64K+BASIC PROGRAMOZÁS

VIDEOTON

TV-Computer

VIDEOTON

ELEKTRONIKAI VÁLLALAT
SZÁMÍTÁSTECHNIKAI GYÁRA

21–00767–511 00A

TV–COMPUTER 64K +

Kezelési útmutató és BASIC programozási segédlet kiegészítés

Készült a Forma-Art Kiadó gondozásában
KŐNYOMAT Sokszorosító és Szolgáltató Kisszövetkezet, 88/8
Felelős vezető: ifj. Kasza Ferenc

TARTALOM

Függelék

Bevezetés

A TV–Computer 64k+ a korábban gyártott TV–Computerek korszerűsített, továbbfejlesztett változata. Az új gép természetesen tudja azokat a szolgáltatásokat, utasításokat is, amelyeket a régebbi gépek tudtak. Az új gépek kezelése, BASIC nyelve a korábbi gépekével azonos „Kezelési útmutató" és „BASIC programozási segédlet" című könyvekből ismerhető meg. Ebben a könyvben a gép teljesítményét növelő többletszolgáltatások ismertetése található.

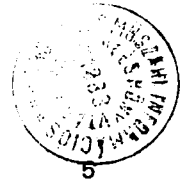
A könyv nemcsak az új BASIC parancsok és utasítások leírását tartalmazza, hanem a gépi kódú programozáshoz szükséges, az operációs rendszerre, rendszerváltozókra vonatkozó tudnivalókat is. Természetesen itt is csak az eltéréseket, új lehetőségeket részletezzük.

A TV–Computer 64k+ hardver alapvetően megegyezik a régebbi gépekével. A bevezetett két módosítás közül az első a lényegesebb: 4x16kB videomemóriát tartalmaz, szemben a korábban beépített 1x16kB videomemóriával. Ez azt jelenti, hogy 4 kép tárolható egyidejűleg, ebből választható ki, hogy melyikre írunk és melyiket jelenítjük meg. A másik módosítás a memórialapozásra vonatkozik.

A felhasználó számára több változást jelent a beépített szoftver módosítása. Az új gépben alkalmazott 2.0 verziójú BASIC és operációs rendszer a grafikai lehetőségekben és a programszerkesztésben nyújt lényeges többleteket. Megjegyezzük, hogy a BASIC 2.0 változata a régebbi gépeknél is alkalmazható, ehhez a gépben levő EPROM-okat kell kicserélni.

Kisebb mennyiségben forgalomba került az általánosan használt BASIC 1.2 javított változata, a BASIC 1.3 is. Ez a könyv az 1.3 változat ismertetését is tartalmazza.

Az ön tulajdonában levő gép típusa a burkolat tetején levő felírat, illetve a burkolat alján található típusszám segítségével azonosítható a következők szerint:



32k–s TV–Computer:

A burkolat tetején levő felírat: VIDEOTON TV–Computer
A burkolat alján a típusszám : 49100

64k–s TV–Computer:

A burkolat tetején levő felírat: VIDEOTON TV–Computer 64k
A burkolat alján a típusszám : 49101

64k+ TV–Computer:

A burkolat tetején levő felírat: VIDEOTON TV–Computer 64k+
A burkolat alján a típusszám : 49108

Bekapcsoláskor a gép kiírja a beépített program változatszámát:

TV–COMPUTER BASIC 1.2

vagy

TV–COMPUTER BASIC 1.3

vagy

TV–COMPUTER BASIC 2.0

Természetesen a későbbiekben a BASIC (és ezzel együtt a változatszám) még módosulhat.

Változások a TV–Computer 64k+ jelű gép hardverében

A TV–Computer 64k+ gép kétféle többletszolgáltatást nyújt az előző 64k-s TV–Computerhez képest:

– A felhasználói RAM 48–64 kB tartományát a processzor „láthatja" a 0–16 kB címtartományban is. Ezt a lehetőséget a 2-es portcímen levő fő-lapozóregiszter b4,b3 bitjeivel lehet elérni. A VT–DOS operációs rendszer működéséhez is szükség van erre.

– 4*16 kB képmemória az előző 1*16 kB képmemória helyett. Ezzel a módosítással együtt bővítettük a memória-lapozási lehetőségeket is. A 4x16 kB képmemória 4 kép tárolását teszi lehetővé. Programból kiválasztható, hogy:

  –– melyik képet írhatja/olvashatja a BASIC, illetve a felhasználó, és
  –– melyik kép legyen látható.

Video lapozóregiszter

Ez az új lapozóregiszter a 15-ös (0Fh) portcímen található. A regiszter b5,b4 bitjeivel választható ki, hogy a 4 kép közül melyiket látjuk a képernyőn. A bitek módosítása azonnal előidézi a képernyőváltást.

Ha a 2-es portcímen levő fő-lapozóregiszter szerint a processzor a 32–48 kB címtartományban videomemóriát „lát", akkor a 15-ös port b3,b2 bitjei választják ki, hogy melyik képet tudjuk itt írni/olvasni. A BASIC mindig ezen a címtartományon keresztül használja a videomemóriát! Ez az oka annak, hogy a b3,b2 bitek módosítása után a BASIC másik képernyőre fog írni.

Új lehetőség, hogy a videomemória a processzor számára a 16–32 kB címtartományban is elérhető. A 2-es portcímen levő fő-lapozóregiszter b2 bitjével lehet ezt elérni. Ha itt videomemória „látható", akkor a 15-ös port b1, b0 bitjei választják ki a négy lehetőség közül a megfelelő videomemóriát.

A videomemória belapozásával óvatosan kell bánni, mert akár a 16–32 kB tartományban, akár a 32–48 kB tartományban eltüntethető a „BASIC szeme elől" a futó program, vagy a szimbólumtábla, vagy a BASIC stack, aminek hatására elvesztjük a gépben levő programot!

A következő példaprogram a 4 képre rendre kiírja: „1.kép", „2.kép", „3.kép", és „4.kép", ezután a billentyűzeten 1-től 4-ig a számokat leütve a megfelelő kép jelenik meg. Ha egyéb billentyűt nyomunk meg, akkor visszaáll a gép alapállapotba (1.kép).

```
10  FOR I=0 TO 3
20  OUT 15,I*4              ! az írt kép kiválasztása
30  CLS
40  PRINT STR$(I+1); ".kép"
50  NEXT I
60  OUT 15,0                ! alapállapot, 1. kép
70  GET A$:A=VAL(A$)
80  IF A<1 OR A>4 THEN GOTO 120
90  N=(A-1)*16              ! b5,b4 bitek beállítása, és
100 OUT 15,N                ! a megjelenített kép kiválasztása
110 GOTO 70
120 OUT 15,0                ! alapállapot, 1. kép
130 END
```

Természetesen az írásra/olvasásra és a megjelenítésre kiválasztott képek lehetnek különbözők is. Például az

```
OUT 15,3*16+2*4
```

azt jelenti, hogy a negyedik képet látjuk (b5, b4=3), de a harmadik képet tudjuk írni/olvasni (b3,b2=2). Ilyenkor a begépelés vakon történik! A harmadik képre vakon felírt szöveg az

```
OUT 15,2*16+2*4          vagy röviden az
OUT 15,40
```

utasítással tehető láthatóvá. Most a harmadik képet látjuk (b5, b4=2), és a harmadik képet tudjuk írni/olvasni (b3,b2=2).

Az utasítások, parancsok, programok begépelése közben végrehajtott képváltás zavarokat okozhat, mert a képernyőszerkesztő nem különbözteti meg a 4 képet:

- az egyik képre írt szöveg a szerkesztő számára akkor is létezik, ha közben átkapcsoltunk másik képre, tehát az előző képre írt szöveget nem látjuk, és
- az átkapcsolás után kiadott CLS parancs a szerkesztő számára törli a szöveget, akkor is, ha visszakapcsolás után a szöveg láthatóvá válik.

A TV–Computer 64k+ lapozási lehetőségeit táblázatban foglaltuk össze. A táblázat jelölései a következők:

LAPO . . . LAP3	:	memórialapok, azaz címtartományok
UO . . . U3	:	a felhasználói memória (USER RAM) 16kB-os egységei
SYS	:	rendszer ROM 16 kB-os fő része
EXT	:	rendszer ROM 8 kB-os kiegészítő része
IOMEM	:	bővítőkártya ROM/RAM, 8 kB
VID	:	a videomemória kiválasztott 16 kB-os lapja
CART	:	dugaszolható programmodul
CSTO . . . CST3	:	bővítőkártya-csatlakozó kiválasztás

A memórialapozásról a „TV–Computer Operációs Rendszer" című könyvben találhatók részletesebb információk.

Megjegyezzük, hogy a TV–Computer bekapcsoláskor csak a Képmemória első 16 kB-os részét teszteli, és csak ezt törli az operációs rendszer, illetve a BASIC. A többi kép tartalma kezdetben véletlenszerű, ezért ajánlatos használat előtt a kiválasztott új képernyőt törölni.

Memórialapozás

LAP0	0000 0 — 3FFF 16383	SYS (00) RESET	UO (10)	CART (08)	U3 (18)
LAP1	4000 16384 — 7FFF 32767	U1 (00) RESET	VID (04)		
LAP2	8000 32768 — BFFF 49151	VID (00) RESET	U2 (20)		
LAP3	C000 49152 — DFFF 57343	CART (00) RESET	SYS (40)	U3 (80)	IOMEM —(CO) EXT
	E000 57344 — FFFF 65535				

I/O cím: 2 (02h) írás: lapozóregiszter

B7	B6	B5	B4	B3	B2	B1	B0
LAP3		LAP2	LAP0		LAP1	–	–
00: CART (00)		0:VID (00)	00: SYS (00)		0:U1 (00)		
01: SYS (40)		1:U2 (20)	01: CART (08)		1:VID (04)		
10: U3 (80)			10: U0 (10)				
11: EXT (C0)			11: U3 (18)				

I/O cím: 3 (03h) írás: bővítő lapozóregiszter

B7	B6	B5	B4	B3	B2	B1	B0
IOMEM		–	–	–	–	–	–
00: CST0 (00)							
01: CST1 (40)							
10: CST2 (80)							
11: CST3 (C0)							

I/O cím: 15 (0Fh) írás: video lapozóregiszter

B7	B6	B5 B4	B3 B2	B1 B0
–	–	CRT	LAP2	LAP1
		00: VID0 (00)	00: VID0 (00)	00: VID0 (00)
		01: VID1 (10)	01: VID1 (04)	01: VID1 (01)
		10: VID2 (20)	10: VID2 (08)	10: VID2 (02)
		11: VID3 (30)	11: VID3 (0C)	11: VID3 (03)

A BASIC 1.3 változásai az 1.2 verzióhoz képest:

A BASIC 1.3-as változata elsősorban az 1.2-es változat hibáit szünteti meg, a felhasználó számára lényeges többletszolgáltatást nem nyújt. A változások a következők:

1. Az 1.2 verzióban a string összehasonlításba beszámított a string hossza, ezért az érdemleges összehasonlításokhoz a stringeket azonos hosszra kellett kiegészíteni (pl. szóközökkel). Most a string hossza nem számít be az összehasonlításba, ezért pl. az „ABC" string most kisebb, mint a „BC" string.

2. Az INPUT utasításban a string maximális hossza most valóban 254 (eddig csak 251 volt).

3. A PRINT USING és a felhasználó által definiált függvények együttes hívásakor fellépő hibák megszűntek.

4. Egy ELSE <sorszám> típusú utasítás végrehajtása az 1.2 verzióban bizonyos esetben 9 bájt stack-mélyítást okozott. Ez a hiba is ki van javítva.

5. Az 1.2 verzióban kettőnél több billentyű egyidejű megnyomása esetenként hibás karaktereket eredményezett. Ez a hiba a SHIFT, CTRL és ALT billentyűk megnyomásakor már nem fordul elő. Hasonlóképpen ezek közül egy le van nyomva egy másik normál alfanumerikus billentyűvel együtt, akkor a billentyűzet kezelő program addig nem vesz figyelembe több billentyűt, amíg ezek közül valamelyiket el nem engedik.

6. A video eszközmeghajtó – amikor a felhasználói koordinátákat képpont koordinátává konvertálja – most kerekít, szemben az előző verzióval, ahol csonkolás történt. Négyszínű üzemmódban él. a PLOT X,O parancs korábban az X=0,1,2,3 értékekre a bal alsó képpontot jelentette, X=4,5,6,7 esetén a második képpontot, stb. Most az X=0,1 esetén jelenti a bal alsó képpontot, X=2,3,4,5 esetén a másodikat, X=6,7,8,9 esetén a harmadikat,stb.

7. Bevezettük a PROGID nevű változót az operációs rendszer számára. A PROGID változó kezdőcíme 3736 (0E98h), és hossza 20 bájt. Ez a terület arra szolgál, hogy egy operációs rendszer alatt futó program bemásoljon egy azonosító stringet, amit a futó program azonosítására lehet felhasználni. A stringet 00h karakterrel kell lezárni. Ezt a területet az operációs rendszer meleg és hideg resetnél is nullázza, és a BASIC a következő stringgel tölti fel:

TV COMPUTER BASIC

Jelenleg ezt a lehetőséget csak a VT–DOS használja. A VT–DOS mielőtt a megfelelő EXT rutincímet beírná megvizsgálja, hogy tényleg a BASIC operációs rendszer fut-e a gépen. Ez a terület az előző verzióban szabad volt.

8. Az operációs rendszer összes eszközmeghajtójánál a blokk írás és olvasásnál megszűnt a HI_MEM túllépésének ellenőrzése. Így lehetőség van olyan pufferek használatára is, amelyek a HI_MEM mutató fölé nyúlnak, vagy teljesen fölötte helyezkednek el. Továbbra sem használható átviteli puffer céljára a felhasználói RAM 48kB fölötti területe, az U3 lap. (Kivétel: lemezműveletek VT–DOS diszk-csatolóval)

9. A soros vonal kezeléssel kapcsolatban változott:
   – A soros vonal inicializálási parancs most már engedélyezi az adás-vételt is.
   – Ha CTRL+ESC (STOP) karakter megnyomása történik soros input közben, akkor az RTS szervíz jel inaktívvá válik, tehát az ellenállomás nem küldhet karaktert.

10. Az 1.2. verzióban az összes karakter pontmátrixa 10 bájton volt megadva. Mivel a 80h alatti kódok esetén a legfelső TV sornak megfelelő bájt mindig nulla, ezért helynyerés céljából ezekhez a karakterekhez csak 9 bájt van letéve a ROM-ba és a rajzoló rutin „tudja", hogy a hiányzó bájt nulla. A 80h kód fölötti karakterek pontmátrixai a RAM-ban változatlanul 10 bájtosak.

A BASIC 2.0 verzió

A BASIC 2.0 minden olyan változtatást tartalmaz, amelyet a „BASIC 1.3 verzió változásai az 1.2 verzióhoz képest" című részben leírtunk. Ezenkívül bizonyos hibák kijavítása csak ebben a verzióban történt meg. Ezek a következők:

– a PRINT USING kerekít,
– a SOUND utasítás nem rontja el a BASIC EXT5 és EXT6 címeket.

A javításokon túlmenően a 2.0 verzió szolgáltatásai jelentősen kibővültek. Az új BASIC és operációs rendszer általában teljesen kompatibilis az előző verziókkal. Minden operációs rendszer funkció és a BASIC szintaxis változatlan maradt. Minden publikált és nem publikált változó címe megmaradt, de eddig nem használt memóriaterületek az új BASIC– és rendszerváltozók számára le vannak foglalva.

Inkompatibilitás forrása lehet, hogy a BASIC szintaxisa nem enged meg olyan változóneveket, amelyek BASIC kulcsszóval kezdődnek (de a változónevek tartalmazhatnak BASIC kulcsszavakat). Mivel a BASIC 2.0 új kulcsszavakat is bevezet, lehetséges, hogy az eddigi érvényes változónevek egy része itt nem lesz használható. Ennek valószínűsége azonban csekély, mivel új kulcsszavaknak meglehetősen hosszú, és ritkán használt szavakat vezettünk be.

Jelölések a BASIC utasítások ismertetésében

Nagybetűs szavak

Ezek a kulcsszavak, a megadott módon kell őket begépelni, de a kis- és nagybetűk tetszőlegesen keverhetők.

Kisbetűs elemek

Ezek paraméterek, amelyeket az utasítások megjelölt helyén kell megadni. Ezek közé számít a vessző és a kerek zárójel is.

Elemek szögletes zárójelben („[" és „] ")

Ezek olyan paraméterek, amelyeket nem kell feltétlenül megadni, a „[" és „]" jelek közti rész elhagyható. A szögletes zárójeleket nem kell begépelni!

Új és kibővített BASIC parancsok, utasítások

A bevezetett új parancsok, utasítások:
- AUTO
- RENUMBER
- FKEY
- ON EXCEPTION GOTO
- EXCEPTION

Új beépített függvények:
- ERRNUM
- ERRLIN
- HEX$
- DEC

Bővített utasítások:
- CONTINUE
- PLOT
  - relatív koordináták
  - RECTANGLE
  - POLYGON
  - ELLIPSE

A továbbiakban ismertetjük a módosításokat, példák a függelékben találhatók.

1.AUTO parancs

AUTO [STEP növekmény] [,AT kezdősorszám]

A program automatikus átszerkesztésére, automatikusan képzett sorszámozású program írására szolgál. A STEP után megadott „növekmény" az új sorszámokhoz használt növekmény, ha nincs megadva, akkor a BASIC 10-nek tekinti. Az AT után megadott „kezdősorszám" lesz az első új sor sorszáma, ha nincs megadva, a BASIC ezt is 10-nek tekinti.

Minden az AUTO által létrehozott új sorhoz megjelenik a sorszám, s közvetlenül utána kerül a kurzor. A sor szövege begépelhető és a RETURN gomb lenyomásával beléptethető. Ha a sor már létezik, akkor nemcsak a sorszám jelenik meg, hanem az egész sor, végén a kurzorral. A sor ezután átszerkeszthető, vagy törölhető, ha nincs rá szükség.

Ha az AUTO által kiírt sorszámot módosítjuk, akkor a következő generált sorszám az új sorszám + a STEP után megadott „növekmény" lesz.

Az AUTO módból való kilépés esetei:

- sorszám nélküli sor beléptetése,
- ha a következő sorszám nagyobb lenne, mint 9999,
- CTRL+ESC megnyomása,
- hibák.

2. RENUMBER parancs

RENUMBER [sortartomány] [,STEP növekmény] [,AT kezdősorszám]

Az utasítás a BASIC programsorok átszámozására szolgál. A „sortartomány" jelöli ki a program átszámozandó részét, hasonló módon mint a LIST és a DELETE utasításoknál. A STEP után megadott „növekmény" lesz az átszámozásnál használt növekmény, ha nem adjuk meg, akkor a BASIC 10-nek tekinti. Az AT után megadott „kezdősorszám" lesz az első átszámozott sor új sorszáma. Ha az AT értéket nem adjuk meg, akkor a számozást a BASIC a „sortartomány"-ban — ennek hiányában az egész programban — létező legkisebb sorszámmal kezdi.

Egy program átszámozása minden változót töröl. A RENUMBER parancs utolsó kell hogy legyen a sorban. A parancs használata során előforduló hibák:

— **„No memory"** (nincs memória)
Nincs elegendő hely a RAM-ban a program átszámozására. Pl. a GOTO 1000 három bájttal több memóriát igényel, mint a GOTO 1. A szükséges memória mérete a programsorok számától függ: soronként 4 bájt. Érdemes ilyenkor kipróbálni, hogy a program több részletben átszámozható-e.

— **„Cannot renumber"** (az átszámozás nem lehetséges)
A programot valamilyen okból nem lehet átszámozni. Ha a hiba érvénytelen RENUMBER paraméter miatt következik be, akkor a BASIC hibás sorként a RENUMBER parancsot írja ki. Ez lehet pl. olyankor, ha az utolsó új sorszám nagyobb lenne mint a maximálisan lehetséges 9999, vagy a „sortartomány" utolsó sora átfedné a következő programsort.
Ha a hibaüzenet után egy programsort ír ki a BASIC, akkor a hiba az adott sorral kapcsolatos. Pl. egy GOTO olyan sorszámra ugrana, ami nincs, vagy a sor hossza az átszámozás után meghaladná a 251 karaktert.

A következő parancsok nem számozhatók át:
AUTO, DELETE, LIST, LLIST, RENUMBER.

3. FKEY parancs

FKEY feladatbillentyű-szám, stringkifejezés

A „feladatbillentyű-szám" értéke 0—9 lehet. Az így meghatározott feladatbillentyűhöz a BASIC hozzárendeli a „stringkifejezés" értékét. A tíz lehetséges feladatbillentyű stringjei számára összesen 100 bájt áll rendelkezésre, beleértve az egyes stringekhez tartozó 1-1 hosszbájtot is. Ha nincs elegendő hely az új string számára, akkor a feladatgombhoz rendelt kifejezés nem fog megváltozni.

A feladatbillentyű elnevezést a LOCK billentyű és egy számbillentyű együttes lenyomására használjuk. A BASIC ennek hatására kiírja a képernyőre a megfelelő feladatbillentyűhöz rendelt stringet. Ha a „stringkifejezés" egy null-string, akkor a feladatgomb lenyomása hatástalan. Bekapcsoláskor ez az utóbbi eset érvényes valamennyi feladatgombra.

Célszerű a gyakran használt, vagy hosszú BASIC parancsokat a feladatgombokhoz rendelni. Például az

```
FKEY 0, "print"
```

parancs a gyakran használt „print" utasítást rendeli a 0-ás feladatgombhoz. Ezután a „print" szó leírása helyett elég a LOCK és a 0 billentyűk együttes megnyomása. Azt is megtehetjük, hogy a stringbe beleírjuk a parancsot záró RETURN kódját is, például:

```
FKEY 1, "run" & chr$(13)
```

Ha most egy üres sorban megnyomjuk a LOCK és az 1-es billentyűket, akkor a képernyőre kiírt „run" parancsot a záró CHR$(13) karakter hatására a BASIC végre is hajtja, tehát már le sem kell ütni a RETURN billentyűt. Természetesen más, a képernyőre hatással levő vezérlőkaraktert is beleírhatunk a „stringkifejezés"-be.

Megjegyzzük, hogy a LOCK billentyű minden egyéb vonatkozásban úgy viselkedik, mint ahogy a „Kezelési Útmutató" című könyvben leírtuk.

4. ON EXCEPTION GOTO utasítás

ON EXCEPTION GOTO sorszám

Ez az utasítás egy hibakezelő BASIC szubrutint definiál. A rutin a „sorszám"-mal megadott sorban kezdődik. Ha a definiálás után valamelyik programsorban hiba lép fel, akkor a beépített hibakezelő helyett a „sorszám"-nál kezdődő BASIC rutin lép működésbe.

Amíg a definiált rutin a hibakezelést végzi, addig minden újabb hibát az operációs rendszer fog kezelni, ez megakadályozza a hibarutin végtelen ciklusban való működését. A hibarutint CONTINUE utasítással kell befejezni, ekkor újra életbe lép a hibakezelés átirányítása a „sorszám"-mal definiált rutinhoz.

Ha a megadott „sorszám" 0, akkor megszűnik a korábbi hibarutin-definíció, és minden hibát az operációs rendszer fog lekezelni, ugyanúgy, mint bekepcsoláskor.

A hibakezelő szubrutin számára az ERRLIN függvény megadja, hogy melyik sorban történt a hiba, az ERRNUM függvény pedig a hiba kódját szolgáltatja.

Lásd még a CONTINUE és EXCEPTION utasítások leírását.

5.EXCEPTION utasítás

EXCEPTION hibakód [,sorszám]

Ezzel az utasítással szándékos hibát lehet előidézni. A „hibakód" lesz a fellépő hiba kódja, értéke 0—255 tartományba kell, hogy essen. Az ERRNUM függvény aktuális értéke is „hibakód" lesz.

Ha szerepel a „sorszám" paraméter, akkor az ERRLIN függvény ezt kapja értékül, ennek hiányában pedig azt a sorszámot, amelyben az EXCEPTION utasítás szerepel. A „sorszám" egy létező programsor száma kell, hogy legyen!

Ennek az utasításnak a segítségével programból, tetszőleges programfeltételek mellett el lehet lehet indítani a korábban definált hibakezelő BASIC szubrutint. A „sorszám" paraméter ilyenkor arra szolgál, hogy kijelölje a hibarutin végrehajtása után a program folytatásának helyét. Parancsként kiadva a beépített hibakezelőt aktivizálja.

Lásd még az ON EXCEPTION GOTO és a CONTINUE utasításokat.

6. ERRNUM függvény

ERRNUM

A hibakezeléshez használható függvény. Aktuális értéke megegyezik az utoljára fellépett hiba kódjával. A használt hibakódok ugyanazok, mint az előző verziókban, néhány új hibakóddal kiegészítve.

A 128–255 hibakódokat operációs rendszer hibának tekintjük, ilyenkor a beépített hibakezelő az alábbi hibaüzenetet írja ki („n" a hibakód):

***** System error n**

A 128 alatti hibakódokat BASIC hibáknak tekintjük. A normál BASIC működése során ebből csak néhány fordul elő, ezekhez a kódokhoz saját hibaüzenet tartozik. A többi kód esetén kiírt hibaüzenet („n" a hibakód):

***** BASIC error n**

Az előző verziókban a 128 alatti, saját hibaüzenettel nem rendelkező hibakódokat ismeretlennek tekintettük és a „*** BASIC corrupted" üzenet került kiírásra. Ez az üzenet a 2.0 verzióban nem fordul elő.

A hibakódok és hibaüzenetek táblázata a függelékben található.

7. ERRLIN függvény

ERRLIN

A hibakezeléshez használható függvény. Aktuális értéke az a sorszám, amelyben az utolsó hiba előfordult.

8. HEX$ függvény

HEX$(x)

A hexadecimális számok használatát támogató függvény. A paraméterként szereplő „x" numerikus kifejezésnek a 0–65535 tartományba kell esnie. Eredményül négykarakteres stringet ad, amely az „x" érték hexadecimális megfelelője. A 9 fölötti hexa számjegyekre nagybetűket használ. HEX$(x) és „x" előjel nélküli 16 bites egész számoknak tekintendők.

9.DEC függvény

DEC(a$)

A hexadecimális számok használatát támogató függvény. Azt a decimális számot adja eredményül, amely az „a$" stringkifejezésben tárolt hexadecimális számnak felel meg. A 9 fölötti hexa számjegyek lehetnek kis- és nagybetűvel is írva. DEC(a$) és „a$" előjel nélküli 16 bites egész számoknak tekintendők.

10. CONTINUE utasítás

CONTINUE [sorszám]

Az ON EXCEPTION GOTO utasítással definiált hibarutin befejezésére szolgál. Hatására a programfutás a „sorszám"-mal adott soron, ha ez nincs megadva, akkor a hibát okozó utasítással, illetve az EXCEPTION utasításban kijelölt helyen folytatódik.

Az 2.0 előtti verziókban a CONTINUE utasítás programban hatástalan volt, csak a STOP-pal leállított program folytatására volt alkalmas parancs módban. A 2.0 verzióban a CONTINUE alkalmas a hiba miatt leállt program folytatására is. Ha a programot megváltoztatjuk, beleértve a program átsorszámozását is, a folytatás lehetetlenné válik.

11. Relatív koordináták a PLOT utasításban

A PLOT utasításban a relatív koordinátákat úgy adjuk meg, mint az abszolut koordinátákat, csak + vagy – jelet teszünk eléjük. Például a

PLOT +80,+0; +0,+80; –80,+0; +0,–80

utasítás az aktuális rajzolási pozícióban egy kis négyzetet fog rajzolni.

12. PLOT RECTANGLE utasítás

PLOT RECTANGLE (v,f [,d [,a [,r]]])

Téglalap és négyzet rajzolása. A paramétereket kerek zárójelek között kell felsorolni. Az első két paramétert mindig ki kell írni, a továbbiakat csak szükség esetén. A paraméterek jelentése a következő:

v, f: Oldalhosszak.
(„v" a vízszintes oldal, ha a szög 0)
d: A megrajzolandó oldalak száma. Ha nem adjuk meg, akkor a BASIC 4-nek tekinti.
a: Annak a szögnek az értéke radiánban, amellyel a „v" oldal hajlik az X tengelyhez. Ha nem adjuk meg, akkor a BASIC 0-nak tekinti.
r: A képgeometria miatt szükséges függőleges nagyítás, értéktartománya: 0 <= „r" < 2. Ha nem adjuk meg, a BASIC 6/5-nek tekinti, mert ennél az értéknél látszik a képernyőre rajzolt négyzet egyenlő oldalúnak.

A rajzolás az aktuális rajzpozícióban kezdődik, és a rajzoló sugár az utolsó kirajzolt ponton marad. Ha az összes oldalt kirajzoltatjuk, a sugár a kiindulópontba tér vissza. A rajzolás az aktuális rajzolási módban, és vonaltípussal történik.

Minden megrajzolt oldalnak rá kell férnie a képernyőre. Ha híváskor a rajzoló sugár kikapcsolt állapotban van, akkor csak pozicionálás történik, de a téglalapból bejárt oldalak ekkor sem nyúlhatnak túl a képernyőn.

13. PLOT POLYGON utasítás

PLOT POLYGON (h, n [,d [,a [,r]]])

Szabályos sokszög rajzolása. A paramétereket kerek zárójelek között kell felsorolni. Az első két paramétert mindig ki kell írni, a továbbiakat csak szükség esetén. A paraméterek jelentése a következő:

h: Az oldalak hossza.
n: A sokszög oldalainak száma, 0-tól 20-ig.
d: A megrajzolandó oldalak száma. Ha nem adjuk meg, a BASIC minden oldalt megrajzol (d=n).
a: Az elsőként rajzolandó oldalnak az X-tengellyel bezárt szöge radiánban. Ha nem adjuk meg, a BASIC 0-nak tekinti.
r: A képgeometria miatt szükséges függőleges nagyítás, értéktartománya: 0 <= „r" < 2. Ha nem adjuk meg, a BASIC 6/5-nek tekinti, mert ennél az értéknél látszik a képernyőre rajzolt sokszög szabályosnak, egyenlő oldalúnak.

A rajzolás az aktuális rajzpozícióban kezdődik, és a rajzoló sugár az utolsó kirajzolt ponton marad. Ha az összes oldalt kirajzoltatjuk, a sugár a kiindulópontba tér vissza. A rajzolás az aktuális rajzolási módban, és vonaltípussal történik.

Minden megrajzolt oldalnak rá kell férnie a képernyőre. Ha híváskor a rajzoló sugár kikapcsolt állapotban van, akkor csak pozicionálás történik, de a sokszögből bejárt oldalak ekkor sem nyúlhatnak túl a képernyőn.

14. PLOT ELLIPSE utasítás

PLOT ELLIPSE (v [,f [,s [,c [,a [,r]]]]])

Ellipszis és kör rajzolása. A paramétereket kerek zárójelek között kell felsorolni. Csak az első paramétert kell mindig kiírni, a továbbiakat csak szükség esetén. A paraméterek jelentése a következő:

v, f : A vízszintes és függőleges féltengely hossza, ha az X tengellyel bezárt szög nulla. Ha „f" nincs megadva, akkor a BASIC „v"-vel egyenlőnek veszi, ilyenkor fog kört rajzolni.

s, c : A megrajzolandó ív kezdő- („s") és végszöge („c") radiánban a pozitív X tengelyhez képest. Ha nem adjuk meg, akkor 0, vagyis kirajzolódik az egész ellipszis.

a : Az ellipszis elforgatási szöge radiánban az X tengelyhez képest. Ha nem adjuk meg, a BASIC nullának veszi.

r : A képgeometria miatt szükséges függőleges nagyítás, értéktartománya: $0 <= „r" < 2$. Ha nem adjuk meg, a BASIC 6/5-nek tekinti, mert ennél az értéknél látszik a képernyőre rajzolt kör valóban körnek.

Az ellipszis középpontja a rajzoló sugár aktuális pozíciója, ezt nem változtatja meg az utasítás. Nem feltétlenül kell az egész ellipszisnek fölférnie a képernyőre. Az ellipszis mindenképpen kirajzolódik, akkor is, ha a rajzoló sugár ki van kapcsolva.

Az ellipszisrajzoló rutin belsőleg korlátozott aritmetikai pontossággal dolgozik, ezért bizonyos ellipsziseket nem teljesen a várt módon rajzol ki. Általában a hosszú, vékony ellipszisek, amelyeket a 0, 45 fok, 90 fok, stb. szögek közelében, de nem pontosan ezeken a szögeken rajzol, kisebbek lehetnek a vártnál és szélsőséges esetben nem elliptikusak is lehetnek. A belső aritmetikai pontosságot csak a rajzolási sebesség csökkentésével lehetett volna javítani, ehelyett inkább a gyorsabb rajzolást választottuk.

Kapcsolódási pontok a BASIC 2.0-hoz

PRNDEF és SERDEF kapcsolódási pontok

A nyomtató eszközmeghajtó a karakterkiírás előtt meghívja a RAM-ban levő **PRNDEF** szubrutint. Ez a rutin alaphelyzetben csak egy RET utasításból áll. Ha a felhasználó a nyomtatóra küldött karaktereket át akarja kódolni, akkor ide írhatja be az átkódoló rutinra történő ugrást. Az átkódolásra például akkor lehet szükség, ha a nyomtató nem tud ékezetes magyar betűket nyomtatni, vagy tud ugyan, de nem a TV—Computerben használt kódokon.

A PRNDEF rutin meghívásakor a C regiszterben a nyomtatandó karakter, a B regiszterben pedig 255 (0FFh) van. A PRNDEF rutinból visszatérve a C-ben levő kód kinyomtatódik, a B-ben levő pedig csak akkor, ha nem 0FFh. Így a karaktereket konvertálni lehet másik karakterré, vagy karakterpárrá.

A soros vonali eszközmeghajtó egy ugyanilyen szubrutint hív, melynek neve **SERDEF**. Így a soros nyomtatókra is van konverziós lehetőség. A függelékben mindkét kapcsolódási pont használatára közlünk programlistát.

STRCMP kapcsolódási pont

A string összehasonlítások közben, minden karakterpár összehasonlítása után a BASIC meghív egy **STRCMP** nevü, RAM-ban levő szubrutint. Alaphelyzetben ez a szubrutin csak egy RET utasításból áll, de ezt helyettesíthetjük egy saját rutinra történő ugrással, ezáltal felülbírálhatjuk az összehasonlítás eredményét. Például erre olyankor lehet szükség, ha a BASIC programban gyakran kell a magyar ábécé szerint szavakat vagy neveket sorbarendezni.

Az STRCMP-be belépéskor az egyik karakter az A regiszterben, a másik (HL)-ben van, így a CP (HL) utasítás a normál ASCII karakterekre jól állítja be a jelzőket.

A függelékben programlistát közlünk példaként a kapcsolódási pont használatára.

BASEXT kapcsolódási pont

A 2.0 BASIC verzió, ha fel nem ismert utasítást észlel, meghívja a RAM-ban levő **BASEXT** szubrutint. Ez alaphelyzetben egy RET utasításból áll. Ha a felhasználó egy új BASIC utasítást elvégző rutint akar használni, akkor a BASEXT címre az új rutinra történő ugrást kell beírnia.

Ha az utasítást a felhasználó rutinja sem ismerte fel, vissza kell térnie egy RET utasítással és a belépő regiszterértékeket kell visszaadnia. Ha az utasítás feldolgozása sikeres volt, visszatérés előtt a stack-mutatót 2-vel növelni kell (pl. POP HL), és a HL'-nek a feldolgozott rész utáni első karakterre kell mutatnia (a kettőspont, a felkiáltójel vagy a sorvégjel „token"-ra).

Az új rutin a feldolgozásban felhasználhatja a BASIC program néhány belső szubrutinját is. Az új BASIC utasítások elkészítéséhez az itt következő fejezet ad részletes útmutatást. A függelékben közlünk egy programlistát, amely példaként a soros vonal paramétereit beállító BAUD utasítást kapcsolja hozzá a BASIC-hez.

A BASIC bővítése új utasításokkal

A BASIC 2.0 lehetőséget ad az utasításkészlet bővítésére a BASEXT kapcsolódási ponton keresztül. Ezeknek az új utasításoknak azonos módon kell kezelniük a processzor regisztereit a beépített BASIC értelmezővel. A fix jelentésű regiszterek:

IX – a BASIC változóterület kezdetére mutat.
IY – a BASIC stack utoljára használt bájtjára mutat. (A BASIC stack nem ugyanaz, mint a processzor által használt rendszer stack.)
HL' – a BASIC program következő szintaktikai egysérére mutat.
B' – a BASIC program aktuális szintaktikai egységének egybájtos kódját tartalmazza (token).

Általában ezeknek a regisztereknek a tartalmát nem szükséges konkrétan ismerni, különösen veszélyes őket önkényesen módosítani. Tartalmaz a BASIC olyan beépített rutinokat (lásd alább), amelyek aktualizálják, karbantartják ezeket az értékeket. Például ilyen a GET beépített rutin.

Tegyük fel, hogy egy új utasítást végrehajtó rutint hozzákapcsolunk a BASIC értelmezőhöz. Ha az megkapja a vezérlést, először azt kell ellenőriznie, hogy számára ismert, vagy ismeretlen parancs vagy utasítás következik-e. Belépéskor HL' mutat az utasítás első bájtjára. Ha úgy találjuk, hogy az egy ismeretlen utasítás, akkor HL' értékét vissza kell állítani, és azon a kódsorozaton kell a működést befejezni, amit a BASEXT-ből a BASIC-be láncolódáskor elmentettünk.

Ha az utasítást felismertük, akkor az ehhez tartozó paramétereket ki kell értékelnünk, és az utasítást ennek megfelelően végre kell hajtani. Ha készen vagyunk, akkor a rendszerstack-ból az első visszatérési címet ki kell venni (pl. egy „POP HL" utasítással), és RET utasítással kell befejezni a működést.

Természetesen az utasítás, és paramétereinek feldolgozása során HL' és B' értékét aktualizálni kell. A feldolgozás akkor korrekt, ha visszatéréskor HL' a kettőspont, a felkiáltójel, vagy a sorvégjel tokenjére mutat. Ha ez nem teljesül, akkor a BASIC a „Not understood" hibát fogja visszatérés után generálni.

A BASIC bővítések számára különösen hasznos, hogy szabványos rutinhívással meg lehet hívni belső BASIC rutinokat, például a numerikus és stringkifejezések kiértékelésére. Sok egyéb rutin is hívható a BASIC számára fenntartott restart vektoron keresztül (lásd alább). Más restart vektorok is hívhatók, ezeket most ismertetjük:

Hibageneráló restart vektor

RST 8

A BASIC hibát generál. A hibakódnak, mint egy operandusnak, közvetlenül az „RST 8" utasítás kódja után kell állnia. A hibakódok összefoglalását a függelékben közöljük. Ha a BASIC-ben definiáltunk hibakezelő rutint, akkor arra adódik át a vezérlés, emellett a nyomkövetési funkciók is működnek. Ha nincs definiált hibakezelő rutin, akkor a megfelelő üzenet fog kiíródni. Az „RST 8" végrehajtása után nem lesz visszatérés a hívó programba. Ez a rutin hívható a korábbi verziókban is.

Hibaellenőrző restart vektor

RST 10h

Hibaellenőrző rutin az operációs rendszer hívások számára. Közvetlenül az operációs rendszer hívása után meghívva (RST 30h és rutinkód után) vagy visszatér a hívóhoz, ha a jelzőbitek közül Z=1, vagy belép a basic hibageneráló rutinba (RST 8). Az A regiszterben várja a hibakódot. Ez a rutin is hívható a korábbi változatokban is.

Restart vektor a BASIC rutinokhoz

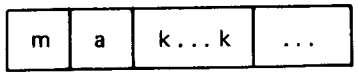
RST 18h

Végrehajt egy beépített BASIC rutint. Az „RST 18h" kódja után kell felsorolni operandusként a BASIC rutinok kódjait, a végrehajtás ennek a felsorolásnak a sorrendjében fog történni. A listában utolsó rutinkód legfelső bitjét 1-be kell állítani, ez jelzi a lista végét, a program pedig a soronkövetkező bájton fog folytatódni.

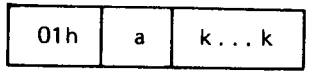
BASIC adatábrázolás

A beépített BASIC rutinok a BASIC stack-ben dolgoznak, esetenként használják a lebegőpontos regisztereket is (X, Y, V vagy W), illetve a szimbólumtábla elemeit. A szimbólumtábla és a BASIC stack felépítését a „BASIC Programozási Segédlet" tartalmazza, azonban itt is szükségesnek látjuk a fontosabb elemeket ismertetni.

String a szimbólumtábla adatmezőjében:

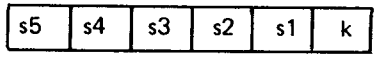


String a BASIC stack-ben:

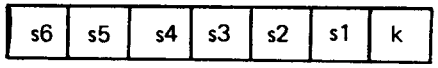


m — karakterek maximális száma
a — karakterek aktuális száma
k . . . k — karakterek

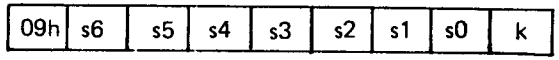
Szám a szimbólumtábla adatmezőjében:



Szám az aritmetikai regiszterben:



Szám a BASIC stack-ben:



s0 — Két számjegy a túlcsorduláshoz.
s1–s5 — Tíz számjegy az ábrázoláshoz. A számokat normalizálva ábrázolja a BASIC, a tizedespont „s1" két BCD– jegye között van. A BCD számjegyek csökkenő sorrendje: „s1" felső, „s1" alsó, „s2" felső, „s2" alsó, . . . , „s5" felső, „s5" alsó BCD jegye.
s6 — Két számjegy a számolási pontosság növelésére.
k — Előjel és kitevő.
b0–b6 — tizes kitevő, 40h-val eltolva.
b7 — a szám előjele: 0=plusz, 1=minusz.

BASIC rutinok ismertetése

Alább ismertetjük a BASIC rutinokat. Közülük csak az első 15 hívható a korábbi változatokban, a többi új a 2.0 változatban. Ha egy rutin számokat vagy stringeket használ a BASIC stack-ben, nem ellenőrzi, hogy valóban

megfelelő típusú-e az ott található adat; ez az ellenőrzés a rutint hívó program dolga. A processzor regiszterei közül IX, IY, HL', DE' és B' értékeit megőrzik a rutinok, kivéve ha külön jelezzük a rutinok leírásánál, hogy a regiszter értéke megváltozik. A többi regiszter általában elromlik, kivéve ha külön jelezzük, hogy értéke megőrződik.

FADD – 0

Lebegőpontos összeadás. A BASIC stack tetején levő két számot összeadja, és helyettesíti őket az eredménnyel. AF regiszterpár értéke nem változik.

FDIV – 1

Lebegőpontos osztás. A BASIC stack tetején levő számmal elosztja a második számot, és az eredménnyel helyettesíti őket. AF regiszterpár értéke az átvitelbit (carry) kivételével nem változik.

FMULT – 2

Lebegőpontos szorzás. A BASIC stack tetején levő két számot összeszorozza, és helyettesíti őket az eredménnyel. AF regiszterpár értéke nem változik.

FSUB – 3

Lebegőpontos kivonás. A BASIC stack tetején levő számot kivonja a másodikból, az eredménnyel helyettesíti őket. AF regiszterpár értéke nem változik.

FNEG – 4

Lebegőpontos előjelváltás. A stack tetején levő számnak ellentétesre változtatja az előjelét. Nem változik BC, DE és HL regiszterpárok értéke.

CPUSH – 5

Belső műveletek számára konstansot helyez a BASIC stack tetejére. Az ezt követő kód jelenti a konstans sorszámát (0–39).

XPUSH – 6

Az X lebegőpontos regiszter tartalmát átmásolja a BASIC stack tetejére.

YPUSH – 7

Az Y lebegőpontos regiszter tartalmát átmásolja a BASIC stack tetejére.

XASSIGN – 8

A BASIC stack tetején levő számot átmásolja az X lebegőpontos regiszterbe. Az átmásolt érték a BASIC stack tetején is megmarad.

YASSIGN – 9

A BASIC stack tetején levő számot átmásolja az Y lebegőpontos regiszterbe. Az átmásolt érték a BASIC stack tetején is megmarad.

XPOP – 10 (0Ah)

A BASIC stack tetején levő számot kiveszi a stack-ből és átírja az X lebegőpontos regiszterbe.

YPOP – 11 (0Bh)

A BASIC stack tetején levő számot kiveszi a stack-ből és átírja az Y lebegőpontos regiszterbe.

FDUP – 12 (0Ch)

A BASIC stack tetején levő számot megkettőzi, azaz mégegyszer ráteszi a stack tetejére.

FPOP – 13 (0Dh)

A BASIC stack tetején levő számot kiveszi a stack-ből és átírja abba a 6 bájtos BASIC változóba, amelynek adatmezőjére a HL regiszterpár mutat.

PNUM – 14 (0Eh)

Kielemez egy zárójelben levő numerikus kifejezést, majd kiértékeli, és az eredményt a BASIC stack tetejére teszi. HL' és B' regisztereket az elemzésnek megfelelően módosítja.

A további rutinok csak a BASIC 2.0 változattól kezdve hívhatók!

FDROP – 15 (0Fh)

A BASIC stack-ből kiveszi a tetején levő számot, az értékét nem őrzi meg.

VPUSH – 16 (10h)
A V lebegőpontos regiszter tartalmát átmásolja a BASIC stack tetejére.

WPUSH – 17 (11h)
A W lebegőpontos regiszter tartalmát átmásolja a BASIC stack tetejére.

VASSIGN – 18 (12h)
A BASIC stack tetején levő számot átmásolja a V lebegőpontos regiszterbe. Az átmásolt érték a BASIC stack tetején is megmarad.

WASSIGN – 19 (13h)
A BASIC stack tetején levő számot átmásolja a W lebegőpontos regiszterbe. Az átmásolt érték a BASIC stack tetején is megmarad.

VPOP – 20 (14h)
A BASIC stack tetején levő számot kiveszi a stack-ből és átírja a V lebegőpontos regiszterbe.

WPOP – 21 (15h)
A basic stack tetején levő számot kiveszi a stack-ből és átírja a W lebegőpontos regiszterbe.

SIN – 22 (16h)
Szinuszfüggvény. A BASIC stack tetején levő szám helyébe ennek a számnak a szinuszát teszi.

COS – 23 (17h)
Koszinuszfüggvény. A BASIC stack tetején levő szám helyébe ennek a számnak a koszinuszát teszi.

ATN – 24 (18h)
Arkusztangens függvény. A BASIC stack tetején levő szám helyébe ennek a számnak az arkusz tangensét teszi.

TAN – 25 (19h)
Tangensfüggvény. A BASIC stack tetején levő szám helyébe ennek a számnak a tangensét teszi.

EXP – 26 (1Ah)
Exponenciális függvény. A BASIC stack tetején levő számmal, mint kitevővel hatványozza a természetes logaritmus alapszámát (e=2,71 . . .), az eredményt a kitevőnek használt szám helyére teszi.

INT – 27 (1Bh)
Egészrész képzése. A BASIC stack tetején levő szám helyébe azt a legnagyobb egészet teszi, ami még nem nagyobb az eredeti számnál.

LOG – 28 (1Ch)
Természetes logaritmus. A BASIC stack tetején levő szám helyébe ennek a számnak a természetes logaritmusát teszi.

SQR – 29 (1Dh)
Négyzetgyökvonás. A BASIC stack tetején levő szám helyébe ennek a számnak a pozitív négyzetgyökét teszi.

INVOL – 30 (1Eh)
Hatványozás. A BASIC stack tetején levő számmal, mint kitevővel hatványozza a második számot, az eredménnyel helyettesíti őket.

FCMP – 31 (1Fh)
Numerikus összehasonlítás. A BASIC stack tetején levő két számot összehasonlítja, és kiveszi mindkettőt a stack-ből. Az eredményt a jelzőbitek tartalmazzák (az A regiszter értéke megfelel a jelzőbitek állásának):

M=1, ha az első szám volt a nagyobb,
Z=1, ha a két szám egyenlő volt, és
P=1, (M=0 és Z=0) ha az első szám volt a kisebb.

FCPL – 32 (20h)
Tizes komplemens. A BASIC stack tetején levő szám helyébe ennek a számnak a tízes komplemensét teszi. (Kivonja 0-ból a mantissza számjegyeket.

GETDIG – 33 (21h)

A BASIC stack tetején levő számból visszaad egy decimális számjegyet. A szám változatlanul a stack tetején marad. A számjegy sorszámát a B regiszterben várja. Az egyesek helyén álló (egyes helyiértékű) számjegy a 4-es sorszámú, a sorszám növelésével a helyiérték csökken. A számjegy értékét az A regiszterben kapjuk vissza, mig a BC, DE és HL regiszterpárok értéke nem változik.

PUTDIG – 34 (22h)

A BASIC stack tetején levő számba beleír egy decimális számjegyet, a szám ezzel a módosítással a stack tetején marad. A számjegy sorszámát a B regiszterben várja, ugyanúgy, ahogy az GETDIG-nél már leírtuk. Az új számjegyet az A regiszterben várja: értéke 0tól 9-ig terjedhet. BC, DE és HL regiszterpárok értéke nem változik.

FNORM – 35 (23h)

Normál alak. A basic stack tetején levő számot normál alakra hozza, azaz számjegyenkénti balra, vagy jobbra igazítással úgy alakítja, hogy a legnagyobb helyiértékű számjegy a 4-es sorszámú helyre kerüljön (lásd GETDIG és PUTDIG rutinokat). A 2-es és 3-as sorszámú számjegyek is szerepet kapnak a műveletnél.

FZERO – 36 (24h)

Zérusvizsgálat. Megvizsgálja, hogy a BASIC stack tetején levő szám minden számjegye nulla-e, és ha igen, akkor a kitevőt tartalmazó bájtot is nullázza, így áll elő a nulla szám belső ábrázolású alakja. A Z jelzőbit értéke 1 lesz, ha a szám nulla volt.

FOUT – 37 (25h)

A BASIC stack tetején levő számot átalakítja a szám nyomtatható, ASCII karakterekből álló megfelelőjére. A számot változatlanul a stack tetején hagyja. Visszatéréskor HL regiszterpár mutat arra a belső BASIC pufferre, ahol az átalakított szám található.

PUSH – 38 (26h)

A HL regiszterpárban levő 16 bites előjeles egész számot lebegőpontos számmá alakítja, és a BASIC stack tetejére teszi.

POP – 39 (27h)

A BASIC stack tetején levő számot átalakítja kétbájtos előjeles egésszé, és a HL regiszterpárba teszi. A számot elveszi a stack tetejéről.

$PUSH – 40 (28h)

A HL regiszterpárral, mint mutatóval kijelölt stringet a BASIC stack tetejére teszi. Az első bájtot figyelmen kívül hagyja – a BASIC ábrázolásban ez a string maximális hossza – a második bájtot tekinti a string aktuális hosszának.

$POP – 41 (29h)

A BASIC stack tetején levő stringet a HL regiszterpárral, mint mutatóval kijelölt helyre teszi. HL címen az átmásolható string maximális hosszának kell lenni.

$CMP – 42 (2Ah)

Stringek összehasonlítása. Összehasonlítja a BASIC stack tetején levő két stringet. Mindkét stringet elveszi a stack tetejéről és az eredményt az AF regiszterpárba teszi ugyanúgy, mint ahogy az FCMP rutinnál (31-es rutin) leírtuk. Minden karakter- összehasonlításnál meghívjaaz STRCMP kapcsolódási ponton levő rutint.

FEXPR – 43 (2Bh)

Kielemez és kiértékel egy numerikus kifejezést, az eredményt a BASIC stack tetejére teszi. HL' és B' regiszterek értékét a kiértékelésnek megfelelően módosítja.

$EXPR – 44 (2Ch)

Kielemez és kiértékel egy stringkifejezést, az eredményt a BASIC stack tetejére teszi. HL' és B' regiszterek értékét a kiértékelésnek megfelelően módosítja.

GET – 45 (2Dh)

BASIC program elemzése. A BASIC program következő szintaktikai egységére HL' regiszterpár mutat. Ez alapján a rutin beállítja B' és DE' regisztereket, míg az AF regiszterpárba különféle jelzőbitek kerülnek. HL' mutatót a következő újabb szintaktikai egységre lépteti.

Ha a szintaktikai egység egy BASIC kulcsszó tömörített alakja, azaz token, akkor ez kerül az A regiszterbe (és ugyanez kerül B'-be is.). Visszatérés előtt a token értékét a kettőspont tokenjével a rutin összehasonlítja. Az átvitel (carry) jelzőbit nulla értéke jelzi az utasítás végét, ez lehet az utasításelválasztó kettőspont tokenje, a megjegyzést bevezető felkiáltójel tokenje, vagy a sorvégjel. Kettőspont token esetén a zérus jelzőbit értéke 1 lesz. A tokenek táblázatát a függelékben közöljük.

Ha a szintaktikai egység nem token, akkor az A regiszter (és B' is) az alábbi jelzőbiteket tartalmazza:

bit 0 – Értéke 1, ha numerikus vagy stringazonosítóról van szó, míg 0, ha szám- vagy stringkonstansról.
bit 1 – Értéke szám esetén 1, string esetén 0.

Ha konstansról van szó (bit 0 =0), akkor ez a konstans a BASIC stack tetejére kerül. Ebben a formában még általában nem használható,mert ez lehet egy kifejezés első tagja is, ezért az EXPR vagy $EXPR rutinok egyikét még meg kell hívni.

Azonosító esetén (bit 0 =1) a DE' regiszterpár a megfelelő szimbólumtábla– bejegyzés első adatbájtjára mutat, a C' regiszterbe pedig ennek a bejegyzésnek a típusbájtja kerül.

LIST – 46 (2Eh)

Azt a tömörített (tokeneket tartalmazó) stringet, amelyre a HL regiszterpár mutat, kifejti és ki is írja. A string végét 255 (0FFh) jelzi, visszatéréskor HL a végjelet követő bájtra mutat. Megjegyezzük, hogy ez az a formátum, amelyben a BASIC a programsorokat tárolja, ezért egy programsort (sorszám nélkül) úgy listázhatunk ki, hogy a HL mutatót a sorszámmező után állítjuk, majd meghívjuk a LIST rutint.

Az operációs rendszer bővítései

Hideg és meleg reset

A 2.0 előtti változatban a reset kétféle lehetett: hideg reset és meleg reset. A hideg reset teljesen alaphelyzetbe állította az operációs rendszert és az eszközmeghajtó programokat, és újraindította a programmodulként a gépben levő programot, vagy ennek hiányában a BASIC-et. A meleg reset ugyancsak alaphelyzetbe állította az operációs rendszert és az eszközmeghajtókat, majd úgy indította el a programmodult vagy a BASIC-et, hogy nem volt szükség az összes belső változó és munkaterület alaphelyzetbe állítására. Például a gépben levő BASIC program meleg reset után újraindítható volt.

Meleg reset programból

A 2.0 változatban a hideg reset hatása változatlan, tehát alaphelyzetbe áll az operációs rendszer, és teljesen alaphelyzetben indítja el a programmodult vagy a BASIC-et. A meleg reset azonban további lehetőséggel bővült. Ha a TV–Computer bekapcsolt állapotban van, akkor az operációs rendszer kétféleképpen kaphat meleg resetet:

– normál módon a reset gomb egyszeri megnyomásával,
– BASIC-ből vagy programmodulból fix memóriacímre való ugrással (RESET cím).

CENBLE változó

A CENBLE változóval engedélyezhető vagy letiltható, hogy a BASIC helyett a programmodul induljon el. Ezt a változót az operációs rendszer a meleg reset során nem változtatja. Alapesetben értéke 0, ami a hagyományos viselkedést eredményezi, tehát ha van programmodul, akkor az indul el, egyébként pedig a BASIC. Ha a CENBLE értéke nullától különböző, akkor meleg reset esetén mindig a BASIC indul el, tehát a programmodul le van tiltva.

A fenti módosításokkal válik lehetővé, hogy programmodulból beállítva a CENBLE változót, majd meleg resetet adva az operációs rendszernek áttérjünk a BASIC-be. A programmodul csak akkor fog újraindulni, ha hideg reset történik, illetve a CENBLE változót BASIC-ből töröljük és meleg resetet adunk az operációs rendszernek.

Reset a programmodulnak és a BASIC-nek

A 2.0 előtti változatokban a WARM_FLAG változó értéke jelezte, hogy hideg vagy meleg reset zajlott. Ezt az értéket az operációs rendszer, miután végrehajtotta saját resetjét, átadta a programmodulnak vagy a BASIC-nek. Ez utóbbiak is ennek megfelelően hajtották végre saját meleg vagy hideg resetjüket, végül törölték WARM_FLAG-et.

RESTART változó

A 2.0 változatban a RESTART változót használjuk arra, hogy az alkalmazói programokat akár hideg, akár meleg resettel el lehessen indítani. Az operációs rendszer meleg reset végén a RESTART változó értéke átíródik WARM_FLAG-be. Általában RESTART értéke nem nulla, tehát a programmodul vagy a BASIC szintén meleg resetet hajt végre. Ha azonban azt akarjuk, hogy az operációs rendszer meleg resetet hajtson végre, de a programmodul vagy a BASIC hideg resetet, akkor törölni kell a RESTART változót.

MOPS WARM változó

A programmodulnak vagy a BASIC-nek szüksége lehet arra, hogy tudja, az operációs rendszer meleg vagy hideg resetet hajtott-e végre. Korábban ezt a WARM_FLAG jelezte, most azonban külön meg kell őrizni az operációs rendszer reset alatti WARM_FLAG értéket. Erre a célra a MOPS_WARM változót hoztuk létre. A WARM_FLAG értéke tehát az operációs rendszer reset végén átíródik MOPS_WARM-be, ezután a programmodul és a BASIC számára a RESTART értéke átíródik a WARM_FLAG-be.

Nemcsak az alkalmazói programoknak (BASIC és programmodul) lehet szükségük arra, hogy tudják, milyen resetet hajtott végre az operációs rendszer, hanem az operációs rendszer egyes meghajtóinak és a bővítőkártyáknak is előre tudniuk kell, hogy az alkalmazói program milyen resetet fog végrehajtani. Ehhez elég a RESTART változót megvizsgálni.

Például a beépített billentyűzetkezelő, amely lehetővé teszi feladatbillentyűk használatát, az operációs rendszer hideg resetje esetén helyet foglal magának a memóriából; egyébként pedig nem. Az operációs rendszer meleg resetje alatt csak akkor kell törölni a feladatbillentyűk definícióit, ha az alkalmazói program nem meleg, hanem hideg resettel fog indulni.

Hasonlóképpen egy bővítőkártyának is szüksége lehet memória lefoglalásra, ha az operációs rendszer hideg resetet hajt végre, és meleg reset esetén csak akkor kell törölnie a lefoglalt területet, ha az alkalmazói program hideg resettel fog indulni.

Áttérés programmodulból BASIC-be

A CENBLE és RESTART változók, valamint a RESET ugrási cím lehetővé teszik, hogy programmodulból elindítható legyen a BASIC, illetve BASIC-ből elindítható legyen a programmodul a TVC RESET billentyűjének használata nélkül.

Van azonban egyszerűbb, és a korábbi változatokban is működő módja a programmodulból BASIC-be való visszatérésnek. Ezt használja például a VT–DOS programmodul, ha nincs a gépben VT–DOS diszkcsatoló. Letiltott IT mellett a 0. memórialapra a RENDSZER ROM-ot (SYSROM) kell kapcsolni, és a DE–22 címre kell ugrani (DE regiszterpárban 0. memórialapra mutató címet kap a programmodul.).

Ez ugyan gyors átlépést biztosít a BASIC-be, de csak a programmodulba való belépés elején ajánlott használni. A BASIC ugyanis feltételezi, hogy az operációs rendszer közvetlenül reset utáni állapotban van, és később ez már nem feltétlenül teljesül. Természetesen ezután, ha meleg reset fordul elő, akkor a CENBLE változó határozza meg, hogy elindulhat-e a programmodul.

Ha tetszőleges egyéb helyen van szükség a programmodulból a BASIC-be átlépni, akkor a CENBLE-t kell beállítani és a RESET címre kell ugrani. Ennek eredményeként a RESTART változó törlődni fog, az operációs rendszer meleg resetet hajt végre, és a CENBLE beállítás miatt elindul a BASIC.

BASIC program betöltése ROM-ból

Ha az alkalmazói programok számára hideg resetet állítunk be a RESTART változó törlésével, akkor akár programmodul ROM-ból, akár bővítőkártya ROM-ból átmásolható egy normál BASIC program a memóriába, és elindítva a BASIC-et a program végre is hajtható. A programot pontosan abban a formában kell átmásolni, ahogy ezt a BASIC-ben SAVE paranccsal kimentettük.

A BASIC program fejblokkjában található 16 bájtot és azt az extra 128 bájtot, ami a floppy lemezen a .CAS végződésű programok elején található, nem kell a BASIC területre másolni. Az így elhagyott részek csak a betöltő programok számára tartalmaznak információt. A betöltés kötelezően a normál BASIC kezdőcímre, 6639-re (19EFh) kell, hogy történjen.

A függelékben két példaprogramot közlünk a ROM-ból való programbetöltéshez.

BASFLG változó

A 2.0 változatban a BASFLG változóból tudja meg a BASIC, hogy a memóriában van-e végrehajtható program. A korábbi változatokban egyszerűen figyelmen kívül hagyja a BASIC a ROM-ból áttöltött programot. A BASFLG változó négy jelzőbitet tartalmaz:

bit 0 – Ha 1-be állítjuk, akkor a BASIC indulásakor a V-alakban kiírt „VIDEOTON" kezdőkép nem jelenik meg.

bit 1 – Ha 1-be állítjuk, akkor a „COPYRIGHT" feliratot és a szabad bájtok számát nem írja ki a BASIC.

bit 2 – Ha 1-be állítjuk, akkor a BASIC nem hajtja végre induláskor a „NEW" parancsot, tehát a memóriába előzőleg betöltött BASIC programunk működőképes marad.

bit 3 – Ha 1-be állítjuk, akkor a BASIC indulásakor egy automatikus „RUN" parancsot hajt végre. Csak bit 2-vel együtt van értelme használni.

A BASFLG változóban ajánlatos az összes bitet együtt beállítani, nem elegendő csak a kiválasztott egyetlen bitet változtatni.

BLINK változó

A 2.0 előtti változatokban a billentyűzet input nem adott villogó kurzort. Ennek vezérlését itt a BLINK változó végzi. Ha billentyűzet input híváskor BLINK értéke nem nulla, akkor van villogó kurzor, egyébként nincsen. BLINK értékét a billentyűzet input változatlanul hagyja, az editor input azonban törli.

BLINK beállításával például az INKEY$ és GET utasítások ideje alatt villogó kurzort kapunk.

Programmodul belépési pont

A RENDSZER ROM-ban a 49933 címen (0C30Dh) egy átkapcsolás található a programmodulba:

```
C20D: 3E 20     LD   A,20h
C30F: D3 02     OUT  (2),A
C311:
```

Ez a programrészlet SYSROM/RAM/RAM/CART memóriaállapotot lapoz be, és a programfutás a programmodul 49937 címén (0C311h) folytatódik. Ezt minden korábbi és későbbi BASIC változatban a programmodul rögzített belépési pontjának tekintjük, a címe ugyanez marad. Ezt használja a VIDEOTON által készített Debugger nyomkövető programmodul az indításhoz.

Új beépített operációs rendszer rutinok

A sokszögrajzolás, az ellipszisrajzolás és a feladatbillentyűk definiálása számára új rutinokkal bővítettük az operációs rendszert. A „TV–Computer Operációs Rendszer" című könyv ismerteti a korábbi változatokba beépített, és az RST 30h utasítással hívható operációs rendszer rutinokat. Itt csak a bővítés rutinok ismertetése található.

A video meghajtó kibővítése a sokszögrajzolással két kisebb problémát hozott felszínre. A 2.0 előtti változatokban egy vonal rajzolásakor a vonal kezdő és végpontját is megjelenítettük. Így két összeérő vonal rajzolásakor (pl. sokszögnél) a találkozási pontot kétszer rajzoltuk ki, ami általában nem baj, de az „XOR" rajzolási módban ez a találkozási pont rossz színnel (a tinta színe helyett a papír színével) rajzolódik ki.

Ezt a problémát a 2.0 verzióban úgy küszöböltük ki, hogy a vonalak első képpontját nem jelenítjük meg. Mivel egy vonal rajzolásához előzőleg be kell kapcsolni a rajzoló sugarat, és a sugár bekapcsolása megjeleníti az első pontot, a vonalak rajzolásában ez nem eredményez változást, viszont lehetővé teszi a sokszögek rajzolását „XOR" módban.

@POLY – Sokszögrajzolás.

Kód : 13 (0Dh)
Input : DE = az input paramétertáblára mutat.
Output : A = hibakód, illetve 0, ha sikeres volt a művelet.

Sokszöget rajzol az input paraméterek szerint. Minden megrajzolt oldalnak rá kell férnie a képernyőre. Ha híváskor a rajzoló sugár kikapcsolt állapotban van, akkor csak pozicionálás történik, de a sokszög kijelölt oldalai ekkor sem nyúlhatnak túl a képernyőn.

Az input paramétertábla felépítése:

1 bájt – A képgeometria miatt szükséges függőleges nagyítás, értéke: 0–7Fh. A 40h felel meg az 1:1 aránynak.
1 bájt – A sokszög oldalainak száma: 0–20.
1 bájt – A megrajzolandó oldalak száma: 0–20.

2 bájt – X irányú előjeles eltolás a rajzoló sugár aktuális pozíciójától az első oldal végéig.
2 bájt – Y irányú előjeles eltolás a rajzoló sugár aktuális pozíciójától az első oldal végéig.

@ ELLIP – Ellipszis ív rajzolás

kód : 14 (0Eh)
Input : DE = a bemenő paraméter táblázatra mutat.
Output : A= hibakód, illetve 0, ha nem volt hiba.

Ellipszis ívet rajzol a bemenő paraméterek szerint. A rajzoló sugár pozíciójában lesz az ellipszis középpontja, ezt a rutin nem módosítja, ugyancsak nem módosítja a rajzoló sugár bekapcsolt vagy kikapcsolt állapotát. Nem szükséges a kijelölt ív minden pontjának a képernyőre esnie. A kirajzolás akkor is megtörténik, ha a rajzoló sugár kikapcsolt állapotban van.

A bemenő paraméter táblázat felépítése:

2 bájt – SQR(R) ..... Q
2 bájt – (–P)/Q
4 bájt – (f↑2)*(SIN(a)↑2)+(V↑2)*(COS(a)↑2) ..... R
4 bájt – (f↑2)*(COS(a)↑2)+(V↑2)*(SIN(a)↑2)
4 bájt – SIN(a)*COS(a)*((f↑2)–(V↑2)) ..... P
1 bájt – 127*COS(s)
1 bájt – 127*SIN(s)
1 bájt – 127*COS(c)
1 bájt – 127*SIN(c)
1 bájt – A képgeometria miatt szükséges függőleges nagyítás, értéke: 0–7Fh. A 40h felel meg az 1:1 aránynak.

Jelölések:
a – az ellipszis X tengellyel bezárt szöge radiánban
v – a vízszintes féltengely hossza (ha a=0)
f – a függőleges féltengely hossza (ha a=0)
s – a megrajzolt ív kezdetének szöge radiánban
c – a megrajzolt ív végének szöge radiánban

@FKEY — Feladatbillentyű stringjének definiálása.

Kód : 20 (14h)
Input : DE = a feladatbillentyű új stringjére mutat
C = feladatbillentyű száma: 0—9
Output : A = hibakód, illetve 0, ha sikeres volt a definiálás
DE = a feladatbillentyű aktuális stringjére mutat

A C regiszterben specifikált feladatbillentyűhöz a DE-ben megadott kezdőcímű stringet rendeli hozzá. A string első bájtja a string hosszát kell, hogy megadja. Visszatéréskor DE a definiált új stringre mutat, ha A=0, vagy a régi stringre, ha a definiálás nem sikerül.

A feladatbillentyűk stringjei számára összesen 100 bájt áll rendelkezésre, beleértve ebbe a stringek hosszát jelentő bájtot is. Ha az aktuális stringet akarjuk lekérdezni, akkor például híváskor egy 255 bájt hosszú stringet adunk meg, ilyenkor visszatéréskor DE a jelenleg érvényes stringre mutat.

Az új rendszerváltozók elhelyezkedése a memóriában

A 2.0 verzió nem változtatta meg a publikált és nempublikált változók és adatterületek memóriacímeit. Néhány új változóra és adatterületre volt szükség, ezek a korábban nem használt helyekre kerültek.

A feladatbillentyűkhöz rendelhető stringek számára összesen 100 bájtot tartunk fenn. Ezt a TVC hidegindításakor (COLD RESET) foglalja le a kezelőprogram a használható memória végéről, és ennyivel csökkenti a HI_MEM változó értékét.

A további használatbavett memóriaterületek:

1792 — 1828 (700h — 724h)
Különféle új BASIC változók, amelyek az új hibakezelő utasításokhoz, valamint a RENUMBER és auto utasításokhoz szükségesek.

1829 PRNDEF (725h)
Kapcsolódási pont a nyomtató rutinhoz. Ha szükségünk van karakter konverzióra — pl. az ékezetes betűk helyett két rövid magánhangzót szeretnénk nyomtatni —, akkor az új rutinra mutató ugró utasítást ide kell elhelyezni.

1832 SERDEF (728h)
Kapcsolódási pont a soros vonali átvitelhez. úgyanúgy használható pl. a soros nyomtatóhoz, mint azt a PRNDEF-nél írtuk.

1835 RESET (72Bh)
Erre a belépési pontra ugorva újraindíthatjuk a ROM-kazetta programját, vagy a BASIC-et, a CENBLE változó értékétől függően. Ez nem kapcsolódási pont, mint az előző kettő, ezért nincs védve a végtelen ciklusban történő hívás ellen.

1838 STRCMP (72Eh)
Kapcsolódási pont a BASIC string-összehasonlításához. Minden karakterpár összehasonlításánál meghívja a BASIC. Ide kell elhelyeznünk az általunk írt összehasonlító rutinra mutató ugrást.

1841 BASEXT (731h)
Kapcsolódási pont a BASIC utasításelemző rutinjához. Ha újabb utasításokat kívánunk beépíteni a BASIC-be, akkor ide kell elhelyezni azt az ugrást, ami a bővítés rutinra mutat.

1844 — 1855 (734h — 73Fh)
Belső használatra.

A fenti 64 bájtos terület a korábbi változatokban a képernyő 25. sorában levő karakterek számára volt fenntartva, de nem volt kihasználva. Ha a felsorolt kapcsolódási pontokhoz gépi kódú rutinokat szeretnénk hozzáfűzni, akkor előbb mentsük el az ottlevő három bájtot, és helyezzük oda a saját rutinunkra mutató ugrást. Ha a saját rutinunk befejezte a működését, akkor az eredeti kapcsolódási pont tartalmával kell befejeződnie. Ha ezt betartjuk, akkor több egymástól független bővítés is képes lesz együttműködni, természetesen mindig az utolsónak lesz a legnagyobb a prioritása.

3756 – 3761 (0EACh – 0EB1h)
Belső használatra

3762 BLINK (0EB2h)
A billentyűzet input alatti kurzorvillogást vezérli. Ha értéke nulla, akkor nincs kurzorvillogás, egyéb érték esetén villog a kurzor. Az editor input mindig törli.

3763 RESTART (0EB3h)
Alkalmazói programok reset választásához. Ha nem nulla, akkor a szokásos meleg reset történik, ha nulla, akkor az operációs rendszer meleg resetje után az alkalmazói program hideg resetet hajt végre.

3764 MOPS_WARM (0EB4h)
Jelzi az alkalmazói programoknak, hogy az operációs rendszer milyen resetet hajtott végre. Akkor van szerepe, ha az alkalmazói program hideg resetet kap.

3765 CENBLE (0EB5h)
Programmodul engedélyező. Nulla értékkel engedélyezi, egyéb értékkel letiltja a programmodul indítását a BASIC helyett.

3766 BASFLG (0EB6h)
A BASIC indítását vezérli.
bit 0 – átlépi a V-alakban kiírt „VIDEOTON" képernyő kiírását
bit 1 – átlépi a „COPYRIGHT" üzenet kiírását
bit 2 – nem hajt végre kezdéskor „NEW" utasítást
bit 3 – automatikusan végrehajt egy „RUN" utasítást

3767 VERSION (0EB7h)
Változatszám. A felső 4 bit a fő-, az alsó 4 bit pedig az alváltozatszámot adja. Pl. a 2.0 változatnál értéke 20h. A korábbi változatoknál értéke nulla.

3768 – 3774 (0EB8h – 0EBEh)
Jelenleg nem használt.

A fenti 19 bájt korábban a stack vége volt.

3736 – PROGID (0E98h)
Programazonosító 20 bájt az 1.3 verziótól kezdve. A BASIC a következőt írja bele: „TV COMPUTER BASIC". A szöveget 00h kóddal kell zárni. Korábban az editor számára volt fenntartva, de nem volt kihasználva.

5804 – 5887 (16ACh – 16FFh)
A BASIC számára lefoglalva. Ez a terület a rendszer stack és a BASIC változók között korábban nem volt használva.

1. Függelék

A BASIC kulcsszavak egybájtos kódjai – Tokentáblázat

143	(8Fh):	Cannot
144	(90h):	No
145	(91h):	Bad
146	(92h):	rgument
147	(93h):	missing
148	(94h):	nem használt
149	(95h):	)
150	(96h):	(
151	(97h):	&
152	(98h):	+
153	(99h):	<
154	(9Ah):	=
155	(9Bh):	<=
156	(9Ch):	>
157	(9Dh):	<>
158	(9Eh):	>
159	(9Fh):	^

160	(0A0h):	;
161	(0A1h):	/
162	(0A2h):	–
163	(0A3h):	=<
164	(0A4h):	,
165	(0A5h):	><
166	(0A6h):	=>
167	(0A7H):	#
168	(0A8h):	*
169	(0A9h):	nem használt
170	(0AAh):	nem használt
* 171	(0ABh):	POLYGON
* 172	(0ACh):	RECTANGLE
* 173	(0ADh):	ELLIPSE
174	(0AEh):	BORDER
175	(0AFh):	USING

176	(0B0h):	AT
177	(0B1h):	ATN
178	(0B2h):	XOR
179	(0B3h):	VOLUME
180	(0B4h):	TO
181	(0B5h):	THEN
182	(0B6h):	TAB
183	(0B7h):	STYLE
184	(0B8h):	STEP
185	(0B9h):	RATE
186	(0BAh):	PROMPT
187	(0BBh):	PITCH
188	(0BCh):	PAPER
189	(0BDh):	PALETTE
190	(0BEh):	PAINT
191	(0BFh):	OR

192	(0C0h):	ORD
193	(0C1h):	OFF
194	(0C2h):	NOT
195	(0C3h):	MODE
196	(0C4h):	INK
197	(0C5h):	INKEY$
198	(0C6h):	DURATION
199	(0C7h):	DELAY
200	(0C8h):	CHARACTER
201	(0C9h):	AND
202	(0CAh):	nem használt
203	(0CBh):	nem használt
* 204	(0CCh):	EXCEPTION
* 205	(0CDh):	RENUMBER
* 206	(0CEh):	FKEY
* 207	(0CFh):	AUTO

208	(0D0n):	LPRINT
209	(0D1h):	EXT
210	(0D2h):	VERIFY
211	(0D3h):	TRACE
212	(0D4h):	STOP
213	(0D5h):	SOUND
214	(0D6h):	SET
215	(0D7h):	SAVE
216	(0D8h):	RUN
217	(0D9h):	RETURN
218	(0DAh):	RESTORE
219	(0DBh):	READ
220	(0DCh):	RANDOMIZE
221	(0DDh):	PRINT
222	(0DEh):	POKE
223	(0DFh):	PLOT

224	(0E0h):	OUT
225	(0E1h):	OUTPUT
226	(0E2h):	OPEN
227	(0E3h):	ON
228	(0E4h):	OK
229	(0E5h):	NEXT
230	(0E6h):	NEW
231	(0E7h):	LOMEM
232	(0E8h):	LOAD
233	(0E9h):	LLIST
234	(0EAh):	LIST
235	(0EBh):	LET
236	(0ECh):	INPUT
237	(0EDh):	IF
238	(0EEh):	GRAPHICS
239	(0EFh):	GOTO

240	(0F0h):	GOSUB
241	(0F1h):	GET
242	(0F2h):	FOR
243	(0F3h):	END
244	(0F4h):	ELSE
245	(0F5h):	DIM
246	(0F6h):	DELETE
247	(0F7h):	DEF
248	(0F8h):	CONTINUE
249	(0F9h):	CLS
250	(0FAh):	CLOSE
251	(0FBh):	DATA
252	(0FCh):	REM
253	(0FDh):	:
254	(0FEh):	!
255	(0FFh):	sorvégjel

Megjegyzések:

– A *-gal megjelölt kódok jelentése csak a 2.0 változattal kezdődően érvényes, korábban ezeket nem használtuk.

– A 128–159 kódú karakterek szerepelhetnek a BASIC változónevekben, ezért ezeket a tömörítő eljárás 128-cal kisebb kódra, a 0–31 tartományba konvertálja (magyar ékezetes betűk és táblázatrajzoló kakrakterek).

– A 143–147 kódú tokenek a BASIC programsorokban nem fordulnak elő, ezeket a belső hibakezelő rutin használja.

2. Függelék

Hibakódok és hibaüzenetek

A 2.0 verzió hibakódjai és hibaüzenetei a korábbi verziók hibakódjainak megfelelően az alábbiak:

255. . .229 (0FFh. . .0E5h)
Ezeknek a hibakódoknak a jelentése azonos az 1.2 verzióval.

228 – .FKEY (0E4h)
Az FKEY utasítással egy érvénytelen feladatbillentyűt jelöltünk ki, vagy nem fér el a rendelkezésre álló 100 bájt szabad részére az új feladatbillentyűhöz rendelt string.

227 – .POLY (0E3h)
Túl sok oldalt adtunk meg a sokszögrajzoló rutin hívásakor. A maximum érték 20.

226. . .192 (0E2h. . .0C0h)
Jelenleg nem használt.

191. . .128 (0BFh. . .80h)
A VT–DOS operációs rendszer számára fentartva.

A fenti hibáknál (128–255 kódok) a kiírt hibaüzenet („n" a hibakód):

***System error n

127. . .18 (7Fh. . .12h)
„BASIC error n" . BASIC hiba, a hibakód „n", a definiált hibakezelő rutinok saját céljaikra felhasználhatják.

17 – .BADREN (11h)
„Cannot RENUMBER". A RENUMBER parancsot nem tudja végrehajtani a BASIC.

A további BASIC hibaüzenetek ugyanazok, mint az előző verziókban.

16 – .BADFIL (10h)
„Bad file". Hibás file-típus.

15 – .VARDEC (0Fh)
„Variable declared twice". Másodszor definiált változó.

14 – .MISMAT (0Eh)
„Type mismatch". Hibás változó típus.

13 – .OVRFLW (0Dh)
„Overflow". Túlcsordulás.

12 – .CANTRD (0Ch)
„Cannot READ". Hibás READ utasítás.

11 – DIVO (0Bh)
„Cannot divide by 0". Nullával való osztás fordult elő.

10 – .CANTCNT (0Ah)
„Cannot CONTINUE". A CONTINUE utasítás nem hajtható végre.

9 – .NOGOSUB
„No GOSUB". RETURN utasítás GOSUB nélkül.

8 – NOFOR
„No FOR". NEXT utasítás FOR nélkül.

7 – .NODATA
„No DATA". A DATA listában elfogytak az elemek.

6 – .NOMEM
„No memory". A művelethez nincs elég memória.

5 – .BADSUB
„Bad subscript". Hibás indexkifejezés.

4 – .BADARG
„Bad argument". Hibás paraméter.

3 – .NOARG
„Argument missing". Hiányzó paraméter.

2 – .NOLINE
„Line missing". Hivatkozás nemlétező sorszámra.

1 – .SYNTER
„Not understood". Ismeretlen kifejezés.

0 – Ez a kód jelzi a hibátlan működést.

3. Függelék

Példaprogramok

A BASIC 2.0 változat új tulajdonságait néhány példaprogramban mutatjuk be.

1. BAUD: új BASIC utasítás

A soros vonal átviteli sebességét egy POKE utasítással lehet változtatni, az egyes sebességekhez tartozó kódot külön táblázatból kell kikeresni. Az átviteli formátum beállításához a kívánt paramétereket (paritás, adat és stopbitek száma) egy bájtba kell kódolni, és POKE utasítással a megfelelő helyre írni. Végül egy harmadik POKE utasítással be kell állítani egy jelzőt, hogy a paraméterek megváltoztak.

A közölt program hozzáfűzi a BASIC utasításkészlethez a BAUD nevű utasítást. Ez lehetővé teszi a soros átviteli paraméterek módosítását, méghozzá könnyen megjegyezhető formában. A program tartalmazza az assembler listát is.

2. String összehasonlítás

A program a STRCMP kapcsolódási pontot felhasználva megváltoztatja a BASIC string összehasonlítást: az A és B betűk sorrendjét felcseréli, tehát „A" < „B" helyett „B" < „A" lesz érvényes. A program tartalmazza az assembler listát is.

3. Nyomtatásra küldött karakterek átkódolása

A PRNDEF kapcsolódási pont használatát mutatja a program. Csak a példa kedvéért a szóköz karaktereket a „<>" karakterpárra cseréli. A programról assembler listát is közlünk.

4–5. BASIC program betöltése ROM-ból

A közölt két program nagyon hasonló, lehetővé teszi BASIC programok betöltését ROM-ból. Az első a programmodul ROM-hoz, a második pedig a bővítőkártya ROM-hoz használható.

6. Hibakezelés BASIC-ben

A közölt program a BASIC hibakezelésre egy egyszerű példa.

1. Példaprogram

```
100 ! Ez a program beír a memóriába egy gépi kódú rutint.
110 ! A rutin a BASEXT kapcsolódási ponton beláncolódik a
120 ! BASIC 2.0-ba, és bővítésként megvalósítja a „BAUD"
130 ! utasítást, amellyel a soros vonal átviteli jellem-
140 ! zőit lehet egyszerűen beállítani. A szintaxis:
150 !
160 !     BAUD <sebesség>,<paritás>,<adatbit>,<stopbit>
170 !
180 ! A paraméterek közül <paritás> stringkifejezés, a
190 ! többi pedig numerikus kifejezés kell legyen.
200 ! <sebesség> lesz az új átviteli sebesség, lehetséges
210 ! értékei 110, 150, 300, 600, 1200, 2400, 4800 vagy
220 ! 9600. A 19200 baud sebességet csak az egyszerűbb
230 ! elemzés miatt nem engedjük meg. <paritás> adja az
240 ! átvitel paritását, lehetséges értéke „N", azaz
250 ! nincs paritásbit, „E", azaz páros paritás és „O",
260 ! azaz páratlan paritás. <adatbit> adja az adatbitek
270 ! számát, értéke 7 vagy 8 lehet, végül <stopbit>
280 ! adja a stopbitek számát, értéke 1 vagy 2 lehet.
290 !
300 ! Az utasításelemzés egyszerűsítése érdekében minden
310 ! paramétert kötelező megadni, nincs alapértelmezés.
320
330 LOMEM 7000                              ! Hely a rutinnak.
340 POKE DEC("1720"),PEEK(DEC("1722"))     ! VLOMEN-be az új
```

```
350 POKE DEC("1721"),PEEK(DEC("1723"))   !   LOMEM-et írja.
360
370 LET ADDR=DEC("19ef")                !  Rutin kezdőcím.
380
390 READ BYTE$                          !  Byte olvasás és
400 IF BYTE$="END" THEN 450             !     memóriába írás
410 POKE ADDR,DEC(BYTE$)
420 LET ADDR=ADDR+1
430 GOTO 390
440
450 POKE dec("731"),DEC("c3")           !  BASEXT-be beírja a
460 POKE DEC("732"),DEC("ef")           !     kezdőcímre ugrást.
470 POKE DEC("733"),DEC("19")
480
490 END
500
```



```
510  DATA d9, e5, 11, a5, 1a, 06, 04, 1a, be, 13, 23, 28, 03, e1, d9, c9
520  DATA 10, f5, c1, d9, df, 2d, 2b, a7, e5, cd, 80, 1a, df, ac, cd, 80
530  DATA 1a, cd, 8c, 1a, f5, cd, 80, 1a, cd, 8c, 1a, 0e, 00, 3d, 28, 05
540  DATA 3d, 20, 4b, cd, f9, f1, fe, 07, 28, 06, fe, 08, 20, 40, cb, d1
550  DATA fd, 7e, 01, 3b, 20, 38, fd, 7e, 02, fd, 23, fd, 23, fd, 23, fe
560  DATA 4e, 28, 18, fe, 6e, 28, 14, cb, e1, fe, 4f, 28, 0e, fe, 6f, 28
570  DATA 0a, fe, 45, 28, 04, fe, 65, 20, 15, cb, e9, d1, c5, 21, 95, 1a
580  DATA 06, 08, 7b, be, 23, 20, 04, 7a, be, 28, 05, 3e, ff, 32, 71, 04
590  DATA 78, 3d, 32, 69, 0b, c1, 79, 32, 6a, 0d, 3e, c9, cf, 01, df, e1
600  DATA c9, d9, 78, b9, fe, a4, 20, 03, df, ad, c9, cf, 01, df, 2b, a7,
610  DATA 7b, 24, 25, c8, cf, 04, 80, 25, c0, 12, 60, 09, b0, 04, 58, 02
620  DATA 2c, 01, 96, 00, 6e, 00, 42, 41, 55, 44
630  DATA "END"
640
650 ! A gépi kódú rutin forráslistája:
660 !
670 !                   ; MOPS változók címei
680 !                   ;
690 !                   $BAUD     EQU   0B69h     ; Átviteli seb.
700 ! 0B69              $FORMAT   EQU   0B6Ah     ; Átv. formátum
710 ! 0B6A              SER_OK    EQU   0B71h     ; FF ha változik
720 ! 0B71                                        ;   az átvitel
730 !
740 !                   ; Hibakódok
750 !                   ;
760 !
```

```
770 ! 0004                .BADARG  EQU   4          ; „Bad argument"
780 ! 0001                .SYNTER  EQU   1          ; „Not understood"
790 !                     ;
800 !                     ; BASIC funkciók
810 !                     ;
820 ! 0027                POP      EQU   27h        ; Szám HL-be
830 ! 002B                FEXPR    EQU   2Bh        ; Num.kif. kiért.
840 ! 002C                $EXPR    EQU   2Ch        ; String kiért.
850 ! 002D                GET      EQU   2Dh        ; Köv. prg.egység
860 !                     ;
870 !                     ;
880 ! 19EF                         ORG   19EFh
890 !                     ;
900 !                     BAUD:
910 ! 19EF  D9                    EXX                ; HL'=prg.mutató
920 ! 19F0  E5                    PUSH  HL           ; Mentés későbbre.
930 ! 19F1  11A51A                LD    DE,STMT      ; Új utasítás neve
940 ! 19F4  0604                  LD    B,STLEN      ;   és hossza.
950 !                     COMPARE:
960 ! 19F6  1A                    LD    A,(DE)       ; Karakterek
970 ! 19F7  BE                    CP    (HL)         ; összehasonlítása
980 ! 19F8  13                    INC   DE
990 ! 19F9  23                    INC   HL
1000 ! 19FA 2803                  JR    Z,CHAROK     ; Ugrás, ha jó.
1010 !                    ;
1020 ! 19FC E1                    POP   HL           ; Régi HL' vissza-
```



```
1030 ! 19FD  D9                   EXX                ;   állítása és
1040 ! 19FE  C9                   RET                ;   visszatérés.
1050 !                    ;
1060 !                    CHAROK:
1070 ! 19FF  10F5                 DJNZ  COMPARE      ; Összes betű has.
1080 !                    ;
1090 ! 1A01  C1                   POP   BC           ; Régi HL' eldobás
1100 ! 1A02  D9                   EXX                ; HL'=új mutató
1110 !                    ;
1120 ! 1A03  DF                   RST   18h          ; BASIC funkció.
1130 ! 1A04  2D                   DEFB  GET          ; Köv. prg-rész
1140 ! 1A05  2B                   DEFB  FEXPR        ; Num.kif.kiért.
1150 ! 1A06  A7                   DEFB  POP OR 80h   ; HL=eredmény
1160 !                    ;
1170 ! 1A07  E5                   PUSH  HL           ; Mentés későbbre.
1180 ! 1A08  CD801A               CALL  COMMA        ; Vesszőt átlépi.
1190 !                    ;
1200 ! 1A0B  DF                   RST   18h          ; BASIC funkció.
1210 ! 1A0C  AC                   DEFB  $EXPR OR 80 h ;   Paritás
1220 !                    ;                             string kiért.
1230 ! 1A0D  CD 801A              CALL  COMMA        ; Vesszőt átlépi.
1240 !                    ;
1250 ! 1A10  CD8C1A               CALL  POPA         ; A=adatbitek
1260 ! 1A13  F5                   PUSH  AF           ; Mentés későbbre.
1270 !                    ;
1280 ! 1A14  CD801A               CALL  COMMA        ; Vesszőt átlépi.
```

```
1290 !                          ;
1300 ! 1A17   CD8C1A                    CALL   POPA            ; A=stopbitek
1310 !                          ;
1320 ! 1A1A   0E00                      LD     C,0             ; $FORMAT törlése.
1330 ! 1A1C   3D                        DEC    A               ; 1 stopbit ?
1340 ! 1A1D   2805                      JR     Z,STOPOK        ; Igen: BIT 7,C=0
1350 !                          ;
1360 ! 1A1F   3D                        DEC    A               ; 2 stopbit ?
1370 ! 1A20   204B                      JR     NZ,BADARG       ; Hibás stopbit.
1380 !                          ;
1390 ! 1A22   CBF9                      SET    7,C             ; Igen: BIT 7,C=1
1400 !                          STOPOK:
1410 ! 1A24   F1                        POP    AF              ; A=adatbitek
1420 ! 1A25   FE07                      CP     7               ; 7 adatbit ?
1430 ! 1A27   2806                      JR     Z,DATAOK        ; Igen: BIT 2,C=0
1440 !                          ;
1450 ! 1A29   FE08                      CP     8               ; 8 adatbit ?
1460 ! 1A2B   2040                      JR     NZ,BADARG       ; Hibás adatbit.
1470 !                          ;
1480 ! 1A2D   CBD1                      SET    2,C             ; Igen: BIT 2,C=1
1490 !                          DATAOK:
1500 ! 1A2F   FD7E01                    LD     A,(IY+1)        ; Paritás string
1510 ! 1A32   3D                        DEC    A               ;   1 byte ?
1520 ! 1A33   2038                      JR     NZ,BADARG       ; Hibás param.
1530 !                          ;
1540 ! 1A35   FD7E02                    LD     A,(IY+2)        ; A=karakter
```

```
1550 !                          ;
1560 ! 1A38   FD23                      INC    IY              ; Kiveszi a BASIC
1570 ! 1A3A   FD23                      INC    IY              ;   stack-ből a
1580 ! 1A3C   FD23                      INC    IY              ;   stringet.
1590 !                          ;
1600 ! 1A3E   FE4E                      CP     'N'             ; Nincs paritás ?
1610 ! 1A40   2818                      JR     Z,PAROK         ; Igen: BIT 4,C=0
1620 !                          ;
1630 ! 1A42   FE6E                      CP     'n'
1640 ! 1A44   2814                      JR     Z,PAROK
1650 !                          ;                               Van paritás:
1660 ! 1A46   CBE1                      SET    4,C             ;   BIT 4,C=1
1670 ! 1A48   FE4F                      CP     'O'             ; Ptl. paritás ?
1680 ! 1A4A   280E                      JR     Z,PAROK         ; Igen: BIT 5,C=0
1690 !                          ;
1700 ! 1A4C   FE6F                      CP     'o'
1710 ! 1A4E   280A                      JR     Z,PAROK
1720 !                          ;
1730 ! 1A50   FE45                      CP     'E'             ; Páros paritás ?
1740 ! 1A52   2804                      JR     Z,SETE          ; Igen: BIT 5,C=1
1750 !                          ;
1760 ! 1A54   FE65                      CP     'e'
1770 ! 1A56   2015                      JR     NZ,BADARG       ; Hibás param.
1780 !                          ;
1790 !                          SETE:
1800 ! 1A58   CBE9                      SET    5,C             ; BIT 5,C=1
```

```
1810 !                  PAROK:
1820 ! 1A5A   D1                    POP     DE          ; DE=sebesség
1830 ! 1A5B   C5                    PUSH    BC          ; C elmentése
1840 !                                                    $FORMAT-hoz.
1850 ! 1A5C   21951A                LD      HL,BAUDTAB  ; Seb.táblázat
1860 ! 1A5F   0608                  LD      B,TABLEN    ; Elemek száma.
1870 !                  CHKBD:
1880 ! 1A61   7B                    LD      A,E
1890 ! 1A62   BE                    CP      (HL)
1900 ! 1A63   23                    INC     HL
1910 ! 1A64   2004                  JR      NZ,BAUDBD   ; Nem találta.
1920 !                  ;
1930 ! 1A66   7A                    LD      A,D
1940 ! 1A67   BE                    CP      (HL)
1950 ! 1A68   2805                  JR      Z,GOTBD     ; Megvan: B-1=kód
1960 !                                                    $BAUD-hoz
1970 !                  BADBD:
1980 ! 1A6A   23                    INC     HL
1990 ! 1A6B   10F4                  DJNZ    CHKBD       ; Köv. táblaelem
2000 !                  ;
2010 !                  BADARG:
2020 ! 1A6D   CF                    RST     8           ; Táblázat vége,
2030 ! 1A6E   04                    DEFB    .BADARG     ;   hibás param.
2040 !                  ;
2050 !                  GOTBD:
2060 ! 1A6F   78                    LD      A,B
```

```
2070 ! 1A70   3D                    DEC     A           ; Sebességkód A-ba
2080 ! 1A71   32690B                LD      ($BAUD),A   ; $BAUD beáll.
2090 !                  ;
2100 ! 1A74   C1                    POP     BC          ; C=formátumkód
2110 ! 1A75   79                    LD      A,C
2120 ! 1A76   326A0B                LD      ($FORMAT),A ; $FORMAT-ba
2130 !                  ;
2140 ! 1A79   3EFF                  LD      A,0FFh      ; FF=param.váltás
2150 ! 1A7B   32710B                LD      (SER_OK),A  ; SER_OK-ba
2160 !                  ;
2170 ! 1A7E   E1                    POP     HL          ; Hibátlan futás.
2180 ! 1A7F   C9                    RET                 ; BASIC-be vissza
2190 !                  ;
2200 !                  ;COMMA rutin az elválasztójelet vizsgálja,
2210 !                  ; ha vessző (,) volt, akkor továbblép.
2220 !                  ;
2230 !                  COMMA:
2240 ! 1A80   D9                    EXX
2250 ! 1A81   78                    LD      A,B         ; A=akt. token
2260 ! 1A82   D9                    EXX
2270 ! 1A83   FEA4                  CP      0A4h        ; Vessző ?
2280 ! 1A85   2003                  JR      NZ,SYNTER   ; Hiba, ha nem
2290 !                  ;
2300 ! 1A87   DF                    RST     18h         ; BASIC funkció
2310 ! 1A88   AD                    DEFB    GET OR 80h  ; Köv. param.
2320 !                  ;
```

```
2330 ! 1A89   C9                    RET
2340 !                        ;
2350 !                        SYNTER:
2360 ! 1A8A   CF              .     RST   8              ; Hibagenerálás
2370 ! 1A8B   01                    DEFB  .SYNTER        ; Hibakód
2380 !                        ;
2390 !                        ; POPA rutin kiértékel egy numerikus
2400 !                        ; kifejezést, az értékét A-ban adja át.
2410 !                        ;
2420 !                         POPA:
2430 ! 1A8C   DF                    RST   18h            ; BASIC funkció
2440 ! 1A8D   2B                    DEFB  FEXPR          ; Num.kif. kiért.
2450 ! 1A8E   A7                    DEFB  POP OR 80h     ; HL=eredmény
2460 !                        ;
2470 ! 1A8F   7D                    LD    A,L            ; A=eredmény
2480 ! 1A90   24                    INC   H
2490 ! 1A91   25                    DEC   H              ; HL > 255 ?
2500 ! 1A92   C8                    RET   Z              ; Nem, visszatér
2510 !                        ;
2520 ! 1A93   CF                    RST   8              ; Hibagenerálás
2530 ! 1A94   04                    DEFB  .BADARG        ; Hibakód
2540 !                        ;
2550 !                        ; Érvényes átviteli sebességek táblázata
2560 !                        ; és a táblaelemek száma
2570 !                        ;
2580 !                        BAUDTAB:
```

```
2590 ! 1A95   8025                  DEFW  9600
2600 ! 1A97   C012                  DEFW  4800
2610 ! 1A99   6009                  DEFW  2400
2620 ! 1A9B   B004                  DEFW  1200
2630 ! 1A9D   5802                  DEFW  600
2640 ! 1A9F   2C01                  DEFW  300
2650 ! 1AA1   9600                  DEFW  150
2660 ! 1AA3   6E00                  DEFW  110
2670 !                        ;
2680 ! 0008                   TABLEN  EQU   ($-BAUDTAB)/2
2690 !                        ;
2700 !                        ; Uj utasítás neve és hossza
2710 !                        ;
2720 !                        STMT:
2730 ! 1AA5   42415544              DEFM  'BAUD'
2740 !                        ;
2750 ! 0004                   STLEN   EQU   $-STMT
2760 !                        ;
2770 ! 1AA9                         END
```

```
100 ! Ez a program beír a memóriába egy gépi kódú rutint.
110 ! A rutin az STRCMP kapcsolódási ponton beláncolódik
120 ! a string összehasonlításba. Csupán a példa kedvéért
130 ! felcseréli az „A" és „B" karakterek kódsorrendjét,
140 ! így ezután „B" string kisebb lesz, mint „A" string.
150
160 LOMEM 7000                              ! Hely a rutinnak.
170 POKE DEC("1720"),PEEK(DEC("1722"))      ! VLOMEM-be az új
180 POKE DEC("1721"),PEEK(DEC("1723"))      !   LOMEM-et írja
190
200 LET ADDR=DEC("19ef")                    ! Rutin kezdőcím.
210
220 READ BYTE$                              ! Byte olvasás és
230 IF BYTE$="END" THEN 280                 !   memóriába írás.
240 POKE ADDR,DEC(BYTE$)
250 LET ADDR=ADDR+1
260 GOTO 220
270
280 POKE DEC("72e"),DEC("c3")               ! STRCMP-be beírja a
290 POKE DEC("72f"),DEC("ef")               !   kezdőcímre ugrást.
300 POKE DEC("730"),DEC("19")
310
320 END
330
340 DATA  d5, e5, f5, 7e, cd, ff, 19, 4f, f1, cd, ff, 19, b9
350 DATA  e1, d1, c9, fe, 41, 28, 06, fe, 42, c0, 3e, 41, c9
```



```
360 DATA  3e, 42, c9
370 DATE "END"
380
390 ! A gépi kódú rutin forráslistája:
400 !
410 ! 19EF                          ORG   19EFh
420 !                    ;
430 ! 19EF    D5                    PUSH  DE          ; Regisztermentés
440 ! 19F0    E5                    PUSH  HL
450 !                    :
460 ! 19F1    F5                    PUSH  AF          ; 1.kar. mentése
470 ! 19F2    7E                    LD    A,(HL)      ; 2.kar. A-ba
480 1 19F3    CDFF19                CALL  TRANS       ; 2.kar. átírása
490 ! 19F6    4F                    LD    C,A         ;   és mentése
500 ! 19F7    F1                    POP   AF          ; 1.kar. vissza
510 !                    ;
520 ! 19F8    CDFF19                CALL  TRANS       ; 1.kar. átírása
530 ! 19FB    B9                    CP    C           ; összehasonlítás
540 !                    ;
550 ! 19FC    E1                    POP   HL          ; Regiszter vissza
560 ! 19FD    D1                    POP   DE
570 ! 19FE    C9                    RET               ; Visszatérés,
580 !                    ;
590 !                    ;
600 !                    TRANS:
610 ! 19FF    FE41                  CP    'A'
```

```
620 ! 1A01    2806              JR    Z,GOTA      ; ,A'-t ,B'-re
630 !                     ;
640 ! 1A03    FE42              CP    'B'         ; ,B'-t ,A'-ra
650 ! 1A05    C0                RET   NZ
660 !                     ;
670 ! 1A06    3E41              LD    A,'A'
680 ! 1A08    C9                RET
690 !                     ;
700 !                     GOTA:
710 ! 1A09    3E42              LD    A,'B'
720 ! 1A0B    C9                RET
730 !                     ;
740 ! 1A0C                      END
```



```
100 ! Ez a program beír a memóriába egy gépi kódú  rutint.
110 ! A rutin a PRNDEF kapcsolódási ponton beláncolódik a
120 ! nyomtatót kezelő programba. Csupán a példa kedvéért
130 ! minden szóköz karaktert kicserél „<>" karakterpárra.
140
150 LOMEM 7000                          ! Hely a rutinnak.
160 POKE DEC("1720"),PEEK(DEC("1722"))  ! VLOMEM-be az új
170 POKE DEC("1721"),PEEK(DEC("1723"))  !   LOMEM-et írja.
180
190 LET ADDR=DEC("19ef")                ! Rutin kezdőcím.
200
210 READ BYTE$                          ! Byte olvasás és
220 IF BYTE$="END" THEN 270             !   memóriába írás.
230 POKE ADDR,DEC(BYTE$)
240 LET ADDR=ADDR+1
250 GOTO 210
260
270 POKE DEC("725"),DEC("c3")           ! PRNDEF-be beírja a
280 POKE DEC("726"),DEC("ef")           !   kezdőcímre ugrást.
290 POKE DEC("727"),DEC("19")
300
310 END
320
330 DATA 79,fe,20,c0,0e,3c,06,3e,c9
340 DATA "END"
350
```

360 ! A gépi kódú rutin forráslistája:
370 !
380 ! 19EF
390 !
400 ! 19EF 79
410 ! 19F0 FE20
420 ! 19F2 C0
430 !
440 ! 19F3 0E3C
450 ! 19F5 063E
460 ! 19F7 C9
470 !
480 ! 19F8

```
                ORG   19EFh
;
                LD    A,C       ; BC változatlan,
                CP    ' '       ;   ha nem szóközt
                RET   NZ        ;   nyomtatunk.
;
                LD    C,'<'     ; "<>" nyomtatása
                LD    B,'>'     ;   szóköz helyett
                RET
;
                END
```



```
;
; Ez a program a programmodul ROM elejére készült.
; Közvetlenül ezután egy BASIC programnak kell
; lennie, abban a formában, ahogy azt a SAVE
; paranccsal kimentettük. A fejléc 16 bájtjára is
; szükség van, mert ez tartalmazza a program
; hosszát, de nincs szükség arra az extra 128
; bájtra, ami a .CAS végződésű lemezes fájlok
; elején található. Indításkor bemásolja az őt
; követő BASIC programot a memóriába, majd úgy tér
; vissza a BASIC-be, hogy a bemásolt program
; RUN-nal induljon.
;
;
0030            P_UUUC      EQU   30h       ; Memória: U0/U1/U2/CART
0020            P_SUUC      EQU   20h       ; Memória: SYS/U1/U2/CART
0002            PAGES       EQU   2         ; Memórialapozó port
;
OB19            HIMEM       EQU   0B19h     ; Memória vége mutató
0B21            WARM_FLAG   EQU   0B21h     ; NZ => meleg reset
19EF            PROGRAM     EQU   19EFh     ; BASIC program terület
;
0EB6            BASFLG      EQU   0EB6h     ; Bit0=1 nincs kezdőkép
                                            ; Bit1=1 nincs Copyright
                                            ; Bit2=1 nincs NEW
                                            ; Bit3=1 automatikus RUN
```

```
C000                            ORG     0C000h
                        ;
C000    4D4F5053                DEFM    'MOPS'          ; Programmodul jelző
                        ;
C004    3E30                    LD      A,P_UUUC
C006    D302                    OUT     (PAGES),A       ; RAM belapozás
                        ;
C008    D9                      EXX                     ; DE' elmentése
                        ;
C009    3A210B                  LD      A,(WARM_FLAG)
C00C    B7                      OR      A               ; Meleg reset ?
C00D    201E                    JR      NZ,BASIC        ; Igen: BASIC-be lép
                        ;
C00F    2A39C0                  LD      HL,(END+2)      ; Fejléc: prg.hossz
C012    E5                      PUSH    HL              ; Mentés későbbre
C013    11EF19                  LD      DE,PROGRAM      ; BASIC prg.terület
C016    19                      ADD     HL,DE           ; Prg. utolsó bájtja
C017    24                      INC     H               ; BASIC stack-nek hely
C018    24                      INC     H               ; Szimbólumtáblának hely
C019    ED4B190B                LD      BC,(HIMEM)      ; Max.használható cím
C01D    B7                      OR      A
C01E    ED42                    SBC     HL,BC           ; Elég a memória ?
C020    C1                      POP     BC              ;   BC=prg.hossz
C021    300B                    JR      NC,BASIC        ; Nem: BASIC-be lép
C023    2147C0                  LD      HL,END+16       ; HL=prg.kezdet
C026    EDB0                    LDIR                    ; Másolás a BASIC RAM-ba
```

```
                        ;
C028    3E0C                    LD      A,0Ch           ; Jelző: nem kell NEW, és
C02A    32B60E                  LD      (BASFLG),A      ;   autom RUN kell
                        BASIC:
C02D    D9                      EXX
C02E    21EAFF                  LD      HL,-22          ; BASIC-be visszatéréshez
C031    19                      ADD     HL,DE
C032    3E20                    LD      A,P_SUUC        ; BASIC ROM belapozása
C034    D302                    OUT     (2),A           ;
C036    E9                      JP      (HL)            ; Vissza a DE' -22 címre
                        END:
C037                            END

Macros:

Symbols:

0EB6    BASFLG  C02D    BASIC   C037    END
0B19    HIMEM   0002    PAGES   19EF    PROGRAM
0020    P_SUUC  0030    P_UUUC  0B21    WARM_FLAG

No Fatal error(s)
```

```
                        ;
                        ; Ez a program a bővítőkártya ROM elejére készült.
                        ;Közvetlenül ezután egy BASIC programnak kell
                        ;lennie, abban a formában, ahogy azt a SAVE
                        ;paranccsal kimentettük. A fejléc 16 bájtjára is
                        ;szükség van, mert ez tartalmazza a program
                        ;hosszát, de nincs szükség arra az extra 128
                        ;bájtra, ami a .CAS végződésű lemezes fájlok
                        ;elején található. Indításkor bemásolja az őt
                        ;követő BASIC programot a memóriába, majd úgy tér
                        ;vissza a MOPS-ba, hogy a bemásolt programot a
                        ;BASIC egy RUN-nal indítsa.
                        ;
                        ;
0B19                    HIMEM       EQU   0B19h     ; Memória vége mutató
0B21                    WARM_FLAG   EQU   0B21h     ; NZ => meleg reset
19EF                    PROGRAM     EQU   19EFh     ; BASIC program terület
                        ;
0EB6                    BASFLG      EQU   0EB6h     ; Bit0=1 nincs kezdőkép
                                                    ; Bit1=1 nincs Copyright
                                                    ; Bit2=1 nincs NEW
                                                    ; Bit3=1 automatikus RUN
                                ORG   0C000h
C000                    ;
C000    4D4F5053                DEFM  'MOPS'        ; Bővítőkártya ROM jelző
```

```
                        ;
C004    04                      DEFB  4             ; Kártyanév hossza
C005    54455354                DEFM  'TEST'        ; Fiktív kártyanév
c009    0000                    DEFW  0             ;   (6 bájtra kieg.)
                        ;
C00B    0EC0                    DEFW  INIT          ; Inic. rutin kezdőcíme
                        ;
C00D    00                      DEFB  0             ; Nincs kártyakez. rutin
                        ;
                        INIT:
C00E    3A210B                  LD    A,(WARM_FLAG)
C011    B7                      OR    A             ; Meleg reset ?
C012    C0                      RET   NZ            ; Igen: vissza a MOPS-ba
                        ;
C013    2A33C0                  LD    HL,(END+2)    ; Fejléc: prg.hossz
C016    E5                      PUSH  HL            ; Mentés későbbre
C017    11EF19                  LD    DE,PROGRAM    ; BASIC prg.terület
C01A    19                      ADD   HL,DE         ; Prg. utolsó bájtja
C01B    24                      INC   H             ; BASIC stack-nek hely
C01C    24                      INC   H             ; Szimbólumtáblának hely
C01D    ED4B190B                LD    BC,(HIMEM)    ; Max.használható cím
C021    B7                      OR    A
C022    ED42                    SBC   HL,BC         ; Elég a memória ?
C024    C1                      POP   BC            ;   BC=prg.hossz
C025    D0                      RET   NC            ; Nem: vissza a MOPS-ba
                        ;
```

```
C026    2141C0          LD      HL,END+16       ; HL=prg.kezdet
C029    EDB0            LDIR                    ; Másolás a BASIC RAM-ba
                ;
C02B    3E0C            LD      A,0Ch           ; Jelző: nem kell NEW, és
C02D    32B60E          LD      (BASFLG),A      ;   autom.RUN kell
                ;
C030                    RET                     ; Vissza a MOPS-ba
C031                END:
                        END
```

Macros:

Symbols:

0EB6	BASFLG	C031	END	0B19	**HIMEM**
C00E	INIT	19EF	PROGRAM	0B21	**WARM_FLAG**

No Fatal error(s)


```
 80 ! BASIC program a hibarutin bemutatására
 90
100 ON EXCEPTION GOTO 1000                  ! hibarutin kijelölés
110 GRAPHICS 4:SET BORDER 16
120 BLINK=3762                              ! kurzorvillogás-vezérlő
130 V=300:F=200:D=2                         ! PLOT RECTANGLE paraméterek
140 A=50:B=180:C=160                        ! egyéb PLOT paraméterek
200
220 FOR S=1 TO 9                            ! 9 féle eset
222 :  PLOT A,B
230 :  ON S GOSUB 410,420,430,440,450,460,470,480,490
232 :  PLOT +C,+0;+0, +C
234 :  POKE BLINK,255 :GET :POKE BLINK,0    ! kurzor villog
236 :  A=A+50 :B=B-20
240 NEXT
250 PLOT  ;POLYGON(F,12, 8,2*PI/3)          ! kimegy a képről
260 END
270
400 !    PLOT paraméterezés elválasztójelekkel
410 PLOT RECTANGLE(V,F,D)    :RETURN
420 PLOT ,RECTANGLE(V,F,D)   :RETURN
430 PLOT ;RECTANGLE(V,F,D)   :RETURN
440 PLOT RECTANGLE (V,F,D),  :RETURN
450 PLOT RECTANGLE(V,F,D);   :RETURN
460 PLOT ,RECTANGLE(V,F,D),   :RETURN
470 PLOT ,RECTANGLE(V,F,D);   :RETURN
```

```
480 PLOT ;RECTANGLE(V,F,D)     :RETURN
490 PLOT ;RECTANGLE(V,F,D);    :RETURN
500
1000 ! hibarutin                                    ! STOP-ra folytatja
1010 IF ERRNUM=245 THEN CONTINUE                    ! hibaüzenettel leáll
1020 EXCEPTION ERRNUM,ERRLIN
```