DC    STA   DC0D      a megszakítás letiltása
FCA5  20 DD FD    JSR   FDDD      CIA ismét az alapértékekre, 1/60 s idő
FCA8  AD A0 02    LDA   02A0      a megszak.vektor alapértékeken?
FCAB  F0 09       BEQ   FCB6      ha igen, akkor kész
FCAD  8D 15 03    STA   0315
FCB0  AD 9F 02    LDA   029F      IRQ visszatöltése
FCB3  8D 14 03    STA   0314
FCB6  28          PLP             a státusz visszatöltése
FCB7  60          RTS

*****************************
FCB8  20 93 FC    JSR   FC93      az IRQ vektor beállítása, X-el indexelt
FCBB  F0 97       BEQ   FC54      az IRQ alapértékre
FCBD  BD 93 FD    LDA   FD93,X
FCC0  8D 14 03    STA   0314      az IRQ vektor beáll. a táblázatból
FCC3  BD 94 FD    LDA   FD94,X
FCC6  8D 15 03    STA   0315
FCC9  60          RTS

*****************************
FCCA  A5 01       LDA   01
FCCC  09 20       ORA   #20       a motor kikapcsolása
FCCE  85 01       STA   01
FCD0  60          RTS

*****************************
FCD1  38          SEC             a végcím elérésének ellenőrzése
FCD2  A5 AC       LDA   AC        az aktuális cím $AC/$AD
FCD4  E5 AE       SBC   AE
FCD6  A5 AD       LDA   AD        a végcím $AE/$AF
FCD8  E5 AF       SBC   AF
FCDA  60          RTS

*****************************
FCDB  E6 AC       INC   AC        a címmutató növelése
FCDD  D0 02       BNE   FCE1
FCDF  E6 AD       INC   AD
```



```
FCE1  60          RTS

*****************************
FCE2  A2 FF       LDX   #FF       RESET
FCE4  78          SEI
FCE5  9A          TXS
FCE6  D8          CLD
FCE7  20 02 FD    JSR   FD02      ellenőrz. a ROM-ban /$8000/
FCEA  D0 03       BNE   FCEF      nincs autostart modul?
FCEC  6C 00 80    JMP   (8000)    ugrás a modul-startra
FCEF  8E 16 D0    STX   D016      videovezérlő - 2.vezérlőregiszter
FCF2  20 A3 FD    JSR   FDA3      a megszakítás előkészítése
FCF5  20 50 FD    JSR   FD50      a munkaterület inicializálása
FCF8  20 15 FD    JSR   FD15      a hardver és az I/O-vektorok beáll.
FCFB  20 5B FF    JSR   FF5B      Video-Reset
FCFE  58          CLI
FCFF  6C 00 A0    JMP   (A000)    BASIC-hidegindításhoz

*****************************
FD02  A2 05       LDX   #05       a ROM ellenőrzése a $8000-en
FD04  BD 0F FD    LDA   FD0F,X
FD07  DD 03 80    CMP   8003,X    összehasonlítás a "CBM80"-nal
FD0A  D0 03       BNE   FD0F
FD0C  CA          DEX
FD0D  D0 F5       BNE   FD04
FD0F  60          RTS

*****************************
FD10  C3 C2 CD 38 30              a ROM modul azonosítása
                                  "CBM80"

*****************************
FD15  A2 30       LDX   #30       a hardver és az I/O-vektorok beáll./beolv.
FD17  A0 FD       LDY   #FD       mutató a táblázatra - $FD30
FD19  18          CLC
FD1A  86 C3       STX   C3
FD1C  84 C4       STY   C4
FD1E  A0 1F       LDY   #1F
FD20  B9 14 03    LDA   0314,Y
FD23  B0 02       BCS   FD27      ha C=1, akkor a vektorok betöltése és C=0
FD25  B1 C3       LDA   (C3),Y
FD27  91 C3       STA   (C3),Y
FD29  99 14 03    STA   0314,Y
FD2C  88          DEY
FD2D  10 F1       BPL   FD20
FD2F  60          RTS

*****************************
FD30  31 EA 66 FE 47 FE 4A F3     a hardver-táblázat és I/O-vektorok
FD38  91 F2 0E F2 50 F2 33 F3
FD40  57 F1 CA F1 ED F6 3E F1
FD48  2F F3 66 FE A5 F4 ED F5

*****************************
FD50  A9 00       LDA   #00       a munkaterület inicializálása
FD52  A8          TAY
FD53  99 02 00    STA   0002,Y    nullás lap
```

```
FD56  99 00 02    STA   0200,Y      a 2. oldal és
FD59  99 00 03    STA   0300,Y      a 3. oldal törlése
FD5C  C8          INY
FD5D  D0 F4       BNE   FD53
FD5F  A2 3C       LDX   #3C
FD61  A0 03       LDY   #03
FD63  86 B2       STX   B2          szalag-puffer mutató $033C
FD65  84 B3       STY   B3
FD67  A8          TAY
FD68  A9 03       LDA   #03
FD6A  85 C2       STA   C2          a RAM vizsgálata $4000-tól kezdődően
FD6C  E6 C2       INC   C2
FD6E  B1 C1       LDA   (C1),Y
FD70  AA          TAX               tárolás
FD71  A9 55       LDA   #55         %01010101
FD73  91 C1       STA   (C1),Y
FD75  D1 C1       CMP   (C1),Y
FD77  D0 0F       BNE   FD88
FD79  2A          ROL   A           %10101010
FD7A  91 C1       STA   (C1),Y
FD7C  D1 C1       CMP   (C1),Y
FD7E  D0 08       BNE   FD88
FD80  8A          TXA               az érték visszaírása
FD81  91 C1       STA   (C1),Y
FD83  C8          INY
FD84  D0 E8       BNE   FD6E
FD86  F0 E4       BEQ   FD6C
FD88  98          TYA
FD89  AA          TAX
FD8A  A4 C2       LDY   C2
FD8C  18          CLC
FD8D  20 2D FE    JSR   FE2D        a RAM-tető beállítása
FD90  A9 08       LDA   #08         a RAM-kezdet $8000-ra
FD92  8D 82 02    STA   0282
FD95  A9 04       LDA   #04
FD97  8D 88 02    STA   0288        a video-RAM $4000-ra
FD9A  60          RTS

*****************************       az IRQ vektorok
FD9B  6A FC CD FB 31 EA 2C F9       $FC6A, $FBCD, $EA31, $F92C

*****************************       a megszakítás inicializálása
FDA3  A9 7F       LDA   #7F         a megszakítás törlése
FDA5  8D 0D DC    STA   DC0D        ICR CIA 1.
FDA8  8D 0D DD    STA   DD0D        ICR CIA 2.
FDAB  8D 00 DC    STA   DC00        a port CIA 1. bill.-nulladik mátrixsor
FDAE  A9 08       LDA   #08
FDB0  8D 0E DC    STA   DC0E        a CIA 1. A timer CRA one shot
FDB3  8D 0E DD    STA   DD0E        a CIA 2. A timer CRA one shot
FDB6  8D 0F DC    STA   DC0F        a CIA 1. B timer CRB one shot
FDB9  8D 0F DD    STA   DD0F        a CIA 2. B timer CRB one shot
FDBC  A2 00       LDX   #00         bemenet
FDBE  8E 03 DC    STX   DC03        a CIA 1. B adatirányregisztere
FDC1  8E 03 DD    STX   DD03        a CIA 2. B adatirányregisztere
FDC4  8E 18 D4    STX   D418        a SID hangerősségének nullára állítása
FDC7  CA          DEX               kihozatal
```



```
FDC8  8E 02 DC    STX   DC02        A CIA 1. adatirányregiszter
FDCB  A9 07       LDA   #07        a videovezérlő a legalsó 16 K-n
FDCD  8D 00 DD    STA   DD00        A CIA 2. A port ATN törlése
FDD0  A9 3F       LDA   #3F        0-5. bitek kiírása
FDD2  8D 02 DD    STA   DD02        A CIA 2. adatirányregisztere
FDD5  A9 E7       LDA   #E7
FDD7  85 01       STA   01          a processzorport, tárfelosztás
FDD9  A9 2F       LDA   #2F         a 0-3. és 5. bitek ki., a 4. bit be
FDDB  85 00       STA   00          adatirány, processzorport
FDDD  AD A6 02    LDA   02A6        NTSC változat?
FDE0  F0 0A       BEQ   FDEC       igen
FDE2  A9 25       LDA   #25
FDE4  8D 04 DC    STA   DC04        a PAL-változat timer-nek beállítása
FDE7  A9 40       LDA   #40         $4025 = 16421 ciklus
FDE9  4C F3 FD    JMP   FDF3
FDEC  A9 95       LDA   #95
FDEE  8D 04 DC    STA   DC04        az NTSC változat timer-ének beállítása
FDF1  A9 42       LDA   #42         $4296 = 17045 ciklus
FDF3  8D 05 DC    STA   DC05        timer felső
FDF6  4C 6E FF    JMP   FF6E        a megszakítás beállítása

*****************************       a file-név paramétereinek beállítása
FDF9  85 B7       STA   B7          hossz
FDFB  86 BB       STX   BB          a cím alsó
FDFD  84 BC       STY   BC          a cím felső
FDFF  60          RTS

*****************************       az aktív file paramétereinek beállítása
FE00  85 B8       STA   B8          a logikai file-szám
FE02  86 BA       STX   BA          az egységszám
FE04  84 B9       STY   B9          a másodlagos cím
FE06  60          RTS

*****************************       a státusz beolvasása
FE07  A5 BA       LDA   BA          egységszám
FE09  C9 02       CMP   #02         2-vel egyenlő?
FE0B  D0 0D       BNE   FE1A        nem
FE0D  AD 97 02    LDA   0297        RS232 státuszának beolvasása
FE10  48          PHA
FE11  A9 00       LDA   #00         a státusz törlése
FE13  8D 97 02    STA   0297
FE16  68          PLA
FE17  60          RTS

*****************************       az op.rendszer üzeneteinek kapcsolója
FE18  85 9D       STA   9D
FE1A  A5 90       LDA   90
FE1C  05 90       ORA   90          a státus beállítása
FE1E  85 90       STA   90
FE20  60          RTS
FE21  8D 85 02    STA   0285        a timeout-flag beállítása az IEC-hez
FE24  60          RTS.
FE25  90 06       BCC   FE2D        MEMTOP a BASIC-RAM felső határ
FE27  AE 83 02    LDX   0283        a carry magas
FE2A  AC 84 02    LDY   0284        a cím az X/Y-ba
FE2D  8E 83 02    STX   0283        a carry törölve
```

```
FE30  8C 84 02     STY   0284          az X/Y címre
FE33  60           RTS

*****************************
FE34  90 06        BCC   FE3C          MEMBOT a BASIC-RAM alsó hat. beolv/beáll.
FE36  AE 81 02     LDX   0281          lásd fent
FE39  AC 82 02     LDY   0282
FE3C  8E 81 02     STX   0281
FE3F  8C 82 02     STY   0282
FE42  60           RTS

*****************************
FE43  78           SEI                 NMI - beugrás
FE44  6C 18 03     JMP   (0318)        JMP $FE47, NMI vektor
FE47  48           PHA
FE48  8A           TXA
FE49  48           PHA                 a regiszter tárolása
FE4A  98           TYA
FE4B  48           PHA
FE4C  A9 7F        LDA   #7F
FE4E  8D 0D DD     STA   DD0D          az interrupt letiltása
FE51  AC 0D DD     LDY   DD0D          a kapcsolók olvasása és törlése
FE54  30 1C        BMI   FE72          RS232 aktív?
FE56  20 02 FD     JSR   FD02          a ROM-modul a $8000-es címen történő eő.
FE59  D0 03        BNE   FE5E          nem,tovább
FE5B  6C 02 80     JMP   (8002)        igen, ugrás a NMI modulra
FE5E  20 BC F6     JSR   F6BC          a stop-billentyű kapcs. beállítása
FE61  20 E1 FF     JSR   FFE1          a stop-billentyű lekérdezése
FE64  D0 0C        BNE   FE72          nincs lenyomva
FE66  20 15 FD     JSR   FD15          az interrupt, I/O standardvektorok beáll.
FE69  20 A3 FD     JSR   FDA3          az I/O inicializálása
FE6C  20 18 E5     JSR   E518          az I/O inic. és a képernyő törlése
FE6F  6C 02 A0     JMP   (A002)        ugrás a BASIC melegindításhoz

*****************************
FE72  98           TYA                 RS232 NMI rutinja
FE73  2D A1 02     AND   02A1          az ICR regiszter
FE76  AA           TAX                 az RS232 NMI-kapcs. vizsgálata
FE77  29 01        AND   #01           továbbító üzem?
FE79  F0 28        BEQ   FEA3          nem
FE7B  AD 00 DD     LDA   DD00          az adat-port olvasása
FE7E  29 FB        AND   #FB           a 2. bit /TxD/ törlése
FE80  05 B5        ORA   B5            a bit továbbítása és
FE82  8D 00 DD     STA   DD00          tárolása az adat-porton
FE85  AD A1 02     LDA   02A1          az RS232 NMI-kapcsoló
FE88  8D 0D DD     STA   DD0D          beírása az ICR-be
FE8B  8A           TXA
FE8C  29 12        AND   #12           az 1. és 4. bit leválasztása
FE8E  F0 0D        BEQ   FE9D
FE90  29 02        AND   #02           az 1. bit hívás a B timerről
FE92  F0 06        BEQ   FE9A          nem, startbit
FE94  20 D6 FE     JSR   FED6          a bit feldolgozása
FE97  4C 9D FE     JMP   FE9D
FE9A  20 07 FF     JSR   FF07          előkészület a köv. byte fogadására
FE9D  20 BB EE     JSR   EEBB          a következő bit fogadása
```



```
FEA0  4C B6 FE     JMP   FEB6          visszatérés a megszakításból
FEA3  8A           TXA
FEA4  29 02        AND   #02           adatfogadás?
FEA6  F0 06        BEQ   FEAE          nem
FEA8  20 D6 FE     JSR   FED6          a bit feldolgozása
FEAB  4C B6 FE     JMP   FEB6          visszatérés a megszakításból
FEAE  8A           TXA
FEAF  29 10        AND   #10           várakozás a startbitre
FEB1  F0 03        BEQ   FEB6          nem
FEB3  20 07 FF     JSR   FF07          előkészület a köv. byte fogadására
FEB6  AD A1 02     LDA   02A1          az RS232 NMI-kapcsoló
FEB9  8D 0D DD     STA   DD0D          ismét az ICR-be
FEBC  68           PLA
FEBD  A8           TAY
FEBE  68           PLA                 a regiszterek visszaolvasása
FEBF  AA           TAX
FEC0  68           PLA
FEC1  40           RTI

*****************************
FEC2  C1 27                            az RS232 timer-állandói, baud-rate,NTSC-vált.
FEC4  3E 1A                            $27C1 = 10177      50baud
FEC6  C5 11                            $1A3E =  6718      75baud
FEC8  74 0E                            $11C5 =  4549     110baud
FECA  ED 0C                            $0E74 =  3700   134.5baud
FECC  45 06                            $0CED =  3309     150 baud
FECE  F0 02                            $0645 =  1605     300 baud
FED0  46 01                            $02F0 =   752     600 baud
FED2  B8 00                            $0146 =   326    1200 baud
FED4  71 00                            $00B8 =   184    1800 baud
                                       $0071 =   113    2400 baud

*****************************
FED6  AD 01 DD     LDA   DD01          NMI rutin az RS232-es bevitelhez
FED9  29 01        AND   #01           a B regiszter-port
FEDB  85 A7        STA   A7            a "Receive Data" bit leválasztása
FEDD  AD 06 DD     LDA   DD06          és tárolása
FEE0  E9 1C        SBC   #1C           B timer LO
FEE2  6D 99 02     ADC   0299          minusz 28
FEE5  8D 06 DD     STA   DD06
FEE8  AD 07 DD     LDA   DD07          az RS232 baud-rate timer-állandói
FEEB  6D 9A 02     ADC   029A
FEEE  8D 07 DD     STA   DD07          beírás a timer-be
FEF1  A9 11        LDA   #11           a B timer indítása
FEF3  8D 0F DD     STA   DD0F          a B vezérlőregiszter
FEF6  AD A1 02     LDA   02A1
FEF9  8D 0D DD     STA   DD0D          interrupt control register
FEFC  A9 FF        LDA   #FF
FEFE  8D 06 DD     STA   DD06          a timer beállítása
FF01  8D 07 DD     STA   DD07          a bit beolvasása
FF04  4C 59 EF     JMP   EF59

*****************************
FF07  AD 95 02     LDA   0295          NMI rutin az RS232-es outputhoz
FF0A  8D 06 DD     STA   DD06
FF0D  AD 96 02     LDA   0296          az RS232 - baud rate - timer-állandók
FF10  8D 07 DD     STA   DD07
```

```
FF13  A9 11      LDA  #11      a B timer indítása
FF15  8D 0F DD   STA  DD0F     a B vezérlőregiszter
FF18  A9 12      LDA  #12
FF1A  4D A1 02   EOR  02A1     a CIA 2. NMI kapcsolója
FF1D  8D A1 02   STA  02A1
FF20  A9 FF      LDA  #FF
FF22  8D 06 DD   STA  DD06     a timer betöltése
FF25  8D 07 DD   STA  DD07     a küldendő bitek száma
FF28  AE 98 02   LDX  0298
FF2B  86 A8      STX  A8
FF2D  60         RTS

*****************************  timer-érték továbbítása /baud-rate küldéshe
FF2E  AA         TAX
FF2F  AD 96 02   LDA  0296
FF32  2A         ROL  A        2-szer
FF33  A8         TAY
FF34  8A         TXA
FF35  69 C8      ADC  #C8      plusz 200
FF37  8D 99 02   STA  0299     timer érték LO
FF3A  98         TYA
FF3B  69 00      ADC  #00
FF3D  8D 9A 02   STA  029A     timer érték HI
FF40  60         RTS
FF41  EA         NOP
FF42  EA         NOP

*****************************  kiugrás a szalag-rutinból
FF43  08         PHP
FF44  68         PLA
FF45  29 EF      AND  #EF      break-kapcsoló törlése
FF47  48         PHA

*****************************  IRQ-beugrás /belépés/
FF48  48         PHA
FF49  8A         TXA
FF4A  48         PHA           a regiszter tárolása
FF4B  98         TYA
FF4C  48         PHA
FF4D  BA         TSX
FF4E  BD 04 01   LDA  0104,X   a break-kapcsoló beolvasása a veremből
FF51  29 10      AND  #10      és vizsgálata
FF53  F0 03      BEQ  FF58     alacsony
FF55  6C 16 03   JMP  (0316)   BREAK rutin
FF58  6C 14 03   JMP  (0314)   interrupt rutin

*****************************  Video-reset
FF5B  20 18 E5   JSR  E518     a videovezérlő inicializálása
FF5E  AD 12 D0   LDA  D012     a raszter-sor
FF61  D0 FB      BNE  FF5E     várakozás a videosor végére
FF63  AD 19 D0   LDA  D019     a raszter-sor megszakítást kér?
FF66  29 01      AND  #01
FF68  8D A6 02   STA  02A6     PAL/NTSC változat tárolása
FF6B  4C DD FD   JMP  FDDD     interrupt timer beállítása

*****************************  interrupt timer beállítása
```



```
FF6E  A9 81      LDA  #81      az A timer aláfutása
FF70  8D 0D DC   STA  DC0D     interrupt control register
FF73  AD 0E DC   LDA  DC0E     az A vezérlőregiszter
FF76  29 80      AND  #80      a 7.bit törlése, óra 60 Hz-s triggerezése
FF78  09 11      ORA  #11      az A timer indítása
FF7A  8D 0E DC   STA  DC0E     az A vezérlőregiszter
FF7D  4C 8E EE   JMP  EE8E     soros ütem ki
FF80  03         ***

*****************************  az op.rendszer rutinjainak ugrási tábl.
FF81  4C 5B FF   JMP  FF5B     video-reset

FF84  4C A3 FD   JMP  FDA3     a CIA-k inicializálása

FF87  4C 50 FD   JMP  FD50     a RAM törlése, ill. vizsgálata

FF8A  4C 15 FD   JMP  FD15     az I/O inicializálása

FF8D  4C 1A FD   JMP  FD1A     az I/O vektorok inicializálása

FF90  4C 18 FE   JMP  FE18     a státusz beállítása

FF93  4C B9 ED   JMP  EDB9     másodlagos cím küldése, LISTEN

FF96  4C C7 ED   JMP  EDC7     másodlagos cím küldése, TALK

FF99  4C 25 FE   JMP  FE25     a RAM végének beállítása/beolvasása

FF9C  4C 34 FE   JMP  FE34     a RAM kezdetének beállítása/beolvasása

FF9F  4C 87 EA   JMP  EA87     a billentyűzet lekérdezése

FFA2  4C 21 FE   JMP  FE21     az IEC-busz time-out kapcs. beállítása

FFA5  4C 13 EE   JMP  EE13     input az IEC-buszról

FFA8  4C DD ED   JMP  EDDD     output az IEC-buszról

FFAB  4C EF ED   JMP  EDEF     UNTALK küldése

FFAE  4C FE ED   JMP  EDFE     UNLISTEN küldése

FFB1  4C 0C ED   JMP  ED0C     LISTEN küldése

FFB4  4C 09 ED   JMP  ED09     TALK küldése

FFB7  4C 07 FE   JMP  FE07     a státusz behozatala

FFBA  4C 00 FE   JMP  FE00     a file-paraméterek beállítása

FFBD  4C F9 FD   JMP  FDF9     a file-név paraméterek beállítása

FFC0  6C 1A 03   JMP  (031A)   $F34A OPEN

FFC3  6C 1C 03   JMP  (031C)   $F291 CLOSE
```

```
FFC6  6C 1E 03    JMP  (031E)
FFC9  6C 20 03    JMP  (0320)
FFCC  6C 22 03    JMP  (0322)
FFCF  6C 24 03    JMP  (0324)
FFD2  6C 26 03    JMP  (0326)
FFD5  4C 9E F4    JMP  F49E
FFD8  4C DD F5    JMP  F5DD
FFDB  4C E4 F6    JMP  F6E4
FFDE  4C DD F6    JMP  F6DD
FFE1  6C 28 03    JMP  (0328)
FFE4  6C 2A 03    JMP  (032A)
FFE7  6C 2C 03    JMP  (032C)
FFEA  4C 9B F6    JMP  F69B
FFED  4C 05 E5    JMP  E505
FFF0  4C 0A E5    JMP  E50A
FFF3  4C 00 E5    JMP  E500
FFF6  52 52 42 59
```

$F20E CHKIN - bemeneti egység beáll.

$F250 CKOUT - kimeneti egység beáll.

$F333 CLRCh - I/O visszaállítás

$F157 BASIN - egy karakter bevitele

$F1CA BSOUT - egy karakter kihozatala

LOAD

SAVE

az idő beállítása

az idő behozatala

$F6ED a stop billentyű lekérdezése

$F13E GET

$F32F CLALL

az idő növelése

SCREEN - a sorok/oszlopok számának beolv.

a kurzor beáll./a kurzorpoz. beolvasása

az I/O modul kezdőcímének beolvasása

```
**********************************
FFFA  43 FE        $FE43
FFFC  E2 FC        $FCE2
FFFE  48 FF        $FF48
```

a hardver-vektorok
az NMI-vektor
a RESET-vektor
az IRQ-vektor

A CBM 64-es kapcsolási rajza

A kapcsolási rajz ismertetése nem nélkülözheti a digitális technika szakkifejezéseit.
A digitális technika alapelemeinek taglalása meghaladja a könyv kereteit. A szerzők kénytelenek voltak feltételezni, hogy az Olvasó tisztában van az olyan alapfogalmakkal, mint a VAGY, illetve ÉS kapu, a hexadecimális aritmetika stb. Ha valaki nem járatos ezekben a témakörökben, feltétlenül javasoljuk, hogy szerezzen be egy alapkönyvet, és egy kicsit tanulmányozza, mielőtt a következő fejezetet elolvassa. A kapcsolási rajzon látható számtalan vezeték, kapu és IC senkit ne riasszon el, a megértéshez nem szükséges, hogy az Olvasó hardveres szakember legyen.
Reméljük, hogy mindenki, aki kellő türelemmel és figyelemmel lát hozzá a fejezet olvasásához, eljut odáig, hogy a számítógépet hardver oldalról is ismerősebbnek látja majd, mint korábban. Természetesen a legapróbb részletek megértéséhez sokkal több időre van szükség, mint amennyit a fejezet egyszeri elolvasása igényel.
Lássunk világosan!
A műszaki beállítottságú számítógéptulajdonosok többsége ég a vágytól, hogy legalább egyszer belenézhessen a gép belsejébe.
Feltesszük, hogy ez alól az Olvasók sem kivételek. A természetes kíváncsiság kielégítése közben azonban nem árt ügyelni arra, hogy nehogy saját magunkban, vagy esetleg a számítógépben valami kárt tegyünk. Legjobb, ha a gépház eltávolítása előtt minden vezetéket kihúzunk a gépből, azaz minden külső egységet lekapcsolunk róla. Ha ez megtörtént, nyugodtan meglazíthatjuk a csavarokat, levehetjük a tetőt és kedvünkre gyönyörködhetünk a látványban. Ami a szemünk elé tárul, az nem egyéb, mint amiről ez a fejezet szól.

Az áramellátás

Bár a számítógépek áramellátása nem igényel különösebb trükköt, a C 64-es fejlesztői igyekeztek olyan megoldást találni, amely minimális ráfordítással maximálisan hatékony.
A hálózati csatlakoztatás eszköze a transzformátor. A transzformátort egy egyenirányítóval együtt beépítették a transzformátorházba, és az egész egy 7 pólusú DIN dugasszal csatlakozik a gép CN7-es aljzatához. A transzformátor 9 voltos váltakozó feszültséget állít elő, amely a CN7-es 6. és 7. lábához érkezik. Az egyenirányító egy másik tekercsen keresztül egy 5 voltos, stabilizált egyenfeszültséget állít elő. Ezt az 5 voltot a CN7-es 5. láb kapja, a földvezeték pedig az 1-es, 2-es és 3-as lábakhoz csatlakozik.
Az aljzat érintkezőitől érkező feszültségeket átvezetik az L5-ös és L4-es tekercsen, és a C20-as, C21-es, C98-as, C99-es és C100-as kondenzátoron, így a hálózati zavarok ugyanis kiszűrhetők. Az SW1 jelölésű kétpólusú kapcsoló a tulajdonképpeni bekapcsoló.
A 9 voltos váltakozófeszültség, amelyet az F1 (1 amperes) biztosíték véd, a user port 11-es és 12-es érintkezőjén áll rendelkezésre. Ezt a feszültséget használhatjuk egyenirányítás és szűrés után a külső egységeken. Ha az áramforrást maximum 100 mA-rel terheljük, a biztosíték kellő védelmet jelent.
Ha a biztosíték kiég, az alapgépen világít a LED, a lemezegység végrehajt egy RESET műveletet, a képernyőn azonban semmi nem látható.
Ilyenkor ellenőrizzük, hogy a tv-t a megfelelő csatornára állítottuk-e, és nem feledkeztünk-e meg a tv kábel csatlakoztatásáról. Ha minden rendben van, ellenőrizzük a biztosítékot. Ha kiégett, cseréljük ki egy 1,25 A-esre. (Bár nem ez az előírt érték, de legalább biztosan állja a terhelést.) Ha ez a biztosíték is kiég, sajnos valószínűleg elromlott az alapgép.
A biztosíték után egy egyenirányítókapcsoló következik, amely stabilizált 5 voltos, váltakozó 9 voltos és stabilizált 12 voltos feszültséget szolgáltat. A kapcsoló a CR4-es hídkapcsolású egyenirányítóból, illetve a CN5-ös és CN6-os diódából áll. A hídkapcsolású egyenirányító utáni 9 voltot a VR2-es – egy integrált 5 voltos feszültségszabályozó – stabilizálja 5 voltra.
A CN5-ös és a CN6-os egyenirányító dióda a váltakozó feszültséget kb. 16 V értékű

szabályozatlan egyenfeszültségre alakítja, amelyet a VR1-es feszültségszabályozó 12 voltra stabilizál.
A transzformátorházból érkező 5 V-os feszültséget a beépített feszültségszabályozó már stabilizálja. Ennek az az előnye, hogy a kialakult hőveszteség nem melegíti a gépet, ami egyébként is elég hőt fejleszt.
Ez a feszültség látja el a gép legtöbb integrált áramkörét és a CN2-es user port 2-es lábára érkezik. Ezáltal a kisebb feladatokhoz is megfelelő feszültség áll a rendelkezésünkre. Ezt a feszültségforrást sem szabad túlterhelni. A maximális áram 100 mA lehet, ami a legtöbb integrált áramkörhöz elegendő.
Az áramkört örvendetes módon rövid időtartamú rövidzárlati szilárdság jellemzi. Az esetleges rövidzárlatot rendkívül egyszerűen megállapíthatjuk, mert ekkor nem jelenik meg kép a tv-n és nem világít a LED, hiszen ezeket is ez a feszültség táplálja.
A CBM 64-esben előállított 5 voltos feszültség jelölése CAN + 5. Ez a feszültség látja el a videovezérlőt (a továbbiakban a rövidített jelölése: VIC), a video kimenőfokozatot és az óra ütem előállításához szükséges összes integrált áramkört. A VIC közvetlenül kapja az 5 voltot, a video kimenőfokozathoz érkező feszültséget pedig az L1-es tekercs, valamint a C61-es, C63-as és a C64-es kondenzátor szűri.
Az ütem előállításához tartozó összes szerkezeti elem az L2-es, C65-ös, C66-os és a C67-es által szűrt feszültséget kapja.
Mivel a kazettás egység nem rendelkezik saját tápegységgel, ehhez is a gép biztosítja a szükséges áramforrást. A kazettásegység meghajtómotorja 6 V-os, a beépített elektronika pedig 9 V feszültséget igényel.
A meghajtómotor a Q1-es, Q2-es, és a Q3-as tranzisztorokon keresztül kapja a feszültséget, amely a CN3-as magnetofonport dugaszoló 3. és C érintkezőjéhez vezet.
Ha a processzor a 5. portbitet magasra állítja, a Q2 tranzisztor átkapcsol. Ezzel rövidrezár a CR2 zenerdióda, a Q1 tranzisztor nem kap bázis-előfeszítést, s a Q1 és a Q3 zár, a meghajtómotor leáll.
Ha viszont a portbit alacsony, úgy a Q2 tranzisztor zár. A Q1 bázisa 7,5 V zenerfeszültséget kap, s a Q1 és a Q3 tranzisztort vezérli. A Q3 tranzisztor emitterére a tranzisztorok két bázis-emitterfeszültségével csökkentett (kb. 1,5 V) zenerfeszültség érkezik, s ez kb 6 V-ot eredményez.
A motorkapcsolási fokozat stabilizálása által állandó motorfordulatszámot érhetünk el.
A kazettás magnetofon elektronikáját a CN3 dugaszoló 2. és B érintkezőjén keresztül látja el a gép.
Most már csak a 12 V feszültség van hátra. Ez a feszültség a VIC-hez, a SID-hez (Sound Interface Device) és a Q8 tranzisztort tartalmazó audió-végfokozathoz szükséges.
Nem közvetlenül az áramellátáshoz tartozik az U27 kapunál levő kis áramkör. Ezzel azonban nem kell külön foglalkoznunk, miután a jeleit a tépegységből kapja.
Az U27 kapu egy logikai ÉS kapcsolás. A 13-as bemeneti láb állandóan 5 V-ra van kapcsolva, a 12-es bemeneti láb pedig az R5 ellenálláson keresztül 9 V váltakozó feszültséget kap. A 12-es lábon tehát az 50 Hz hálózati frekvenciájú feszültség megváltozna.
Így a TTL-bemenet egy 9 V-os feszültséget már nem bír ki, tehát ennél a bemenetnél a – 9 V negatív feszültséget kerülni kell, ellenkező esetben ugyanis az IC használhatatlanná válik.
A bemenőfeszültséget a rákapcsolt CR1 zenerdióda határolja. Amikor a váltakozófeszültség + 2,7 V fölé emelkedik, a zenerdióda visszaszorítja erre az értékre. Ez egy magas logikai jelet eredményez.
A zenerdióda a negatív feszültséget – 0,7 V-ra korlátozza, vagyis olyan értékre, amellyel a TTL-bemenet még jól megbirkózik, s ezt a rendszer logikai alacsony jelként észleli. A feszültség tehát az U27 12-es lábra kapcsolt hálózati frekvencia ritmusában az alacsony és magas szint között ingadozik. Ezzel párhuzamosan a kimenet is ugyanilyen ütemben változik. A R37 ellenállás egy pozitív visszacsatolás feladatát látja el, amely felgyorsítja a felfutási és a lefutási időt, hogy a további folyamatokhoz tiszta négyszög-impulzusokat kaphassunk.

Mi az, amit még tudnunk kell?

A kapcsolási rajzon látjuk, hogy az 50 Hz-es értékek az U1 és U2 integrált áramkörhöz, vagyis a két CIA-hoz érkeznek. A kapcsolási rajz további leírásában még részletes tudnivalókat olvashatunk a CIA-ról. Egyelőre azonban csak annyit, hogy
– a hálózati frekvencia a legegyszerűbben előállítható frekvenciaállandó-jel. Éppen ezért főként olyan esetekben hasznos, amikor időt kell mérni. Nos a jel is ezt a feladatot látja el a CIA-kban. A valós idejű órák (timerek) az ütemüket a hálózati frekvenciáról kapják.

Az ütem előállítás

A számítógép szabályos működése szempontjából rendkívül fontos a stabil és zavarmentes áramellátás, de ugyanilyen meghatározó az ütemjelek állandósága és stabilitása is. Így érdemes több figyelmet szentelni arra, hogy az ütem előállítási módját megismerjük.
A CBM 64-es nyomtatott áramköri kártyáját megnézve és a kapcsolási rajz alapján végigjárva az integrált áramköröket, előfordulhat, hogy az egyik vagy másik IC-t nem tudjuk az első pillantásra felfedezni. Az elrejtett IC-k közé tartoznak például az órajel előállításáért felelőek is. Ezek a VIC-kel együtt egy fémdobozban helyezkednek el a nyomtatott áramköri lap közepén (nem a tv csatlakozást tartalmazó doboz, mert ez az UHF-modulátor).
Ez a fémház (fémtok) árnyékolja az órajel előállításánál keletkező nagyfrekvenciás zavaró sugárzást.
A megfelelő árnyékolás nélküli gépeknél megfigyelhető, hogy a közelben levő rádiókészülékek hangszórójából csak sípoló vagy sistergő hang hallatszik. Még rosszabb a helyzet az ilyen zavaró sugárzások által befolyásolt tv készülékeknél. Ha a Commodore 64 nem biztosítana megfelelő zavarszűrést, a géphez csak monitort (képernyős adatmegjelenítő) használhatnánk.
Az összes meghatározó ütemfrekvenciát az Y1-es kvarc állítja elő. Ennek megértéséhez azonban elengedhetetlen némi előzetes magyarázat. A következő adatok egy az NSZK-ban forgalmazott, PAL kimenetű készülékre vonatkoznak.
Az Y1 kvarc 17.734472 MHz frekvenciával rezeg, s a C70-en keresztül csatlakozik az U31 IC-hez. Az U31 IC egy 74LS629 jelölésű TTL-IC, s 2 egymástól független VCO-t tartalmaz. Egy VCO nem más, mint egy feszültség által vezérelt oszcillátor. A vezérlőbemenetre kapcsolt egyenfeszültség a frekvenciát egy meghatározott tartományban megváltoztathatja. Az 1 VCO-nak ez a vezérlőbemenete az 1-es láb. Az említett bemeneten levő R27 potenciométer segítségével is módosítható a kimenő frekvencia. Miután azonban a kvarcokra is jellemző egy bizonyos tűrés, a potenciométer segítségével a névleges frekvenciaérték is beállítható.
Az 1 VCO kimenete a 10-es láb. Az erre kapcsolt frekvencia 0COLOR jelként közvetlenül a VIC-hez érkezik.
A jelet egyidejűleg az U30 IC is megkapja. Ez frekvenciaosztóként kapcsolt 74LS193 jelölésű integrált áramkör. Az osztón beállítható az osztási viszony. Az 1-es, 9-es, 10-es és 15-ös lábakon kialakuló szinttől függően minden egyes osztási viszonyt 0:1 és 15:1 között lehet szabályozni. Esetünkben az osztási viszony 9:1-re van beállítva, s így a 17.734 MHz 9-cel kerül leosztásra. Így a 6-os lábon 1.9704 MHz frekvenciát kapunk.
Az U29-ben két flip-flop található. A 11-es lábra érkező órajel minden egyes pozitív éle az 1-es flip-flop 12-es adatbemeneti lábára érkező információt a 9-es láb –Q kimenetére továbbítja. A –Q kimenet (8-as láb) is megkapja a bemeneti információt, de ellenkező polaritással.
Az ismertetett kapcsolásban a 9-cel osztott kvarcfrekvencia szolgáltatja az FF1 számára az órajelet. Az adatbemenet össze van kapcsolva a – Q kimenettel.
Ha a – Q kimenet magas, a 11-es lábra érkező, következő pozitív homlokú magas jel a Q kimenetre érkezik.
Ezzel egyidejűleg a – Q kimenet alacsony lesz. A következő pozitív ütemű él az alacsony jelet Q-ra kapcsolja, amikor is a – Q ismét magas lesz és így tovább.
Ezek a folyamatok jobban érthetőek, ha megvizsgáljuk a közölt ábrát, amely a frekvenciákat és a fázisállapotokat szemlélteti. (295. old)

Az elmondottak szerint minden második ütemimpulzussal megváltozik a kimenetek állapota. Ez 2-vel való frekvenciaosztást jelent, vagyis a kimeneten 985,248 kHz jelenik meg. Ez a processzor ütemfrekvenciája.

A jelet azonban nem közvetlenül ütemként használja a rendszer, a dolog egy kicsit összetettebb. A 7,88198 MHz frekvenciájú Dot Clock frekvenciaosztással nem vezethető le a kvarcfrekvenciából. A megoldást egy PLL kapcsolással megvalósítandó frekvenciaszintézis adja.

A PLL rövidítés: Phase Locked Loop (fázisban szabályozott ciklus).

A Commodore 64-esben ezt a feladatot az U32, U31 és a VIC integrált áramkörökön felépülő PLL látja el.

A PLL legfontosabb alkotórésze egy kétbemenetű fáziskomparátor. A fáziskomparátor kimenő egyenfeszültsége arányos a képjel fázisállapotával. Ezt a funkciót az U32 IC és a Q7 transzformátor biztosítja.

A részletes működési elv a következő:

Az U32 integrált áramkör 1-es láb bemenetére az U29 flip-flop kimenetéről érkezik a 985 kHz frekvencia. A PLL másik, 3-as lába az 0o jelet kapja, amely nem más, mint a VIC által a processzor számára továbbított órajel, de még meghatározatlan frekvenciával.

A VIC-nek ez az 0o jele az U31-ben a 2 VCQ 8-cal osztott kimenőjele. Az említett 8-as frekvenciaosztás közvetlenül a VIC-ben történik.

A 2 VCO frekvenciáját nem egy kvarc, hanem a C86 kondenzátor határozza meg. A 2 VCQ vezérlőfeszültségét az U32 fáziskomparátor kimenete szolgáltatja.

Amikor a 2 VCQ vezérlőfeszültsége 3 V értékű, akkor ez 7,88198 MHz frekvenciát jelent.

Ha olyan esetet veszünk alapul, amikor az U29 flip-flop frekvenciája nagyobb az Oo-nál, vagyis amikor a 2 VCO-n például csak 7.7 MHz van, úgy a fáziskomparátor 8-as láb-kimenete egy 3 V-nál kisebb feszültséget ad, amely a VCO nagyobb frekvenciájú rezgését teszi lehetővé. Ezzel a fáziskomparátor 3-as lábának frekvenciája is növekszik, megközelíti az 1-es láb referencia-frekvenciáját, s a vezérlőfeszültség egyre inkább a 3 V érték felé tart. Amikor az 1-es és a 3-as láb frekvenciája azonos értékű, a VCO továbbra is szabályozott állapotban marad egészen addig, amíg a jeleknek nemcsak a frekvenciája, hanem a fázisa is azonossá válik.

Ugyanez a folyamat játszódik le akkor is, ha a VCO-n magasabb frekvencia van. Igy a vezérlőfeszültség 3 V-nál nagyobb lett. Ha a VCO-n alacsonyabb frekvencia van, a vezerlőfeszültség leesik mindaddig, amíg a jelek frekvencia- és fáziscsatoltak lesznek. A Dot Clock jel rákapcsolására csak ezután kerül sor. A fentiekben vázolt szabályozási folyamatok rövid időt vesznek igénybe. Legkésőbb 100 ms. elteltével az összes frekvencia rendelkezésre áll.

Végezetül vizsgáljuk meg még röviden a FF2 feladatát és az NTSC színkimenetű 64-es készülékben végbemenő folyamatokat.

Az Amerikában forgalmazott készülékeknél a FF2-be egy 14.31818 MHz-es kvarc van beépítve. A NTSC változatnál a gép egy másik, 6567 jelű VIC chipet tartalmaz. A PAL változatnál ez a 6569-es. A harmadik különbség az, hogy az E1 és az E2, vagy az E3 pont közötti huzaláthidalások másként helyezkednek el. A PAL készülékeknél az említett áthidalás az E1-et és az E2-t kapcsolja össze. Igy az U30 1-es és 10-es lába + 1 V-ot kap. Az U29 integrált áramkör 4 lába is magasfeszültség-szintű. Ez a 4-es láb a FF2 ún. preset-bemenete. Az ilyen FF-ek clock-, adat- és clear-bemenete a földre van kapcsolva. A clear-bemeneten megjelenő alacsonyszintű jel a flip-flop-ot határozott állapotba billenti át. Függetlenül a többi bemenőjeltől, a Q bemenet alacsony, a — Q bemenet pedig magas lesz.

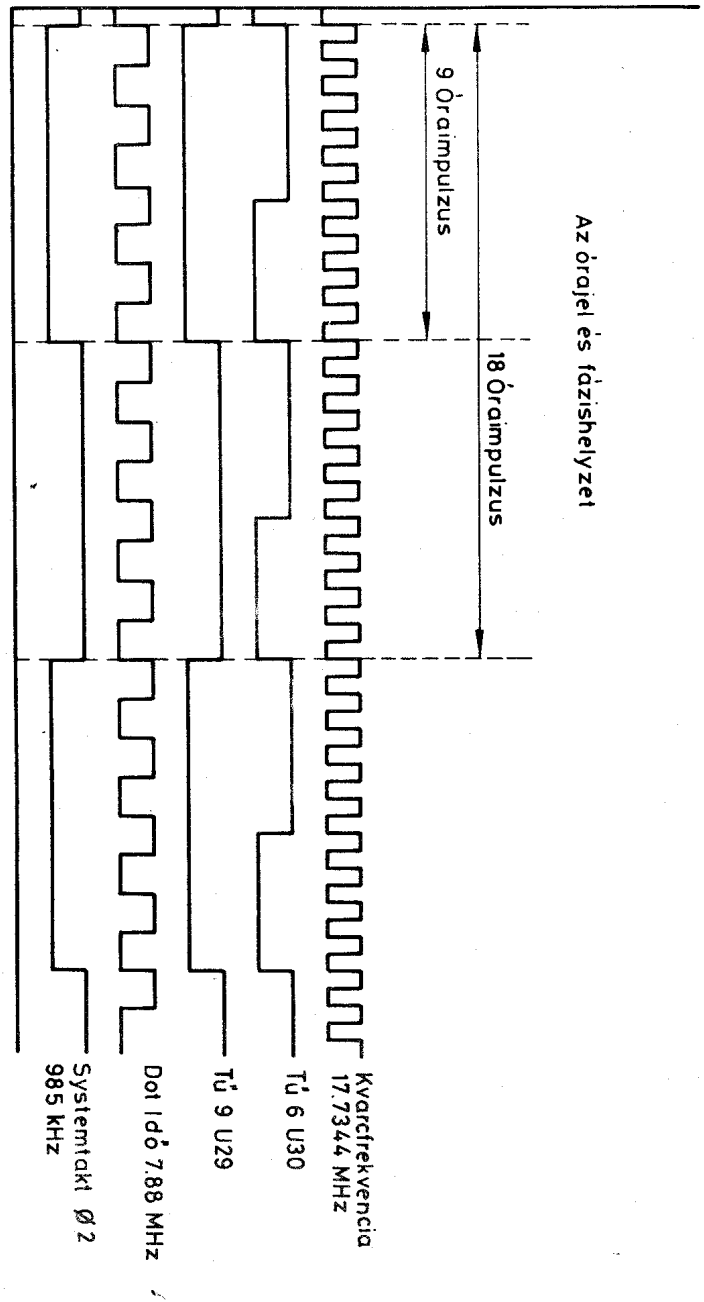
Itt is van azonban egy fontos megkötés. A preset-bemenet magasszintű kell, hogy legyen.

A PAL készülékeknél ezt a feltételt a huzaláthidalások teljesítik.

Az U30 számláló 1-es, 9-es, 10-es és 15-ös bemenete binárisan kódolt formában határozza meg a számláló kezdőértékét. Miután a számláló mindig 16-ig számol, a kezdőértékkel beállítható az osztási viszony. A számláló nem 0-val indul, hanem a programozott értékkel.

Az "A" bemenet az alacsonyabb értékű bit, a "D" bemenet pedig a legnagyobb értékű bit. Ezeken a bemeneteken egy decimális 7-es áll. A számláló 16-ig továbbszámlál, majd ismét 7-tel kezd. Ehhez 9 számlálóimpulzus szükséges, vagyis 9-cel oszt.

A FF2 tehát nem más, mint egy egyszerű inverter. Ha a bemenet magas, a kimenet alacsony és megfordítva.



A NTSC-nél az U29-es preset-bemenete alacsony. Az adatlap szerint most a Q és a – Q kimeneten egyaránt magas a jelszint, ami tulajdonképpen egy szokatlan állapot, de mégsem okoz kárt az integrált áramkörben.
Az U30-as osztási viszonya most 7:1, a következő flip-flop-é 14:1, s így a processzor ütemfrekvenciája 1.0227 MHz, valamivel nagyobb, mint a PAL-munkafrekvencia.

A processzor

Mint már az előzőekből tudjuk, a CBM 64-es számítógépbe egy 6510-es processzort építettek be. Ez az új processzor az ismert 6502-estől főként egy, a processzorchipben integrált portban tér el. Ez a port 6 programozható IO-vezetékkel rendelkezik (IO = a tetszés szerint kimenetként vagy bemenetként kapcsolható input-output vezeték rövidítése). A 8-bites processzorok kapcsán a 6-os szám minden bizonnyal kissé szokatlannak tűnik. A rendelkezésre álló 40 pólusú házaknál azonban már nem volt több szabad vezeték, s így nem volt lehetőség a teljes 8-bites port beépítésére.

A 6510 processzor lábkiosztása:

Láb	jelölés	feladat
1	0IN	bemenet, a VIC 17-es lábról érkező rendszerütem
2	RDY	bemenet, az U27 3-as lábról érkező ReaDY
3	–IRQ	bemenet, Interrupt Request
4	–NMI	bemenet, Non Maskable Interrupt
5	AEC	bemenet, Adress Enable Control
6	VCC	+ 5 V üzemi feszültség
7–20	A0–A13	kimenet 0–13 címbit
21	GND	föld-üzemi feszültség
22	A14	kimenet, 14. címbit
23	A15	kimenet, 15. címbit
24–29	PB5–PB0	BE-kimenet, 5–0 portbit
30–37	D7–D0	BE-kimenet, 7–0 adatbit
38	R/–W	kimenet, Read/-Write
39	02	kétfázisú ütemkimenet (Phase Two), a továbbiakban 02
40	–RES	bemenet, RESet

Sok más processzorhoz hasonlóan tehát a 6510-es is egy 8-bites adat- és egy 16-bites címbusszal rendelkezik. Így 64 k tárterület közvetlen címzésére alkalmas.
A 0IN és a 02 jel lényegében a rendszer órajele, úgy is mondhatjuk, hogy a gép pulzusa. A 0IN jelet a VIC állítja elő, s a frekvenciája hozzávetőlegesen 985 kHz. Ebből a jelből állítja elő a processzor a 02 jelet. A processzor és a külső egységek együttműködésében a 02 rendkívül fontos szerepet tölt be, mert ez képviseli a processzor összes műveleteinek a referencia-ütemét.
A –RES jel feladata az, hogy a processzort és más integrált áramköröket egy meghatározott kezdeti állapotba hozza. Ez a reset, a bekapcsolás pillanatában alakul ki.
Nézzük meg közelebbről a bekapcsolási pillanatot. A –RES jelet az U20 integrált áramkör állítja elő. Ez a valóságban egy NE556 integrált áramkör, amely két azonos óra-fokozatot tartalmaz.
Ezekkel az órákkal egyszerű külső kapcsolással oszcillátor- vagy impulzusadó-fokozatot alakíthatunk ki. A mi esetünkben az IC impulzusadó feladatát látja el.
Az üzemi feszültség rákapcsolásával a C105 kondenzátor a R50 ellenálláson keresztül feltöltődik. Egyidejűleg a C24 kondenzátor is feltöltődik az R34 ellenálláson keresztül. Ha most

egy bizonyos idő elteltével (néhány 10 ms) a C105 feszültsége 1,6 V fölé emelkedik (az üzemi feszültség 1/3-a), indul a tulajdonképpeni impulzus. A C24 kondenzátor a 13 csatlakozáson keresztül lökésszerűen kisül. Ezzel egyidejűleg a 9-es láb – az óra kimenete – 5 V-ra kapcsol. Ezután a C24 az R34 ellenálláson keresztül ismét feltöltődik. Most azonban már a feszültséget a 12-es láb bemenet ellenőrzi. Abban a pillanatban, ahogy a feszültség túllépi az üzemi feszültség 1/3-át (3,3 V), a kimenet ismét alacsonyszintű lesz. Ennek eléréséhez kb. 5 másodperc szükséges. Az óra 9-es lábkimenetén levő inverter ezt a pozitív impulzust negatívvá alakítja át. A kimenetén a tulajdonképpeni –RES jel áll rendelkezésre.
A magas/alacsony átváltás pillanatában kezdi a processzor a munkáját. Mindenekelőtt behozza a $FFFC és $FFFD címről (Reset vektor néven ismeretes) a következő feldolgozandó utasítás címét. Az operációs rendszer ezen a címen kezdődik.
A R/-W jelölésű láb jelzi, hogy a processzor ír vagy olvas. Ha ez a vonal magas, a processzor adatokat olvas a RAM-ból, a ROM-ból vagy az interface-chipből. Ha ugyanez a vonal alacsony, a processzor ír, vagyis adatokat tárol a mindenkor megcímzett területen. Az írás természetesen csak akkor értelmes, ha a megcímzett modul az adatok tárolására is képes. A RAM-ba irányuló írásnak nincs értelme, mivel a ROM adatait már a gyártáskor meghatározták, s ezek nem módosíthatók.
A –NMI jelölésű láb (Non Maskable Interrupt) egy éppen futó program megszakítását teszi lehetővé
A „nem maszkolható" azt jelenti, hogy a megszakítás mindig megengedett, nem lehet szoftveresen letiltani.
Ezt a csatlakozást a földre kapcsolva, az éppen lefutott gépi kódú utasítás befejeztével a rendszer kilép a futó programból. A processzor az NMI-vektorból ($FFFA és $FFFB) behozza az interrupt-rutin címét és ennek alapján végrehajt egy elágazást. A CBM 64-esben az NMI-t három különböző esemény indíthatja.
A RESTORE billentyű lenyomásával az U20 második órája egy megfelelő impulzust állít elő. A billentyű lenyomása ütésszerűen kisüti a C38 kondenzátort. Az R35 ellenálláson keresztül a kondenzátor akkor is újra feltöltődik, ha a RESTORE billentyű még lenyomott állapotban van. A tulajdonképpeni NMI-impulzus akkor indul, amikor az U20 6-os lábán a feszültség 1,6 V fölé emelkedik. Az óra 5-ös lábkimenete magas lesz, az U6 inverter kimenetén egy alacsony szintű jel jelenik meg, a C23 kondenzátor kisül az U20 1-es lábán keresztül, majd az R33-on keresztül ismét töltődni kezd.
Mintegy 18 µs elteltével a C23 feltöltődik az üzemi feszültség 1/3-ára, az 5-ös lábkimenet ismét alacsony, s a processzor –NMI-bemenete pedig ismét magas lesz. A második esetet az U2 –CIA hozza létre. Bizonyos események bekövetkezésekor az említett integrált áramkör 21-es lábán alacsony jelszint alakulhat ki. Ennek a –NMI-nek az előállításával a CIA-kat tárgyaló fejezet foglalkozik.
A harmadik eset a Cartridge Expansion "D" csatlakozásának a rövidrezárása. Itt külső modulok indíthatnak megszakítást. Hasonló a –NMI-hez a –IRQ (Interrupt Request). A –NMI-hez viszonyítva itt lényeges eltérésként a –IRQ megszakítási vektora tekintendő. Ez a vektor a $FFFE és a $FFFF címen van. Ez a megszakítás szoftveresen letiltható.
Ha a processzor állapotregiszterében az I-kapcsoló (2 bit) magas, az összes fellépő megszakítás figyelmen kívül marad. A –NMI-hez viszonyítva még az a tény is eltérést jelent, hogy a –IRQ nem élvezérlésű. A megszakításnak tehát legalább addig kell tartania, amíg a processzor ezt a csatlakozást ellenőrzi.
A –IRQ előállítása szintén három különböző módon történhet. Meghatározott programozható állapotok elérésekor az U2 CIA-hoz hasonlóan, az U1 CIA is egy alacsony jelszintet hoz létre a 21-es lábon, ami a processzornál egy –IRQ-t idéz elő.
Az interrupt előállításának egy másik lehetősége a VIC. Előzetesen programozással rögzített, meghatározott események bekövetkezésekor a CIA-khoz hasonlóan, a VIC 8-as lábon is alacsony szint alakul ki, s ez –IRQ-hoz vezet. A –IRQ előállításának harmadik lehetősége a „Cartridge Expansion" csatlakozó (CN6) 4-es kapcsának rövidrezárása. Ezáltal külső kapcsolásokkal is generálható a –IRQ. A RDY láb jelzi a processzornak, hogy az adatbuszra kihelyezett

információk érvényesek vagy érvénytelenek. Ha ez a láb alacsony szintű, a processzor tudja, hogy még nem veheti át az adatokat. A processzor ekkor ún. várakozási állapotba kerül és felfüggeszti tevékenységét. Az egyes ütemimpulzusokkal csak azt ellenőrzi, hogy az RDY lábon mikor alakul ki újra magas szint.
A régebbi rendszereknél ezt a lehetőséget arra használták, hogy lassabb működésű tár- és perifériamodulokat csatlakoztassanak a processzorhoz. A CBM 64-es gépben ezt a lehetőséget a VIC jel használja.
A VIC a RAM memóriát normális körülmények között csak a processzor által ki nem használt ütemközi szünetekben éri el (02 = alacsony). A VIC bizonyos műveleteinél, például a sprite-ok ábrázolásakor, a VIC-nek több időre van szüksége, mint amely az ütemközi szünetekben rendelkezésre áll. Ekkor a BA csatlakozásnál (Bus Avaialble) a VIC alacsony jelszintet állít elő, amely az U27 ÉS-kapun keresztül a processzor RDY-bemenetére érkezik, amikor is a processzor a buszt a szükséges időtartamra a VIC rendelkezésére bocsátja.
Az AEC szintén a VIC által az alapkonfigurációban előállított jelként tekinthető.
Amikor a VIC lefoglalja a buszt, az említett csatlakozás 0 értékű. Ezt a jelet a processzor AEC-lába kapja, s a hatásaként a processzor a busz vonalait átállítja nagyellenállású, ún. Tri-State állapotba A gyakorlatban mindez úgy néz ki, mintha a processzor nem is a saját IC-tokjában lenne. Amíg az AEC alacsony szintű, az említett állapot megmarad, s más integrált áramkörök, például egy külső processzor vagy egy VIC is lefoglalhatja a rendszerbuszt. A processzorban integrált port a 24-től 29-ig terjedő lábakat foglalja le. A CBM 64-es gépben ez a port különböző feladatokat lát el. Ezeknek a feladatoknak a részletes leírását az alábbiakban olvashatjuk:
A 0-ás portbit jelölése: —LOWRAM. A $A000-tól $BFFF-ig terjedő címtartományban ez a bit végzi a RAM/ROM átkapcsolást, vagyis alacsony szint esetén az említett címtartományra a RAM van bekapcsolva.
A —HIRAM jelölésű 1-es portbit ugyanezt a feladatot látja el a $E000-tól a $FFFF-ig terjedő címtartományban.
A —CHAREN jelölésű 2-es portbit a karakter ROM-ot választja ki, ha alacsony a szintje.
A karakter ROM és az ún. IO-terület ugyanezt a SD000-tól SDFFF-ig terjedő címtartományt foglalja le. A —CHAREN dönti el, hogy a karakter-ROM, vagy ugyanezt a címtartományt használó IQ- vagy periféria-modulok (VIC, SID vagy CIA) kiválasztására van-e szükség.
A fennmaradó három bit a kazettás egység számára van fenntartva.
A kazettás egység felé irányuló adatokat a 3-as portbit szolgáltatja. Ez a láb közvetlen összeköttetésben van a kazetta-port E és 5 csatlakozásával.
A 4-es portbit (Cass Sense) ellenőrzi, hogy a kazettás egységen le van-e nyomva a Play billentyű. Ez a bit közvetlenül a kazetta-port F és 6 csatlakozására érkezik.
A kazettás egység meghajtómotorját az 5-ös bit vezérli. A motorvezérlési funkcióról már az áramellátással foglalkozó fejezetben volt szó.

Címdekódolás

A 6510-es processzor egyszerre egy 64 k-os területet tud megcímezni, ezt azonban a 64 k RAM lefoglalja. A fennmaradó tárterületek szervezését logikai úton kellett megoldani. A bővített társzervezést egy speciális integrált áramkör az ún. címkezelő (Adress-Manager) végzi el. A címkezelő tulajdonképpen egy FPLA (Field Programmable Logic Array) egység, amit a kapcsolási rajzon U17-tel jelöltünk. Ezt az integrált áramkört csak megfelelő programozással lehet felkészíteni a speciális logikai feladatok elvégzésére. Programozhatóságával önmagában képes helyettesíteni sok olyan logikai kaput, amelyekre a hagyományos megoldásban szükség lenne.

A 28-pólusú integrált áramkör lábkiosztása

Láb	Jelölés	Feladat
1	FE	használaton kívül
2	I7	bemenet, A13 a 6510 20-as lábról
3	I6	bemenet, A14 a 6510 22-es lábról
4	I5	bemenet, A15 a 6510 23-as lábról
5	I4	bemenet, —VA14 a CIA 2 portról, a 2-es láb 0. bitje
6	I3	bemenet, —CHAREN a 6510-es portról, 27-es láb 2. bitje
7	I2	bemenet, —HIRAM a 6510-es portról, 28-as láb 1. bitje
8	I1	bemenet, —LOWRAM a 6510-es portról, 29-es láb 0. bitje
9	I0	bemenet, —CAS a VIC 19-es lábról
10	F7	kimenet, —ROMH a bővítő nyílás B lábára
11	F6	kimenet, —ROML a bővítő nyílás 11-es lábára
12	F5	kimenet, —I/O az U15 dekódoló 1-es lábára
13	F4	kimenet, GR/-W az U6 szín-RAM 10-es lábára
14	GND	föld, üzemi feszültség
15	F3	kimenet, —CHARON az U5 karakter ROM 20-as lábára
16	F2	kimenet, —KERNAL az U4 KERNAL ROM 20-as lábára
17	F1	kimenet, —BASIC az U3 BASIC-ROM 20-as lábára
18	F0	kimenet, —CASRAM a RAM 15-ös lábára
19	—OE	bemenet, Output engedélyezés a testre
20	I15	bemenet, —VA12 a VIC 28-as lábról
21	I14	bemenet, —VA13 a VIC 29-es lábról
22	I13	bemenet, —GAME a bővítő nyílás 8-as lábról
23	I12	bemenet, —EXROM a bővítő nyílás 9-es lábról
24	I11	bemenet, R/-W a 6510-es 38-as lábáról
25	I10	bemenet, —AEC a VIC 16-os lábáról
26	I9	bemenet, BA a VIC 12-es lábáról
27	I8	bemenet, A12 a 6510-es 19-es lábáról
28	V_{cc}	+ 5 V üzemi feszültség

Hogyan befolyásolják ,a különböző bemenőjelek az AM kimeneteit? 16 bemenővezetéken 65536 különböző bemeneti kombináció lehetséges. Miután az AM csak 8 kimenettel rendelkezik, világos, hogy több bemeneti kombináció azonos kimenetet eredményez. A 256 lehetséges kimeneti kombináció közül csak kevés olyan van, amely a gép szempontjából fontos. A célszerű kombinációkat a jobb áttekinthetőség végett a 12. oldalon közölt táblázatban foglaltuk össze, és a következő oldalakon a lehetséges tárkiosztásokat is ábrázoltuk.
A szerzők tájékoztatásul közlik, hogy még akkor is, ha a lehetséges bemeneti és a hozzájuk tartozó kimeneti kombinációk csak egy sort foglalnának le, a teljes jegyzék közléséhez 1093 oldalra lenne szükség.

A 6569-es videovezérlő

Minden számítógép két legfontosabb perifériája a beviteli és a kiviteli egység, hiszen ezek teremtik meg a kapcsolatot a gép és az ember között. A CBM 64 kimeneti egysége alaphelyzetben egy tv készülék, vagy egy monitor.
A CBM 64-esben a VIC szolgáltatja a tv monitor üzemeléséhez szükséges jeleket. Ezek a jelek a szinkronizáló- és a fényerőimpulzusok, illetve a színek.
A VIC ezen kívül más feladatokat is ellát. Előállítja például a processzor által igényelt órajelet, vezérlőjeleket biztosít a dinamikus RAM-ok üzemeléséhez stb.

A 40 pólusú tok lábkiosztása a következő:

Láb	Jelölés	Feladat
1–7	D6–D0	processzor adatbusz
8	–IRQ	kimenet, megszakítás kérelem
9	–LP	bemenet, fényceruza (light pen)
10	–CS	bemenet, választás (Chip Select)
11	R/–W	Read/-WRITE
12	BA	Bus Available
13	VDD	+ 12 V üzemi feszültség
14	COLOR	kimenet, színinformáció
15	SYNC	kimenet, sor- és képszinkronizáló impulzus
16	AEC	kimenet, Address Enable Control
17	0OUT	kimenet, rendszerütem
18	–RAS	kimenet, Row Address Select
19	–CAS	kimenet, Column Address Select
20	GND	föld üzemi feszültség
21	0COLOR	bemenet, színfrekvencia
22	0IN	bemenet, Dot-frekvencia
23	A11	processzor címbusz
24–29	A0/A8–A5/A13	multiplexelt RAM címbusz
30	A6	(Video) RAM busz
31	A7	Video) RAM címbusz
32–34	A8–A10	processzor címbusz
35–38	D11–D8	szín-RAM adatbusz
39	D7	processzor adatbusz
40	VCC	+ 5 V üzemi feszültség

A lábkiosztást megvizsgálva látjuk, hogy néhány jelöléssel már találkoztunk. Például a BA, AEC, 02 és az R/-W lábak szerepét már a processzor tárgyalásánál láttuk. Nem esett még szó azonban a –CS, –RAS és a –CAS jel, valamint a D8-D11 adatvezetékek feladatáról. A multiplexelt címbusz is ujdonság, hiszen a processzornál minden címjel külön-külön áll rendelkezésre az egyes lábakon.
A gépben lezajló folyamatok időbeli sorrendjét az órajel (Dot Clock) szabja meg. A CBM 64-esben az órajel frekvenciája kb. 7,85 MHz. A VIC-be beépített fokozat ezt a frekvenciát 8-cal osztja. Az eredmény kb. 980 kHz frekvencia, amely a 17-es lábon, 0OUT rendszerütemként jelentkezik.
Az órajelből alakulnak ki a tv-n megjelenő kép szinkronizálásához szükséges jelek is. Az órajel határozza meg azt az időt, amely alatt a karakter egyes pontjai a képernyőn láthatóvá válnak.
A CBM 64-esben a 0COLOR jel frekvenciája 17.734472 MHz. Ez az Y1 kvarc rezgési frekvenciája, ami a színekkel kapcsolatos információk előállításához szükséges.
Ezek a frekvenciák arra az esetre érvényesek, amikor a gép egy PAL-rendszerű tv-készülékkel üzemel.
Valahányszor a processzor el akarja érni a VIC regisztereit, szüksége van a VIC címzésére. Ennek a legfontosabb feltétele az, hogy a –CS jelölésű vonal alacsony szintű legyen. A processzor csak ezután veheti igénybe a címbuszon található címen keresztül a kívánt regisztert. Nem árt tudni, hogy hogyan alakulhat ki a –CS vonal alacsony szintje. Miután a VIC az ún. IO területen ($D000-tól $DFFF-ig) a $D000-tól $D3FF-ig terjedő címeket foglalja le, az említett címterület elérésekor az AM alacsony szintű jelet hoz létre, a 12-es lábon (–I/O jel). Ezt az alacsony jelet átveszi az 1-es láb, vagyis az U15 dekódoló. A dekódoló felszabadul és a 2-es és a 3-as lábakra kapcsolt A10 és A11 címvezetékektől függően a megfelelő dekódoló-kimenet alacsony szintű lesz. A VIC báziscímét és végcímét binárisan ábrázolva, az alábbi bitmintát kapjuk:

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	= $D000
1	1	0	1	0	0	1	1	1	1	1	1	1	1	1	1	= $D3FF

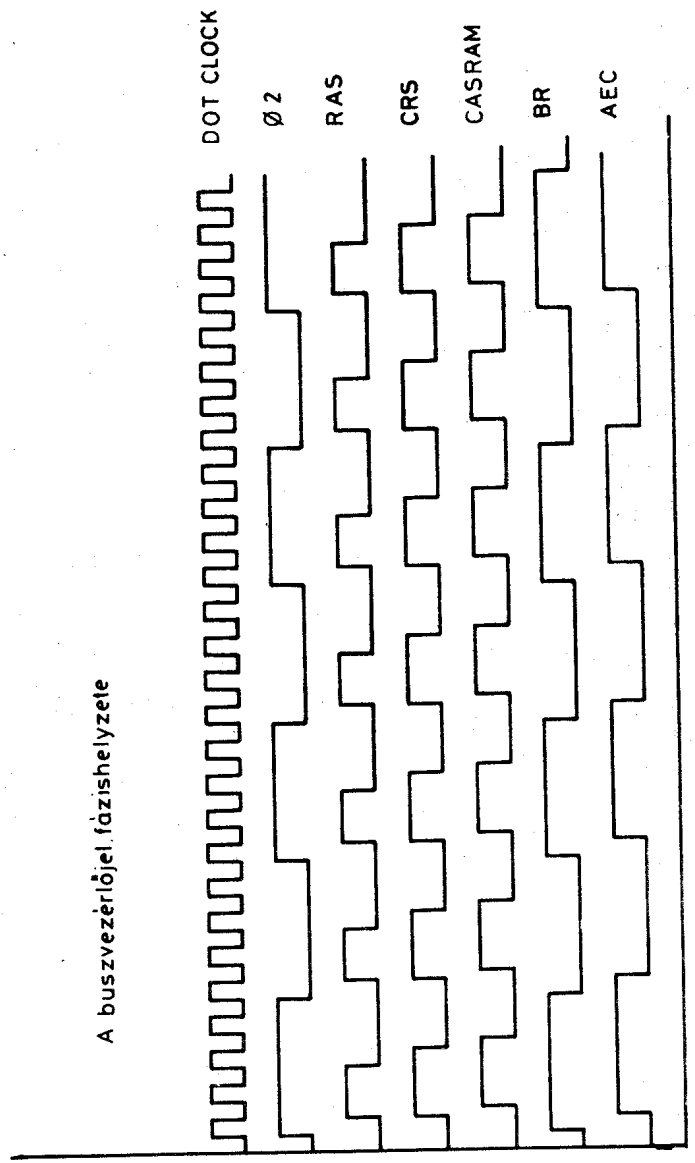
Látható, hogy ezen a címterületen az A10 és az A11 címbit alacsony szinten marad, és így sor kerül a dekódoló alacsony jelű Y0 kimenetének, vagyis a VIC címzésére.
Csak a következő $D400 címnél vált át az A10 magas szintre.
Az Y0 magas lesz, a dekódoló Y1 kimenete alacsonyra vált és így a SID-et címezheti a processzor.
A VIC csak 16 k-s címterülettel tud dolgozni, mert csak az A0-tól A13-ig terjedő címbitekkel rendelkezik. A processzortól eltérően, itt a címvezetékek nem az egyes lábakra vannak rákapcsolva, hanem multiplexeltek. A 24-es láb tehát nemcsak a 0. hanem egyben a 8. címbit is.
Hogyan lehetett ezt megoldani?
A válasz rendkívül egyszerű. A csatlakozás több címbit szerepét tölti be. Csak ún. kisegítőjelekkel lehet eldönteni, hogy egy adott időpontban milyen feladatot lát el a csatlakozás, miként értelmezhető. A segédjelek –CAS és –RAS néven ismeretesek, s többek között a dinamikus RAM-modulok vezérléséhez is szükségesek, miután az utóbbiak is egy multiplexelt címbusszal rendelkeznek.
A tárelérés időbeni lefutása az alábbiak szerint írható le:
A –CAS és a –RAS jel magas szintű. A rendszer először az alacsonyabb értékű címbyte-ot helyezi ki a buszra. Rövid idő elteltével a –RAS jel alacsony lesz. Ezáltal a címbyte-ot átveszik a RAM-ok és tárolják. Most megváltozik a busz-információ.
Az A0-ból A8, az A1-ből A9 lesz, és így tovább. Rövid idő elteltével a –CAS jel is átvált alacsony szintre. A lefutó él az AM-be érkezik és a –CASRAM kimeneten egy, az időben valamelyest késleltetett lefutó homlokot állít elő, így a RAM-ok átvehetik a felső byte-ot is.
Miután a teljes cím rendelkezésre áll, az adatok megjelennek az adatbuszon. A vázolt folyamatokat a közölt idődiagram szemlélteti (302. old.).

A RAM és a VIC közötti illesztő (interface)

Mivel a VIC csak az A0-tól A13-ig terjedő címbiteket állítja elő, de a teljes 64 k RAM címzéséhez további két bitet kell igénybevenni.
Erre szolgál a CIA 2 A portja.. A 0-ás és az 1-es portbit szolgáltatja a 14-es és 15-ös címbitet. Ahhoz, hogy ezek a jelek is részt vehessenek a multiplex-folyamatban, az U14 integrált áramkörön haladnak át.
Az U14 4 db 2-ről 1-re invertáló multiplexert tartalmaz. A multiplexer egy egyszerű váltókapcsolónak tekinthető. A két bemenet egyike tetszés szerint rákapcsolható a hozzátartozó kimenetre.

A folyamat részletes leírása a következő:

A multiplexereket az S bemeneti jel csatlakoztatja. Ha az S bemenet alacsony, az A jelölésű bemenetek kimenetre kapcsolnak át, az S bemenet pedig magas lesz, majd ezt követi a B bemenetek átkapcsolása.
Az A6 és az A7 címbit a VIC-ből a multiplexerhez érkezik, pontosabban az A6 a 13-as és 14-es, az A7 pedig a 10-es és 11-es bemenetre. Ha most az S jellel ide-oda kapcsolunk a bemenetek között, a kimeneteken nem lesz változás, miután a címbitek mindkét bemenetet elérik. A multiplexer invertáló hatása folytán csak a jelek polaritása cserélődik fel a kimeneteken. Ezek az invertált címjelek a két másik multiplexer B bemeneteire érkeznek, pontosabban a –A6 a 3-as, az –A7 pedig a 6-os lábra. Az A bemenetek a CIA 2 említett portbitjeit kapják, ahol a 0. portbit –VA14-ként a 2-es lábra, az 1-es portbit pedig a –VA15-ként az 5-ös lábra érkezik.



Miután az S bemenetet a —CAS vezérli, ha a —CAS magas szintű, a láb kimenet a mégegyszer invertált 6-os címbitet, alacsony —CAS jel esetén pedig az A14 címbitet kapja. A 7-es láb kimenet végzi a —A7 és a —VA15 közötti megfelelő átkapcsolást. A multiplexer invertálása folytán ez a jel A7-ként vagy A15-ként jelenik meg.
Az U14 15-ös lába az AEC jellel van összekapcsolva. Ennek jelölése —OE, (Output Enable).
Ha az AEC magas, az U14 kimenetei lekapcsolnak, vagyis az ún. Tri State állapotba kerülnek. Ez fontos, mert magas AEC jel esetén a processzor lefoglalja a buszt, s a címeit az U13 és az U25 multiplexeren keresztül helyezi ki a buszra.
Ha viszont az AEC alacsony, a VIC lefoglalhatja a buszt, és ekkor az U14 kimenetei felszabadulnak.

16 szín négy bittel — a szín-RAM

Ha mind az 512 lehetséges karaktert, s ráadásul még 16 különböző színben szeretnénk ábrázolni, szükségünk lesz még négy további adatbitre. Erre szolgál a VIC 35-ös, 36-os, 37-es és 38-as lába. Ezekhez a lábakhoz csatlakozik az adatvezetékeivel az U6 Color-RAM. Az U6-os integrált áramkör egy 4096 tárrekesszel rendelkező statikus RAM. Az egyes rekeszek egy-egy bitet tárolhatnak. Minden négy tárrekeszhez egy címet rendelünk.
A címzés a —CS jel segítségével valósul meg, az U6 csatlakozásnál. Ha ez a csatlakozás alacsony szintű, a címzés a RAM-ra vonatkozik, és az adatvezetékek kilépnek a Tri-State állapotból.
A —CS jelet az U27 ÉS kapu két különböző módon állítja elő.
Bár egy kissé furcsán hangzik, ez az ÉS kapu a kapcsolásban VAGY kapuként működik. Az ÉS kapu a kimenetet mindig magasra állítja, ha az összes bemenete magas. Ezt a logikát egy kissé megfordítva, azt is mondhatjuk, hogyha egy másik bemenet VAGY-ja alacsony, akkor a kimenet is alacsony. A CBM 64-es is ezt az üzemmódot használja.
A szín-RAM a SD800-tól SDBFF-ig terjedő címterületet foglalja el. Ha az AEC jel magas, a processzor lefoglalja a buszt. Ezáltal az ÉS kapu egyik bemenete is magas lesz. Ha a processzor nem a szín-RAM-ot igényli, az U15 dekódoló —COLOR kimenete is magasszintű. Ezzel a szín-RAM —CS bemenete átvált magas szintre, s a címzés a szín-RAMra mutat. Ha a processzor a szín-RAM-ot el akarja érni, a megfelelő tárcímeket kihelyezi az adatbuszra. A dekódolás ugyanúgy megy végbe, mint a VIC-nél, azzal az eltéréssel, hogy az A11 címbit értéke 1. Ezáltal az U15 dekódoló —COLOR kimenete alacsony szintű lesz. Most az ÉS kapu egyik bemenete alacsonyra vált át, s így a kimenet is, és így a címzés a szín-RAM-ra mutat. Mivel az AEC ekkor magas, az U16 integrált áramkörben levő négy analógkapcsoló zárt állapotban van, s a szín-RAM adatvezetékei a processzor négy alacsonyabb értékű adatvezetékével vannak összekapcsolva. A szín-RAM tehát írható és olvasható. Ha az AEC jel alacsony és a VIC veszi át a buszt, az analóg kapcsolók nyitnak. Ezzel egyidejűleg az U27 ÉS kapu 8-as lábkimenete alacsonyra vált, s a szín-RAM ismét elérhető, de most a VIC számára. A VIC azonban csak az A8—A11 címvezetékeken van összekapcsolva, az A0-A7 további címbiteket máshonnan kell biztosítani. Ezt a feladatot az U26 integrált áramkör látja el. Ez a 74LS373 jelölésű TTL-integrált áramkör közbenső memóriát tartalmaz. Az IC bemenetei a multiplexelt címbusszal vannak összekapcsolva. Ha a —RAS alacsony, sor kerülhet az adatok tárolására. Ez az az időpont, amikor az alacsonyabb értékű címbyte a buszon van. Az U16 kimenetei a processzor-busz alacsonyabb értékű címbyte-jával van összekapcsolva és a címinformációkat továbbítják, ha a processzor Tri-State állapotban van.
Így a szín-RAM-ot a VIC is címezheti.
A közbenső tár is kapcsolatot tart fenn az AEC jellel.
Az U16 az 1-es lábon keresztül ezt a jelet magas szintre állítja, s nagy ellenállású állapotba kerülnek, és nem zavarják a processzort.
Ha a fentieket alaposan átgondoljuk, felmerül egy érdekes kérdés. Miként lehetséges, hogy az A0-A13 multiplexelt címbuszokon kívül a VIC még a 23-as, 32-es, 33-as és a 34-es lábra kapcsolt A8-A11 címvezetékekkel is rendelkezik?

A VIC-nek minden képernyőtárbeli címhez egy színtárbeli címet ($D800-tól $DBFF-ig) is igénybe kell vennie.
A két különböző tármező egyidejű eléréséhez egy másik buszra van szükség. A hiányzó buszt a négy különválasztott címbit adja.

A karaktergenerátor

A képernyőn minden karakter 8 * 8 képpontból áll. Ha minden ponthoz hozzárendelünk egy bitet, egy karakter ábrázolásához 8 byte-ra van szükségünk.
Első hallásra ez kissé furcsának tűnik, hiszen tudjuk, hogy a képernyőtárban a rendszer minden karakterhez csak egy byte tartozik.
Vizsgáljuk meg közelebbről ezt a kérdést, hogy az összefüggések érthetőbbé váljanak!
Nézzük meg közelebbről az "A" betűt.
Töröljük a képernyőt, majd nyomjuk le az "A" billentyűt.
A tv (vagy monitor) minőségétől függően többé, kevésbé felismerhetjük a karakter pontmintáját.

	1	2	3	4	5	6	7	8
A.	.	.	.	*	*	.	.	.
B.	.	.	*	.	.	*	.	.
C.	.	.	*	.	.	*	.	.
D.	.	*	*	*	*	*	*	.
E.	.	*	.	.	.	.	*	.
F.	.	*	.	.	.	.	*	.
G.	.	*	.	.	.	.	*	.
H.	.	.	.	.	.	.	.	.

Az A-tól a H-ig terjedő sorokban minden egyes "*"-nak megfelel egy 1 értékű, a "."-nak pedig egy 0 értékű bit. Minden sorban 8 bit van, vagyis 1 byte. 8 sorral számolva összesen karakterenként 8 byte-ot kapunk. Ezt a 8 byte-ot tárolja a karaktergenerátor.
A képernyőtárban a rendszer a karakter helyett csak egy kódot tárol, ami valóban elfér egy byte-on. A képernyőkód tulajdonképpen egy cím, nevezetesen a karakter ROM-on belüli kezdőcíme. Ahhoz, hogy mind a nyolc byte-ot megcímezhessük, további címvezetékek szükségesek. Ezt a feladatot a speciális címbusz A8-A10 vezetéke látja el.
A címbusz még szabad A11-es vonala különleges szerepet kap. A képernyőtár egy byte-ján tárolható legnagyobb számérték 256, amivel csak 256 különböző karakter címezhető, holott a Commodore 64-es 512 különböző karaktert képes ábrázolni.
Ez a lehetőség a következőképpen magyarázható:
A CBM 64-es két karakterkészlettel rendelkezik. Az első a bekapcsolás után használt karakterkészlet a ,nagybetűket és a grafikus karaktereket tartalmazza, azaz összesen 128 karaktert. Minden karakter ábrázolható inverz módban is, ez összesen 256 lehetséges karakter.
A második ábrázolási módot a COMMODORE és a SHIFT billentyűk lenyomásával kapcsolhatjuk be. Ez a karakterkészlet tartalmazza a nagy- és kisbetűket. Ebből ismét 128 különböző karakter adódik, amit ismét megduplázhatunk az inverz karaktereket is figyelembe véve. Összesen tehát valóban 512 különböző karakterrel dolgozhatunk.
A két karakterkészlet közötti átkapcsolást a még szabad A11-es címbit végzi el.

A processzor és a RAM

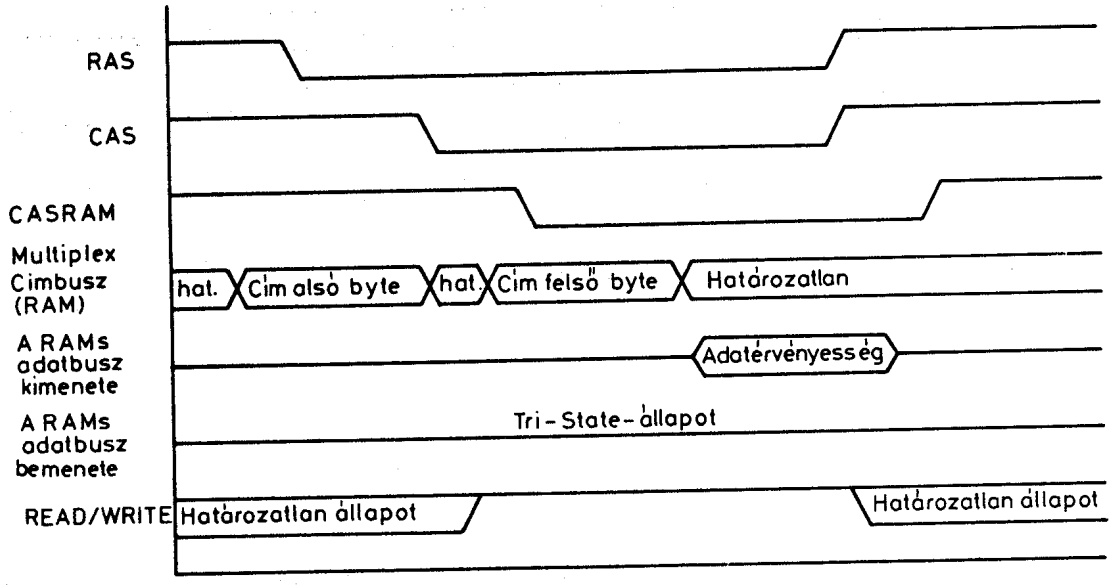
Eddig azzal az esettel foglalkoztunk, amikor a 64 k-s tárat a VIC használja, nem esett azonban még szó azokról a munkafolyamatokról, amikor a processzor akar hozzáférni a RAM-hoz. A

processzor olvasási elérései hasonlóak a VIC hozzáférésekhez. A R/-W jel mindkét esetben magas szintű (olvasásnál magas, írásnál alacsony).
Foglalkozzunk először az olvasási elérésekkel.
Mint azt a RAM-VIC interface leírásából tudjuk, a RAM egy multiplexelt címbuszt igényel. Ezt az igényét azonban a processzor nem tudja kielégíteni. A multiplexelést további integrált áramkörökkel kell elvégezni.
Ezt a feladatot két 74LS257 típusú integrált áramkör, az U13 és az U25 látja el.
Ezeknek az integrált áramköröknek a működése ugyanolyan, mint az U14-é (leírását lásd a RAM-mal és a VIC-kel foglalkozó fejezetben). Az U14-eshez viszonyított eltérés mindössze annyi, hogy ezek a multiplexerek nem invertálják a kimenő jeleket.
A két multiplexer IC bemeneteire A0-tól egészen az A15-ig rá van kapcsolva a teljes processzor-címbusz. A bemenetek kapcsolása olyan, hogy a kiválasztó (Select) jel mindig átkapcsol az A0-ról az A8-ra, az A1-ről az A9-re, és így tovább. A RAM címzése is három fázisra bontható fel.
Az első fázisban a multiplexer Select bemenete magas. Ez az alsó címbyte-ot a RAM-okra kapcsolja.
A –RAS jel lefutó homlokával a byte-ot átveszik a RAM-ok. Rövid idő elteltével a –CAS jel is alacsony szinten lesz. A multiplexerek átkapcsolnak, a másik bemenetük átvált a megfelelő kimenetekre és így a RAM-ok megkapják a magasabb értékű címbyte-ot.
Az AM egy kissé késlelteti a –CAS jelet. A –CASRAM kimenet itt tulajdonképpen a –CAS feladatát veszi át.
A –CASRAM lefutó homloka most a cím felső byte-ját tárolja a RAM-okban.
A RAM-okban a rendszer megkeresi a megcímzett tárrekeszeket és az adatok megjelennek az adatbuszon.
A processzor írási elérése egy lényeges ponton eltér az olvasási ciklusoktól.
Írásnál, miután a processzor a megfelelő tárrekesz címét kihelyezte az adatbuszra, a R/-W processzor-láb alacsony szintű lesz. Így kapja meg a RAM azt a jelzést, hogy az adatbuszon levő byte-ot ebben a rekeszben kell tárolnia.
Az alkalmazott RAM-modulok egy meghatározott követelményt támasztanak az említett R/-W jellel szemben.
A R/-W jel csak ,azután vehet fel alacsony szintet, amikor már a –RAS is alacsony. A –CASRAM azonban még magas. Az R/-W alacsony szintjének a –RAS és a –CASRAM lefutó élei között kell kialakulnia.
A –RAS és –CAS, valamint a –CASRAM jelek időbeni lefutása ugyanolyan, mint az olvasási eléréseknél.
A fentiekben vázolt folyamatok megértését segítik a következő ábrák (306–307. old.).

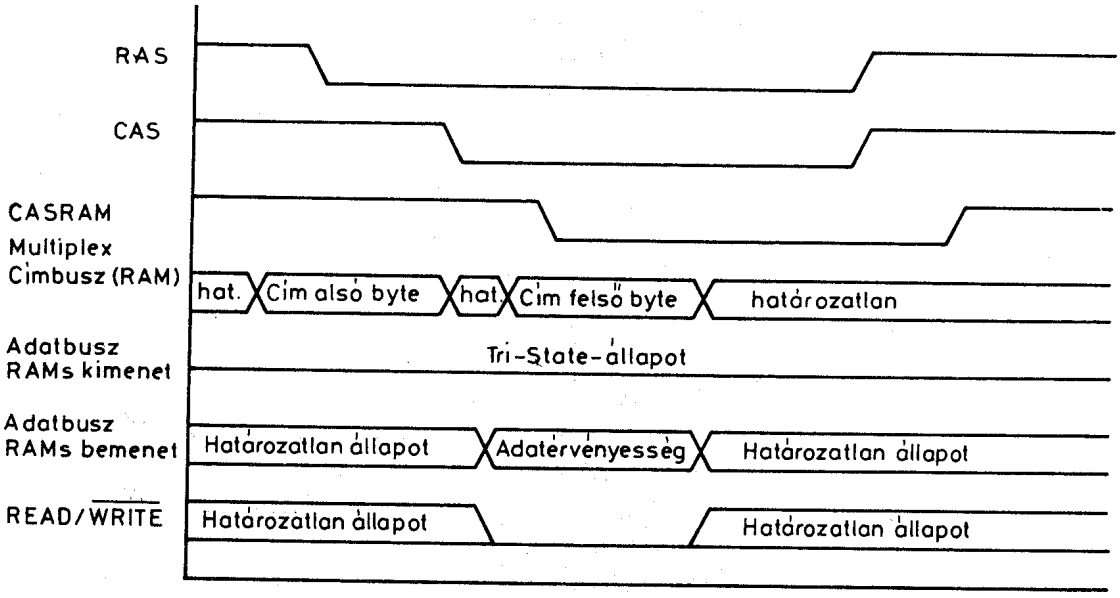
A 6581-es SID

Ez az integrált áramkör pontosan ugyanolyan, mint a VIC, a félvezetőipar lehetőségeinek egy kiemelkedő példája. A SID a CBM 64-esen fantasztikus hangzási lehetőségeket biztosít. Néhány évvel ezelőtt egy szintetizátor egymagában is igénybevette volna a Commodore 64 teljes nyomtatott áramköri kártyáját.

Olvasás hozzáférés a dinamikus RAM-hoz



Íráshozzáférés a dinamikus RAM-hoz



A 6581-es lábkiosztása a következő:

Láb	Jelölés	Feladat
1	CAP_{1A}	frekvenciaszűrő külső kondenzátora
2	CAP_{1B}	lásd az 1. lábat
3	CAP_{2A}	lásd az 1. lábat
4	CAP_{2B}	lásd az 1. lábat
5	—RES	bemenet, reset-jel (REset-Signal)
6	02	bemenet, ütemjel
7	R/—W	bemenet, Read/-Write
8	—CS	bemenet, Chip Select
9—13	A0—A4	bemenet, 0—4 címbit
14	GND	föld, üzemi feszültség
15—22	D0—D7	kétirányú 0—7 adatbitek
23	POTY	bemenet, 2-es AD-átalakító
24	POTX	bemenet, 1-es AD-átalakító
25	Vcc	+ 5 V üzemifeszültség
26	EXT IN	bemenet, külső jelforrás
27	AUDIO IOUT	szintetizátor-kimenet
28	Vdd	+ 12 V üzemi feszültség

A legtöbb jelet már ismerjük az előző fejezetekből. Eddig azonban még nem fordult elő a CAP_{1A}-tól a CAP_{2B}-ig terjedő jelölés. Mint a kapcsolási rajzon is látható, ezekre a csatlakozásokra van rákapcsolva a C10 és C11 kondenzátor. Ezek a kondenzátorok a SID-ben integrált frekvenciaszűrőkhöz szükségesek.
A szűrőt és annak feladatát mindenki ismeri. Egy kávészűrőnek például az a feladata, hogy a vizet és a kávépor vízben oldódó anyagait áteressze, s a kávépor vízben nem oldódó részeit pedig visszatartsa.
Ugyanígy működik egy elektronikus frekvenciaszűrő is. Bizonyos frekvenciákat átenged, más frekvenciákat pedig visszatart.
Négy különböző frekvenciaszűrő ismeretes, az aluláteresztő, a felüláteresztő szűrő, a sávszűrő és a sávzárószűrő.
Az aluláteresztő szűrő csak egy meghatározott maximális frekvenciáig engedi át az alsó frekvenciákat. A sztereoberendezésekben ezt a funkciót a basszusszabályozó látja el. Ezzel a szűrővel állítható be az áteresztendő legnagyobb frekvencia, az ún. határfrekvencia.
A felüláteresztőszűrő ennek pontosan a fordítottja, ez a szűrő egy meghatározott frekvenciánál nagyobb frekvenciákat ereszt át. (Sztereoberendezésekben az ún. Trebleszűrő, illetve a magasság-szabályozó.)
A sávszűrő és a sávzárószűrő is egymással ellentétes hatást váltanak ki.
A sávszűrő nem más, mint az aluláteresztő és a felüláteresztő szűrő együttese. Egy meghatározott fekvenciánál nagyobb frekvenciákat áteresztí, de csak egy maximális frekvenciáig. A sávzárószűrő egy meghatározott frekvenciatartományban az összes frekvenciát lezárja. (a sztereoberendezésekben ennek megfelel a zajszűrő, amely csak egy meghatározott frekvenciát, a hálózati frekvencia 50 Hz értékét szűri ki.)
A felsorolt szűrők a SID-ben programozhatóak.
Az U18 Reset-bemenete, vagyis az 5-ös láb az integrált áramkört egy meghatározott állapotba hozza. Mint már tudjuk, a bekapcsolást követő kb 0,5 másodperc elteltével ez a csatlakozás alacsony lesz, így a 6581-es összes regisztere törlődik.
A Reset nélkül a bekapcsolást követően a regiszterek tartalma véletlen értékű lenne, s az audiokimeneten jelentkező véletlen jel kárt okozhatna a csatlakoztatott tv-készülékben vagy erősítőben.

A 02-es jel frekvenciájából alakul ki a SID összes hangfrekvenciája. A többi külső egységhez hasonlóan itt is a 02-es jel adja a processzor írási és olvasási eléréséhez szükséges órajelet.
Itt is a R/-W vonaltól függ, hogy a SID-ben levő regiszterekben írásra vagy olvasásra van szükség. Magas jelszint az olvasásra, alacsony jelszint pedig az írásra utal. A művelet elvégzésének előfeltétele a SID pontos címzése. A 6581-es címterülete $D400-tól $D7FF-ig terjed. Ezt a címmezőt a VIC-hez hasonlóan az AM és az U15 integrált áramkörbe épített dekódoló adja. Amikor a processzor a címterületről egy címet kihelyez a buszra és a —CASRAM jel magas, a 74LS139 5-ös lába alacsonyra vált, s ezzel egyidejűleg a SID —CS bemenete is alacsony szintű lesz.
A SID egyes regisztereinek címzéséhez igénybe kell vennie az A0-tól A4-ig terjedő 5 címvezetéket.
Ha a címbitek mindegyike alacsony és a —CS jel a SID-et választja ki, a 0-regiszter írható vagy olvasható; ha csak az A0 címbit magas, ez az 1. regiszterre utal és így tovább. Hasonlóan hivatkozhatunk a többi regiszterre is.
A 15-től 22-ig terjedő lábakhoz csatlakozó D0-D7 adatvezetékek össze vannak kapcsolva a processzor adatbusszal. Amíg a —CS magas, a SID adatvezetékei Tri-State állapotban vannak. Ha viszont a —CS alacsony szintű, az R/-W dönti el, hogy az adatvezetékek bemenetként (regiszterek írásakor), vagy kimenetként (regiszterek olvasásakor) működjenek. A POTX és a POTY az A/D átalakító bemenetei.

Mit is jelent tulajdonképpen az A/D?

Az A/D átalakító az analóg-digitális rövidítése. Egy digitális jel két állapotot tud működtetni, a magas és az alacsony állapotot, s mint tudjuk a CBM 64-esben és még sok más egyéb digitális készülékben a + 5 V a magas, a 0 V pedig az alacsony szintet jelöli.
Az analóg jel már korántsem ilyen egyértelmű, mert egy adott intervallumban tetszőleges értéket felvehet.
Ugyanakkor gyakran szükség van arra, hogy a gép egy analóg jelet képes legyen fogadni, értelmezni és feldolgozni. Az analóg jelek feldolgozásának lehetőségét már beépítették a CBM 64-es számítógépbe.
Az A/D átalakítókat elsősorban a forgatható szabályozógombok (Paddle-k) illesztésére használjuk.
A szabályozó lényegében egy változtatható ellenállás, vagyis potenciométer, amit röviden potméternek szoktunk nevezni. A potenciométer ellenállásértéke az elforgatással változik. A paddle-kba beépített potenciométerek minimális ellenállása kb. 100 Ω. A két érték között az ellenállás elméletileg tetszőleges értéket felvehet.
Az A/D átalakító ebből az ellenállás értékéből egy digitális jelet — esetünkben egy 1 byte-os jelet — állít elő, ami egy SID-regiszterből leolvasható.
Az átalakítás a C48-as és a C93-as kondenzátorok segítségével megy végbe.
Ezek a kondenzátorok 0,25 ms. alatt töltődnek fel a potenciométereken keresztül. Ha a kondenzátorok feszültsége nagyobb, mint a SID-ben előállított összehasonlító feszültség, a SID-ben leáll egy számláló, s a számláló állása lesz a beállított ellenállás mértéke. Minél nagyobb a potenciométer ellenállása, annál lassabban töltődik fel a kondenzátor feszültsége annál később éri el a referencia-feszültség szintjét. Így a számláló hosszabb időn keresztül nő és értéke nagyobb lesz.
Ha az ellenállás túl nagy (kb. 200 kΩ), a mérési idő alatt az A/D átalakító bemeneti feszültsége nem éri el a referencia-feszültséget. Ekkor a számláló a végértékéig fut, s az A/D regiszter értéke 255 lesz.
Ha viszont az ellenállás a megengedettnél kisebb (kb. 200 Ω), a kondenzátor olyan gyorsan feltöltődik, hogy a számláló azonnal leáll, s így a regiszter értéke 0 marad.
A 0,25 ms. mérési idő elteltével a megfelelő A/D kimeneten keresztül a kondenzátorok ütésszerűen kisülnek. A számláló most 0-ra állítódik, s a további 0,25 ms. elteltével egy új

mérési ciklus indul. Egy teljes ciklus 0,5 ms-t igényel, s a rendszer egy másodperc alatt 2000-szer méri az aktuális ellenállás értékét.
Az ellenállás nem lehet kisebb 100 Ω-nál. Ellenkező esetben a kondenzátorok kisülésekor fellépő áram túl nagy lenne, s a bemeneti kisütő-fokozatok használhatatlanná válnának.
A POTY és a POTX bemenet azonban nem közvetlenül csatlakozik a Commodore 64-es gép aljzataira. A két bemenet az U28-as integrált áramkör 2-es, 3-as, 9-es és 10-es lábára van kapcsolva Ez a 4066 jelölésű „CMOS-modul" integrált áramkör négy ún. analóg-kapcsolót tartalmaz, így két paddle párt, vagyis összesen négy potenciométert csatlakoztathatunk a géphez.
Egy analóg kapcsoló úgy működik, mint egy relé. Ha a vezérlőbemenetre feszültség érkezik, az analóg-bemenet kimenetre kapcsol át, s a kapcsoló zár. Ha viszont a vezérlő-bemenetet a földre kapcsoljuk, a bemenet és a kimenet között nem jön létre kapcsolat, ami a kapcsoló nyitott állapotát jelenti.
Az analóg bemenetek a CN8 és a CN9 vezérlő-kapuval vannak összekapcsolva. A vezérlő-kapuknál az 5-ös és a 9-es érintkezőre kapcsolhatók a paddle-ok.
A vezérlő-bemenetek feladatát az 5-ös, 6-os, 12-es és a 13-as láb látja el. A 13-as láb ellenőrzi az 1-es és 2-es csatlakozás közötti 1-es kapcsolót, az 5-ös láb a 4-es és 3-as közötti 2-es kapcsolót, a 6-os láb a 8-as és 9-es közötti 3-as kapcsolót, s végül a 13-as láb a 11-es és 10-es csatlakozás közötti 4-es kapcsolót. Ha viszont a CIA 9-es lábon a jel magas szintű, az 1-es és a 2-es analóg kapcsoló zár és a CN8-ra kapcsolt paddle-ok átváltanak az A/D átalakító bemeneteire.
Eddig még nem beszéltünk a 6581-es EXT IN és AUDIO OUT csatlakozásáról.
Az AUDIO OUT a szintetizátor kisfrekvenciájú kimenete.
Itt állnak rendelkezésre a szintetizátorban előállított hangok és zörejek. Maximális hangerőn a kimenőjel 2Vss. A Q8 tranzisztor emitter-követőként csatlakozik a kimenethez. A tranzisztor emitteréről a jelet az R38 ellenállás leveszi, így a tranzisztor mentes a feszültségerősítéstől. Így a CN5 8-pólusú video-audio-csatlakozó 3-as lábkimenetén is 2Vss értékű jelet kapunk.
Erre a kimenetre közvetlenül rákötheto egy kis 8 Ω-os hangszóró, de a hangerő rendkívül alacsony szintű. A megfelelő hangerő biztosítása érdekében legcélszerűbb, ha a jelet egy sztereoberendezésbe vagy egy jóminőségű táskarádióba továbbítjuk, de használhatjuk a tv-készülékbe beépített hangszórót és a képpel átvitt hangjelet is.
Az EXT IN segítségével külső jeleket olvashatunk a szintetizátorba. Külső jelek lehetnek például a mikrofon-jelek, amelyeknek az előzetes felerősítéséről egy kis erősítő gondoskodik. Megfelelő felerősítés után egy gitár vagy egy orgona is szolgáltathatja a bemenőjelet, vagy például egy második SID, vagyis egy másik CBM 64-es.
A bemenőjellel szemben támasztott egyetlen követelmény az, hogy a jel nem haladhatja meg a 3 Vss-értéket.
Ezt a bemenetet a C12-es kondenzátor kapcsolja össze a CN5 8-pólusú Audio-Video 5-ös érintkezőjével.

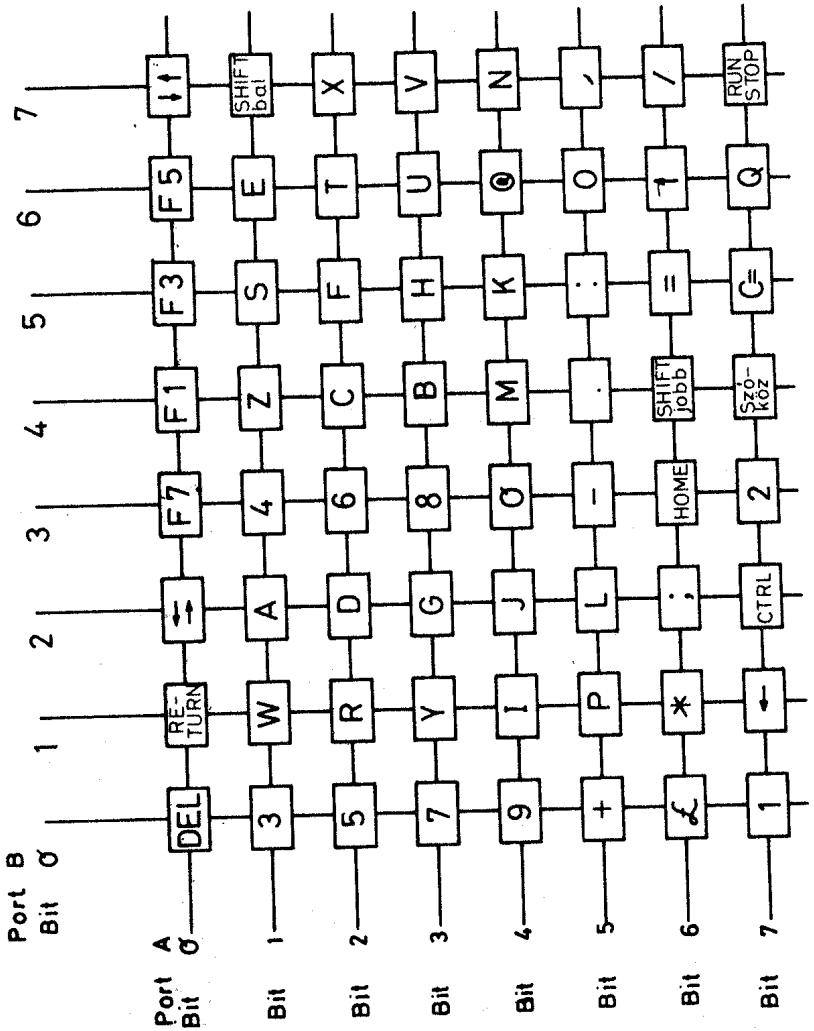
A két 6526-os CIA (Complex Interface Adapter)

Az U1 és U2 jelölésű modulok számtalan feladatot látnak el a CBM 64-es rendszerben. Ez a két modul szervezi a billentyűzet és a botkormány lekérdezését, a soros buszon lezajló adatforgalmat, az RS232-es soros illesztő tevékenységét, az analóg bemenetek kapcsolását, vezérli a kazettás egységet, előállítja a VIC számára az A14-es és A15-ös címbiteket stb. A 40 pólusú 6526-os CIA a következő részekből áll:
— 16 külön-külön programozható input/output vonal
— két intervallum-óra, egy programozható riasztó idővel ellátott valós idejű (real time) óra
— egy nyolcbites léptetőregiszter a soros input/outputhoz

A CIA lábkiosztása:

Láb	Jelölés	Feladat
1	GND	föld — üzemi feszültség
2–9	PA0–PA7	A input/output port, 0–7 bit
10–17	PB0–PB7	B input/output port, 0–7 bit
18	–PC	kimenet, Port Control
19	TOD	bemenet, Time Of Day
20	Vcc	+ 5 V üzemi feszültség
21	–IRQ	kimenet, Interrupt Request
22	R/–W	bemenet, Read/-Write
23	–CS	bemenet, –Chip Select
24	–FLAG	bemenet, lásd a Port Controlt
25	02	bemenet, rendszerütem
26–33	D7–D0	processzor adatbusz
34	–RES	bemenet, Reset-jel
35–38	RS3–RS0	bemenet, Register Select
39	SP	kétirányú soros port
40	CNT	kétirányú CouNT (számlálás)

A PA0-PA7 és a PB0-PB7 vonal — a 16 kétirányú I/O-vezeték.
A mindenkori programozástól függően ezek a vezetékek bemenetet vagy kimenetet képviselnek.
Az 1 CIA ezt a 16 I/O vonalat a billentyűzet csatlakoztatására szolgáló dugaszra kapcsolja. A billentyűzet egy 8 * 8 vonalból álló mátrix. Ha megszámoljuk a Commodore 64-es billentyűit, az eredmény 66. A 8 * 8 mátrixban azonban csak 64 billentyű kérdezhető le. Ezt a problémát a RESTORE és a SHIFT LOCK billentyű oldja meg.
A RESTORE billentyűt a mátrix nem tartalmazza, amint azt már említettük, ez a billentyű az U20 bemenetét kapcsolja a földre és egy NMI-jelet állít elő.
A SHIFT LOCK billentyű egyszerűen párhuzamos kapcsolatban van az egyik SHIFT billentyűvel, s így nem igényel saját helyet a mátrixban.
Az egyes billentyűknek a mátrixban elfoglalt pontos helyét az ábrán láthatjuk.
A billentyűzet lekérdezésének megértéséhez egy rövid előzetes magyarázat szükséges.
Ha a CIA valamelyik portbitje bemenet, de nem foglalt, a CIA ennél a csatlakozásnál magas jelszintet észlel.
A PA0-PA7 vezetékek kimenetként vannak kapcsolva, a PB0-PB7 vezetékek pedig bemenetként.
Amikor az operációs rendszer le akarja kérdezni a billentyűzetet, az A port csatlakozásai rövid időre alacsony szintűek lesznek. Ha például a billentyűzeten a "H" betűt nyomjuk le, ebben a pillanatban a B port 5-ös bitje is alacsony lesz. A gép ebből ismeri fel, hogy az F3, S, F, H, K, :, = vagy a COMMODORE billentyű valamelyikét nyomtuk le.
Hogy pontosan melyiket, azt a rendszer ebben az időpontban még nem tudja meghatározni.
Egy billentyű észlelésekor azonban az A port kimenetei egymás után alacsonyra váltanak. A kimenetek minden egyes szintátkapcsolása után a rendszer megvizsgálja, hogy a B port bemenetei között van-e alacsony.
A mi példánkban ez az eset akkor áll fenn, ha az A port 3. bitje alacsony lesz. A rendszer ebből pontosan meg tudja állapítani a lenyomott billentyű pozícióját a mátrixon belül. A billentyűk pontos elrendezését és a CIA csatlakozásukat is mutatja az ábra.
A botkormányt is az 1-es CIA kérdezi le.
A botkormány 5 kapcsolót tartalmaz. Ezek közül 4 a négy különböző irányt adja, az 5. kapcsoló pedig az ún. tüzelőgomb. Ezek a kapcsolók nem mátrixban helyezkednek el, hanem földre kapcsolnak egy-egy portbitet.
Az 1-es botkormány a B port 0–4., a 2-es pedig az A port 0–4. portbitjeire van kapcsolva.
Az 1 CIA A port-ja a 6. és a 7. bittel még a paddle-ok átkapcsolásáról is gondoskodik, s ha még



emlékszünk, az ezzel kapcsolatos tudnivalókat az A/D átalakítóval foglalkozó fejezet tartalmazza.

A CIA-k 19-es lábának jelölése TOD. A TOD az a bemenet, amely a CIA-ba beépített valós idejű órát ütemjelekkel ellátja Erre a bemenetre érkezik az U27 kapu segítségével előállított 50 Hz-es négyszögjel. A CIA ezt a frekvenciát 5-tel osztja. A 10 Hz másodpercenként 10 impulzust jelent, minden impulzus egy tizedmásodperc, ami egyben a legkisebb mérhető időegység.

A 21-es láb jelölése —IRQ. A VIC-hez hasonlóan, ez a láb itt is megszakítások előállítására szolgál. A CIA 1 21-es lába a processzor —IRQ bemenetével van összekapcsolva.

A láb alacsony, ha a CIA-ban programozható események lépnek fel.

Az R/-W — a 22-es láb — itt is az adatátviteli módot vezérli.

Olvasás közben a láb jelszintje magas, íráskor alacsony.

Ha a $DC00-tól $DFFF-ig terjedő tárterületen a processzor műveletet végez, és ugyanakkor a —CHAREN magas, az U15 dekódoló 7-es lába alacsony lesz. Emiatt az U15-ben levő másik dekóder Enable-bemenete alacsonyra vált és a 2-es dekódoló felszabadul. Az A8 és az A9 címbit a 2-es dekóder 13-as és 14-es láb-bemenetére van rákapcsolva. A címbitek négy kisebb címmezőre osztják fel az 1-es dekódoló által adott címterületet. A kapott címmezők mindegyikének kapacitása 256 byte.

Az első 256 byte a $DC00-$DCFF címmezőt foglalja le (U15 12-es lába) és a CIA 1-et, a másik mező pedig a $DD00-tól $DDFF-ig terjed (11-es láb) és a CIA 2-őt választja ki. A két szabadon kódolt címmező IO1-ként és IO2-ként érkezik a bővítő modul (Cartridge Expansion) 7-es és 10-es csatlakozására. A IO1 a CP/M bekapcsolásához, a IO2 pedig a későbbi bővítési lehetőségekhez szükséges. Az utóbbi tármezőt felhasználhatjuk saját elképzelésünk szerint.

A két CIA valamelyikének kiválasztása után a RS0-RS3 vonalak határozzák meg, hogy melyik belső regisztert kell igénybe venni. Az RS0-RS3 bemenetek a processzorbusz négy alacsonyabb értékű címbitjére vannak kapcsolva.

A D0-D7 adatvezetékek (CIA-k 32—36 csatlakozásai) a processzor-adatbusszal tartanak fenn kapcsolatot. Ezeken keresztül zajlik az adatforgalom, a regiszterek írása és olvasása.

A CIA 1 —FLAG jelölésű csatlakozása kettős feladatot lát el.

Egyrészt össze van kapcsolva a CN3 kazetta-port D és 4 csatlakozásával. Ezen a vezetéken keresztül fogadja a processzor a kazettás magnetofontól érkező jeleket. Másrészt a —FLAG bemenet kapcsolatot tart fenn a soros busz 1-es lábával. A láb jelölése —SRQIN. Ezen az ún. Service ReQuest (szolgálat kérelem) bemeneten keresztül jelezhetik a külső egységek az alapgépnek, hogy mikor szándékoznak adatokat küldeni. Ezt a funkciót azonban a Commodore cég által forgalmazott egyetlen készülék sem használja.

A —FLAG bemenet „éltriggerezett". Minden magas/alacsony váltáskor jelez a CIA és a 21-es lábon létrejöhet egy megszakítás.

A —PC jelölésű bemenet handshake jelet ad a B, vagy együtt az A és B portnak.

Ha a B-portot bemenetként használjuk, a —PC-re továbbított negatív impulzussal átadhatjuk az adatokat a gépnek.

A mindenkori programozásnak megfelelően a —PC bemenet jelszintjének lefutó élével csak a B port vagy mindkét port adatai olvashatók be.

A CIA 1-es ezt a bemenetet nem használja.

A SP jelölésű (Serial Port) 39-es láb a mindenkori programozástól függően tetszés szerint lehet a CIA lépegetőregiszterének bemenete vagy kimenete.

A léptetőregiszter segítségével minden különösebb ráfordítás nélkül csatlakoztathatunk egy soros illesztőt.

A CIA 1-esen ez a vonal a CN2 User Port 5-ös csatlakozásával van összekapcsolva és az RS232 illesztő használja, de a csatlakozás saját célokra is alkalmazható.

A CIA 1 CNT csatlakozása is a User Portra van rákapcsolva, ahol a 4-es lábat foglalja le. Ezen a csatlakozáson keresztül közvetíthetjük (ki/be) a léptetőregiszter ütemét. A csatlakozást a CIA-ba beépített számláló ütembemeneteként is programozhatjuk.

A CIA 2 A portja több feladatot lát el.

Az A port 0. bitje a VA 14 kisegítő video-címbit, az 1. bitje pedig a VA 15. A PA2 az A port egyetlen bitje, amelynek semmilyen szerepe nincs ("Csak" az User-Port M csatlakozására van kapcsolva). A fennmaradó 3-tól 7-ig biteket a soros IEC busz használja. A 3-as bit képezi a kimenetet a —ATN-jel előállításához. Ez a csatlakozás az U8 integrált áramkör 1-es lábára van kivezetve.
Az U8 integrált áramkör 6 ún. puffert — jelerősítőt — tartalmaz, amelyek nem invertálják a jelet. A pufferek ezen kívül még Open Collector kimenetekkel is rendelkeznek, s a kimeneten + 5 V-nak megfelelő munkaellenállást igényelnek. A —ATN-jel 2-es „puffer-kimenet" lába a soros busz 3-as lábával van összekapcsolva, de egyidejűleg az user-port 9-es lábára is csatlakozik. Így a portbitet a saját user-port kapcsolásainknál is felhasználhatjuk, de csak kimenetként.
A 4-es és az 5-ös portbit pufferezéséről szintén az U8 IC gondoskodik. A 4-es portbit a CLK OUT, az 5-ös bit pedig a DATA OUT jel. A két puffer kimenetei a soros busz 4-es és 5-ös lábával vannak összekapcsolva, ahol a 4-es láb a CLK, az 5-ös láb pedig a DATA jelnek felel meg. Nyugalmi állapotban az említett kimenetek magasak. A puffer-kimenetekre a 6-os és 7-es portbitek is rá vannak kapcsolva. Ezt az indokolja, hogy mind a CLK, mind pedig a DATA kétirányú jel, s ezeket nemcsak a "64" állítja elő, hanem egy csatlakoztatott lemezegység vagy nyomtató. A 6-os és 7-es portbit ennek megfelelően bemenetként van programozva és a külső egység pedig a jeleket a földre kapcsolhatja.
A B port mind a nyolc vonala az user porthoz csatlakozik, s így ezek a felhasználó rendelkezésére állnak. A nyolc vonal ki- és bemenetként egyaránt használható. Az user-port programozásával és alkalmazásával kapcsolatos részleteket a könyv 1.6 alfejezete tartalmazza.
Az user-port 8-as érintkezője a CIA 2 —PC vonalával van összekapcsolva (18-as láb). A bemeneten kialakuló negatív él a B port adatainak érvényességét jelzi akkor, amikor a B portot bemenetként programozzuk. Ezzel jelezhetjük a gépnek, hogy az adatokat átveheti.
A CIA 2 —FLAG vonala (24-es láb) a B user-port-érintkezőre van kapcsolva. Ha a B portot kimenetként programoztuk, ez a vonal jelezheti az adatok érvényességét.
A CIA 1-hez hasonlóan a CIA 2 —IRQ kimenete is megszakítást idézhet elő. A CIA 2 által előállított megszakítás azonban —NMI-t indít a processzorhoz, amely szoftveresen le nem tiltható megszakítást eredményez.
A címterület kódolása is úgy történik, mint a CIA 1-nél.
Ezt a feladatot az U15 integrált áramkörben levő dekódoló látja el. A CIA 2 azonban a $DD00-tól $DDFF-ig terjedő címterületet foglalja le.
A 23-as láb által igényelt —CS-jelet a dekódoló 11-es lába állítja elő.
A két CIA 34-es lábára érkezik a rendszer Reset jele is.
Emiatt a bekapcsolás után a külső egységek összes regisztere is kiindulási állapotba kerül.
Az U2-be érkező többi jel különösebb magyarázatot nem igényel, mert feladatuk ugyanaz, mint a CIA 1 megfelelő vonalainak feladata.

A modulátor

A VC 20 készüléktől eltérően a "64" modulátora már gyárilag beépített a készülékbe.
A modulátor kapcsolási rajza sajnos nem áll rendelkezésre, de talán ennek ellenére is meg fogjuk érteni az ebben a részben végbemenő folyamatok lényegét.
A modulátor alapját egy oszcillátor képezi, amely egy UHF-tv-tartományon belüli frekvenciát szolgáltat. Ezt az oszcillátor-jelet a VIC 6569 SYNC + LUB és COLOR jele, valamint a SID 6581 audio jele modulálja. Az így kialakuló jel a „Chinch-csatlakozóhüvelynél" áll rendelkezésre és egy koaxiális kábelen keresztül vezethető a tv-készülék antenna bemenetéhez.
Ne feledkezzünk meg arról, hogy a modulátor nyílásain keresztül látható kiegyenlítő elemek semmilyen körülmények között nem állíthatók át. A modulátor kiegyenlítését gyárilag végezték, s minden bizonnyal ez az optimális.

Az alkalmazott félvezetők jegyzéke

A már szakértői szinten dolgozó felhasználók részére a szerzők összeállították a CBM 64-ben alkalmazott integrált áramköröket, s az alábbi összefoglalás a gyártó cégekre is kiterjed. Az összes integrált áramkör beszerezhető a Data Becker cégtől. A javításokat így bárki saját maga is elvégezheti.
Annak ellenére, hogy minden javítás SÜRGŐS, soha ne fogjunk hozzá a gép javításához anélkül, hogy részletesen ismernénk a gép hardverfelépítését, vagy ne rendelkeznénk a szükséges mérőműszerekkel. Az esetleges sikertelen kísérlet után többnyire jóval többe kerül a javítás, mintha nem nyúltunk volna a géphez.

Jelölés	Típus	Gyártó
U1	6526 CIA	Commodore MOS
U2	6526 CIA	Commodore MOS
U3	2364A BASIC	Commodore MOS
U4	2364A KERNAL	Commodore MOS
U5	2332A CHARACTER	Commodore MOS
U6	2114L—3 COLOR RAM	különböző gyártók
	például OKI	MSM 2114L-3
	FAIRCHAILD	2114L-3
	HITACHI	HM2114L-3
	MOS	MPS2114L-30
	MOTOROLA	MCM2114L-30
	NEC	uPD2114L-1
U7	6510 MPU	Commodore MOS
U8	7406	különböző gyártók
U9	4164 RAM	különböző gyártók
U10		
U11	például NEC	uPD4164-2
U12	MOSTEK	MK4164-10
U21		
U23		
U24		
U13	SN74LS257	különböző gyártók
U14	SN74LS278	különböző gyártók
U15	SN74LS139	különböző gyártók
U16	MC4066	különböző gyártók
U17	825100	Commodore által programozott Signetics
U18	6581 SID	Commodore MOS
U19	6589 VIC	Commodore MOS
U20	556	különböző gyártók
U25	SN74LS257	különböző gyártók
U26	SN74LS373	különböző gyártók
U27	SN74LS08	különböző gyártók
U28	MC4066	különböző gyártók
U29	SN74LS74	különböző gyártók
U30	SN74LS193	különböző gyártók
U31	SN74LS629	különböző gyártók
U32	MC4044	Motorola
VR1	7812 12 V Regler	különböző gyártók
VR2	7805 5 V Regler	különböző gyártók