BASIC
programozási segédlet

VIDEOTON

TV-Computer

B A S I C
programozási segédlet

Írta:

BALOGH LÁSZLÓ

Szakmai ellenőr:

DR. MOLNÁR IMRE okl. villamosmérnök

Szerkesztőbizottság:

GARAI GÉZA okl. villamosmérnök
GÁSPÁR CSABA okl. villamosmérnök
VELKEI ZOLTÁN okl. villamosmérnök-közgazdász

Felelős kiadó: DR. BARÁTH CSABA

Készült a Delta Szaklapkiadó
és Műszaki Szolgáltató Leányvállalat gondozásában
MŰSZOLG Print 850050

Tartalomjegyzék

BEVEZETÉS

A könyv első kötetéből Ön megismerhette a TV-COMPUTER felépítését, üzembehelyezésének módját. Az ott leírt példák segítségével kipróbálhatta a gép lehetőségeinek egy részét, eközben egy kicsit már a kezelést is gyakorolta.

Mivel ezt a könyvet fellapozta, biztosak vagyunk benne, hogy szeretne megismerkedni a BASIC programozással és ezáltal egyre jobban azzal a számítógéppel, mely talán kicsit még idegen az Ön számára, de amellyel biztosan barátságot fog kötni.
Szeretnénk felhívni a figyelmét, hogy nem BASIC tankönyvet tart a kezében! (Az egy másik kiadványunk lesz.)

A BASIC napjainkban a világ talán legelterjedtebb programozási nyelve. Karrierje elsősorban a mikroszámítógépek és a személyi számítógépek rohamos terjedésével magyarázható. A BASIC nyelv egyszerű, a köznapi gondolkodáshoz közel álló szabályai, áttekinthetősége következtében vált népszerűvé a nem szakmai felhasználók körében is. Hozzájárult ehhez az is, hogy a BASIC képernyőre és billentyűzetre alapozott ember–gép kapcsolatot valósít meg. Ezeken keresztül a felhasználó párbeszédet folytat a géppel. A témában számos magyar nyelvű kiadvány jelent meg, ezeket figyelmébe ajánljuk.

Ez a könyv a teljességre való törekvés igénye nélkül szeretné bevezetni az olvasót a TV-COMPUTER programozásába, ismertetve a BASIC nyelv elemeit, a parancsokat, az utasításokat és a függvényeket, példákkal illusztrálva használatukat.

Az előkészület alatt álló TV-COMPUTER BASIC tankönyv és példatár, és az Assembly programozási tankönyv még behatóbb ismereteket fog Önnek nyújtani a TV-COMPUTER programozásához, alkalmazásához.

Ha Ön már programozott BASIC-ben, akkor is hasznos lehet ezen könyv elolvasása, mivel a TV-COMPUTER BASIC nyelve több helyen eltér más BASIC megvalósításoktól.

Sok sikert kívánunk!

A TV COMPUTER BASIC ELEMEI

Néhány szó a programozásról

Bármilyen egyszerű feladatot szeretnénk egy számítógéppel elvégeztetni, a megoldás minden lépését a gép számára érthető nyelven kell leírnunk. Ezt a leírást nevezzük programnak. A számítógép nem "gondolkodó gép", csak annak a programnak a végrehajtására képes, amelyet az ember, a programozó megalkotott. Felmerült ezek után a kérdés, hogy mi a számítógép előnye, haszna? Elsősorban az, hogy mentesítheti az embert a mechanikus rutinfeladatok elvégzése alól; ezeket a gép precízen és gyorsan hajtja végre. A számítógép műveletvégző sebessége sokszorosan meghaladja az ember ilyen irányú képességeit. Fontos tényező, hogy a számítógép alkalmazása, ezen belül a programozás, fejleszti a minden emberben meglevő kreatív képességeket.

A BASIC program felépítése

A program sorokból áll, a sorok számát a memóriakapacitás korlátozza. Általánosan így épül fel egy sor:

sorszám utasítások REM megjegyzés

Egy sorra a sorszáma alapján hivatkozhatunk, ezt kötelező megadni. Értéke egy 1 és 9999 közé eső szám.

A memóriában a programsorok a sorszámaik szerint növekvő sorrendben helyezkednek el. Ezért célszerű, ha a programot ötösével, vagy tizesével sorszámozzuk, hogy lehetőségünk legyen utólag sorok beszúrására. Egy sorban egy, vagy több utasítás írható, az utasításokat a : választja el egymástól. A megjegyzést a REM (remark) kulcsszó után lehet elhelyezni, de ezután újabb utasítás nem állhat, mert a BASIC nem hajtja végre. A sor teljes hossza maximum 250 karakter lehet.

Az utasítások felépítése

Minden utasításnak van egy kulcsszava – pl. INPUT, READ – amelyről a BASIC azonosítani tudja a kívánt műveletet. Egy utasítás azonban többféle módon is működhet, a megadott paraméterektől függően, pl. a PRINT "TV-COMPUTER" utasítás a TV-COMPUTER szöveget írja ki a képernyőre, a PRINT A, B pedig az A és B változók értékeit.
A paramétereket kifejezésekkel adhatjuk meg. Egy kifejezés tartalmazhat konstanst, változót, tömbnevet, függvényt, aritmetikai és logikai műveleti operátorokat.

Konstansok

A konstans a programon belül egy állandó értéket képvisel és két típusa lehet: numerikus és string.

A numerikus konstans legnagyobb értéke 0,9999999999E+63. Az E utáni szám az exponens. A legkisebb szám a 0,1E-63.

"Ez egy string konstans", vagyis egy szöveg, idézőjelek közé téve. (Elterjedt még a karakterlánc vagy karakterfüzér elnevezés is.)

Stringkonstansba beírhatók:
- betűk
- számok
- írásjelek
- táblázatrajzoló karakterek
- felhasználó által definiálható karakterek.

A TV-Computer BASIC-ben a stringkonstansokat csak LET esetén kötelező "-jelek közé zárni, DATA és INPUT esetében a "-jel legtöbbször elhagyható. (Lásd DATA és INPUT.)

Példák:

```
LET a=22.3 : LET b=-32.43e+5
LET c$="ezt idézőjelbe kell tenni"
```

Egy string maximum 254 karakter hosszúságú lehet.

Változók

A változók olyan szimbólumok, amelyekhez a programban különböző értékek rendelhetők.

A változó egy string, amely tetszőleges hosszúságú lehet, ennek alapján lehet hivatkozni egy műveletben a változó értékére. Ez a string tartalmazhat kis- és nagybetűket, számjegyeket és a ?,[,\,],_,. írásjeleket. Az első karakter csak betű lehet.

Egy változó numerikus és string típusú lehet, hasonlóan a konstanshoz. A string típusú változó utolsó karaktere a $, ez az ún. típusazonosító. A numerikus típust nem kell külön azonosítani. Természetesen egy változó típusának és értékének összhangban kell lennie.

Egy stringváltozó hossza maximum 18 karakter lehet, az ennél hosszabb stringeket dimenzionálni kell.

Példák változókra:

X$	string változó
SZÖVEG$	
Eredmény	numerikus változó
C	

Tömbnevek

A tömbnév formailag olyan, mint egy változó, de a tömbnév után zárójelben a tömbelemet is azonosítani kell. (Lásd DIM utasítás.)

Függvények

A függvények lehetnek standard BASIC függvények, vagy ún. felhasználói függvények, amelyeket a programozó definiál. (Részletesen lásd DEF utasítás, ill. a függvények fejezetét.)

Aritmetikai és logikai műveleti operátorok

Egy kifejezésben műveleti operátorokkal lehet kijelölni a műveleteket (pl. összeadás, kivonás stb.). Ha több különböző műveletet kell végezni, akkor a végrehajtás sorrendjét az ún. precedenciaszabály határozza meg. Ez azt jelenti, hogy először mindig a magasabb prioritású műveleteket hajtja végre a BASIC.

Az alábbi táblázat összefoglalja a műveleti operátorokat és megadja az egyes prioritási szinteket:

legmagasabb prioritás	( )	zárójel
	^	hatványozás
	*, /	szorzás, osztás
	+, -	összeadás, kivonás
	=, <>, <, <=, >, >=	egyenlő, nem egyenlő kisebb, kisebb vagy egyenlő, nagyobb, nagyobb vagy egyenlő
	NOT	bitenkénti negáció
	AND	bitenkénti ÉS kapcsolat
legalacsonyabb prioritás	OR, XOR	bitenkénti VAGY kapcsolat és KIZÁRÓ VAGY kapcsolat

Az azonos prioritási szinten lévő műveleteket balról jobbra haladva, sorrendben hajtja végre a BASIC.
Mivel először mindig a zárójelben lévő műveletek kerülnek sorra, ezért érdemes a kifejezéseket zárójelzéssel áttekinthetővé tenni.

Példák a műveletek végrehajtásának sorrendjére:

PRINT 2+3 *5	eredmény: 17
PRINT (2+3)*5	eredmény: 25
PRINT 20-2*9+4/2*3	eredmény: 8
PRINT 2<3	eredmény: -1 (IGAZ)
PRINT 3<2	eredmény: 0 (HAMIS)
PRINT 100 OR (1<2)	eredmény: -1

Perifériák (eszközök)

Egyes input-output utasítások paraméterlistájában megadható egy fizikai egység – periféria –, amelyet az utasítás a művelet során használ. Általában ezt a paramétert akkor érdemes használni, ha az utasításhoz rendelt standard fizikai perifériáktól el akarunk térni.

A perifériákra sorszámmal lehet hivatkozni:

0 – képernyő
1 – billentyűzet
2 – editor
3 – hanggenerátor
4 – párhuzamos nyomtató
5 – kazettás magnetofon
6 – bővítő kártya

A hivatkozás az utasításokban # perifériasorszám alakú.

PARANCSOK ÉS UTASÍTÁSOK

A BASIC nyelvű program utasításait és parancsait a BASIC interpreter értelmezi és dolgozza fel. Ha a begépelt kulcsszó és annak paraméterei előtt nem áll sorszám, akkor azt az interpreter parancsként fogja értelmezni és a kívánt műveletet azonnal végrehajtja. A legtöbb művelet parancsként és utasításként egyaránt végrehajtható.

Az alábbi műveleteket jellemzően parancsként célszerű használni:

NEW
RUN
LIST és LLIST
DELETE
TRACE
CONTINUE

A fejezet ismerteti a parancsokat és az utasításokat. Az utasítások alakjának leírásában a . . . az előző paraméter ismételhetőségére, a [ ] pedig az opcionális (nem kötelező) részekre utal.
A szegmensek listája a parancsokban sorszám-sorszám vagy sorszám alakban megadott programrészt jelent.

NEW

Alakja: NEW

A parancs törli az aktuális programot a memóriából. Amennyiben a program-nyomkövetés előzőleg be volt kapcsolva (TRACE ON), kikapcsolja azt (TRACE OFF).

Példa:

NEW

Parancsként kiadva törli a memóriában lévő programot.

100 IF A$ = "VÉGE" THEN NEW

A programban is lehet alkalmazni a NEW-t.

Ha az A$ változó értéke = "VÉGE", akkor a program kitörli saját magát.

RUN

Alakja: RUN
vagy RUN sorszám

A memóriában kazettáról, vagy diszkről, illetve a billentyűzetről betöltött, vagy beírt program a RUN paranccsal indítható el.
Ha a sorszámot nem adjuk meg, úgy a program végrehajtása a legalacsonyabb sorszámú

utasítássorral kezdődik. A sorszám megadása esetén a kijelölt sor első utasításával indul a program.
Lényeges megjegyezni, hogy a parancs törli a szimbólumtáblában lévő változókat és a függvénydefiníciókat. Ezt különösen akkor kell figyelembe vennünk, amikor a programot nem az elejéről indítjuk el.
A memóriában lévő programot a SAVE utasítással lehet kazettára vagy diszkre felírni. Ha ezt megelőzően az AUTO BASIC változót (5895 decimális cím) 255-re állítjuk be, akkor betöltéskor (LOAD) a program automatikusan elindul, nincs szükség a RUN parancsra.
A RUN utasításként is használható.

LIST

Alakja: LIST
vagy LIST szegmensek listája
vagy LIST periféria:
vagy LIST periféria: szegmensek listája

A program megadott területeiről, vagy a teljes programról készít listát a parancs a képernyőre, vagy a kijelölt perifériára. A CTRL—P gombokat lenyomva a listázás felfüggeszthető, majd bármely gomb megnyomásával folytatható.

Példák:

LIST
Lista a teljes programról.

LIST 50, 100—200
Lista az 50-es sorról és a 100 és 200 közötti sorokról, a határoló sorokat is beleértve.

LLIST

Alakja: LLIST
vagy LLIST szegmensek listája
vagy LLIST periféria:
vagy LLIST periféria: szegmensek listája

Az LLIST parancsot csak abban az esetben érdemes használni, ha nyomtató is van a rendszerben.
Az LLIST parancs segítségével lista készíthető a memóriában lévő teljes programról, vagy annak egy meghatározott részéről a nyomtatóra, vagy az adott perifériára.

Példák:

LLIST
A teljes programról lista készül.

LLIST 100–400, 500
Lista készül a 100 és 400 közötti sorszámú utasítássorokról és az 500-as utasítássorról.

DELETE

Alakja: DELETE szegmensek listája

A programban lévő hibás, vagy feleslegessé vált sorokat a DELETE paranccsal lehet kitörölni a memóriából.

Példák:

DELETE 100
Törli a 100-as sorszámú utasítássort.

DELETE 100–200
A parancs törli az utasítássorokat 100-tól 200-ig.

DELETE 100–200,540,600–
A parancs törli 100-tól 200-ig az összes sort, törli az 540-es sort és 600-tól a program végéig valamennyi sort.

A törlés a CTRL–ESC gombok egyidejű megnyomásával megszakítható.

TRACE

Alakja: TRACE periféria: ON
vagy TRACE periféria: OFF
vagy TRACE ON
vagy TRACE OFF

A TRACE a programbelövés hatékony eszköze. Az ON bekapcsolja a programnyomkövetést; hatására a megadott perifériára kiíródnak a végrehajtott sorok sorszámai. A belövéshez érdemes még a STOP és a PRINT utasításokat is alkalmazni a program kritikus pontjain a változók értékeinek kiíratására.
Periférianév hiányában a sorszámlista a képernyőre kerül.
A nyomkövetést a TRACE OFF parancs kapcsolja ki.

CONTINUE

Alakja: CONTINUE

A CONTINUE parancs segítségével indítható tovább a STOP utasítással, vagy a CTRL–ESC gombokkal megszakított program.
Az alábbi esetekben nem használható a CONTINUE:

ha

- a program szintaktikai vagy szematikai hiba miatt szakadt meg;
- a program STOP utasítás vagy a CTRL–ESC gombok megnyomása miatt állt meg, de utána a programot módosítottuk;
- END vagy DELETE miatt szakadt meg a program.

Lásd: STOP, END, DELETE.

LET

Alakja: LET változó = kifejezés
vagy LET tömbelem = kifejezés
vagy LET rész-string = string kifejezés

A magas szintű nyelvek egyik jellegzetessége, hogy adatokhoz szimbólikus neveket rendelhetünk hozzá, illetve egy szimbólikus névhez a program futása során több különböző adat tartozhat. Ezt a hozzárendelést nevezzük értékadásnak.
A LET utasítással értéket lehet adni egy változónak, tömbelemnek vagy rész-stringnek.

A LET kulcsszó kiírása nem kötelező.

Példák:

```
10 LET A = 625 : LET B = A/5
20 LET KUTYA$ = "PULI"

50 LET ADATTÖMB (2,1) = 3
60 LET NEVEK$ (0,0) = "KISS"
```

Az egyenlőségjel bal oldalán álló változó felveszi a jobb oldalon álló kifejezés értékét. Az olyan többszörös értékadás, mint a

```
LET A = B = C = 2
```

nem megengedett.

Értékadás előtt a tömböket dimenzionálni kell. Lásd DIM.

Az alábbi táblázat példa a rész-stringek értékadására, egyben bemutatja a rész-string képzés szabályait is.

LET A$ = "123456"

rész-string képző m	n	PRINT A$(m:n)	LET A$(m:n)="78" PRINT A$	LET A$(m:n)=" " PRINT A$
3	4	34	127856	1256
2	5	2345	1786	16
3	3	3	1278456	12456
3	2000	3456	1278	12
5	4	1234	7856	56
	3	123	78456	456
4		456	12378	123
1		1	7823456	23456
0		null string	78123456	123456
-10		null string	78123456	123456
6		6	1234578	12345
10		null string	12345678	123456

CLS

Alakja: CLS

A CLS utasítás törli a képernyőt és a karaktermutatót (cursort) a bal felső pozícióba viszi. A teljes képernyőt a SET PAPER utasítással beállított színre festi.

Lásd SET.

PRINT

Alakja: PRINT
vagy PRINT [paraméterek:] [kifejezés] [elválasztójel [kifejezés]] . . .

ahol paraméterek: periféria
vagy/és AT sor, oszlop
vagy/és USING formátumstring
egymástól vesszővel elválasztva.

elválasztójel: , (vessző) vagy ; (pontosvessző) vagy TAB (pozíció)

Az AT és TAB csak a 0, 2, 6 perifériákra hatásos.

A PRINT a TV-Computer BASIC talán legfontosabb utasítása. Egyik funkciója adatok megjelenítése a képernyőn. Ez alapvető fontosságú, hiszen az elvégzett műveletek eredményeit és az ábrákat látni szeretnénk. Az adatok numerikus vagy string típusúak lehetnek, ezeket konstansként és kifejezésként is meg lehet adni.

Példa:

```
PRINT 2*3
PRINT "TV-COMPUTER"
```

A kiírás mindig a cursor által kijelölt pozíción kezdődik. A cursor az AT, a TAB, a "," (vessző) és a ";" (pontosvessző) segítségével pozicionálható.

Az AT után a koordináták megadásával a képernyő bármely pontjára állítható a cursor, természetesen a beállított GRAPHICS üzemmód figyelembe vételével:

2 színű mód:	64 karakter/sor
4 színű mód:	32 karakter/sor
16 színű mód:	16 karakter/sor

A sorok száma mindig 24.

Egy PRINT utasítás több listaelemet is tartalmazhat; ezeket pontosvesszővel, vagy vesszővel kell egymástól elválasztani. A pontosvesszővel elválasztott adatok közvetlenül egymás után lesznek a képernyőre írva, míg a vessző hatására a cursor a következő tabulációs mező elejére áll. A tabulációs mező szélessége 8 karakter. Az elválasztáshoz a TAB(n) is használható, ekkor a következő kifejezés a soron belül az n. pozícióra kerül.

Paraméterek nélküli PRINT a következő sor elejére viszi a cursort.

Példa:

```
10 LET A=5:LET B=2:LET C=A*B:CLS
20 PRINT "Szorzat="; C
30 PRINT "Összeg="; A+B,
40 PRINT "Hányados="; A/B
```

A PRINT utasítás végén is állhat vessző vagy pontosvessző, ez ekkor a következő PRINT vagy INPUT hatására megjelenő karakterek kezdőpozícióját határozza meg. Záró vessző vagy pontosvessző hiányában a következő megjelenítés a következő sor elején kezdődik.

INPUT utáni PRINT mindig a következő sor elején kezdi az írást.

Példa:

```
10 PRINT "Gépeljen be egy szöveget:"; : INPUT X$
20 PRINT "Ismét gépeljen egy szöveget:" : INPUT Y$
```

A második adatbevitel új sorban kezdődik.

A PRINT USING segítségével az adatok meghatározott formátum szerint jeleníthetők meg; ez különösen táblázatok készítésénél lehet fontos.
A formátum egy string típusú kifejezés, amely speciális formátumvezérlő szimbólumokat és betűket, számokat, írásjeleket tartalmazhat.

formátum kifejezés	nyomtatási kép
PRINT USING "###.###":1	__1.000
PRINT USING "###.###^^^^":1	100.000E-02
PRINT USING "$##" :1	_$1
PRINT USING "+## Ft":1	_+1_Ft
PRINT USING "-##":1	__1
PRINT USING "*###.###":1	***1.000
PRINT USING "%###.###":1	0001.000
PRINT USING "N=## K=##":1,2	N=_1_K= 2
PRINT USING "##,###;###":1E7	_1,000,000
PRINT USING "*##":1;2	**1**2
PRINT USING "*##":1,2	**1 (tab) **2
PRINT USING "<######":"BASIC"	BASIC__
PRINT USING ">######":"BASIC"	__BASIC
PRINT USING "#######":"BASIC"	_BASIC_

A PRINT utasítás másik fő funkciója segítségével adatokat lehet vinni háttértárolóra; kazettára vagy diszkre. Írás előtt a file-t az OPEN utasítással meg kell nyitni, majd a PRINT utolsó alakja alkalmazható az adatok mentésére. Az összes adat felírása után a file-t a CLOSE utasítással le kell zárni.

Példa:

```
10 OPEN #5:OUTPUT "ADAT"
20 PRINT #5:A,B,C$
30 CLOSE #5:OUTPUT
```

Lásd GRAPHICS, OPEN, CLOSE.

LPRINT

Alakja: LPRINT
vagy LPRINT [paraméterek:] [kifejezés] [elválasztójel [kifejezés]] . . .
ahol paraméterek: periféria
vagy/és AT sor, oszlop
vagy/és USING formátumstring
egymástól vesszővel elválasztva.
elválasztójel: , (vessző) vagy ; (pontosvessző) vagy TAB (pozíció)

Az AT és a TAB csak a 0, 2, 6 perifériákra hatásos.

A LPRINT utasítás funkciója azonos a PRINT-tel. Az eltérés csupán annyi, hogy periféria megadása nélkül LPRINT esetén a megjelenítés a nyomtatón történik.

Ennek megfelelően a PRINT #4: . . . és az LPRINT . . ., valamint,
a PRINT . . . . és az LPRINT #2: . . . hatása azonos.

Lásd PRINT.

INPUT

Alakja: INPUT
vagy INPUT változó [, változó . . .]
vagy INPUT PROMPT string kifejezés : változó [, változó . . .]
vagy INPUT periféria: változó [, változó . . .]

Az INPUT utasítás felhasználásával adatok gépelhetők be a billentyűzetről, vagy olvashatók be a háttértárolóról (pl. a kazettás magnetofonról) az aktuálisan megnyitott file-ból. A bevitt adatokat a változók veszik fel értékül. Az adatok formátumát a változók típusa határozza meg. Numerikus változó esetén a számjegyek, a + és - előjelek, a tizedespont és az exponenst jelző E karakter használható.
Stringek esetén az utasítás a 32 – 223 kódú karaktereket fogadja el. A stringeket csak akkor kell " jelek között megadni, ha bevezető szóköz, felkiáltójel vagy vessző is előfordul benne.
Helytelen formátumú adat megadása nem okoz hibajelzést, de a megfelelő numerikus változó 0, string változó pedig " " (üres string) értéket kap.

Alapértelmezésben – a periféria megadása nélkül – az INPUT a billentyűzetről várja az adatok begépelését, amelyet a RETURN gomb megnyomása zár le.
A CTRL–ESC gombokkal megszakítható az utasítás végrehajtása és a CONTINUE-val indítható újra.

Ha egy változó nem kap értéket – ez abban az esetben fordul elő, ha a listában több változó szerepel, mint az adatok száma –, akkor a numerikus változók 0, a string változók " " (üres string) értéket kapnak.

A PROMPT szó után egy szöveg adható meg, amely kiíródik a képernyőre. Változó megadása nélkül az INPUT csak a RETURN megnyomására vár, értékadás nem történik.

Példa:

1. 100 INPUT PROMPT "DATUM (EV, HONAP, NAP):":E,H$,N
   110 PRINT E,H$,N

2. 10 OPEN #5:"PROGRAM"
   20 INPUT #5:A$,B$
   30 CLOSE #5:INPUT

Lásd OPEN, CLOSE.

INKEY$ (függvény)

Alakja: INKEY$

A függvény a klaviatúrán utoljára leütött, de még be nem olvasott karaktert veszi fel értékül. Ha nincs karakter, akkor ez az érték a " " (üres string).

Az INPUT és az INKEY$ függvény közti különbségek:
- Az INKEY$ egyetlen karaktert ad vissza, numerikus értéket vagy hosszabb stringet nem.
- Az INKEY$-ral beolvasott karakter nem jelenik meg a képernyőn.
- Az INKEY$ nem várakozik a beírásra, hanem a következő utasítással azonnal folytatódik a program. Az INPUT utasítás mindaddig várakozik, míg a kezelő RETURN-nel nem jelezte a bevitel végét.
- Az INKEY$ minden karakterkódot elfogad.

Példa:

```
10 PRINT "Várjon egy kicsit"
20 FOR N=1 TO 20:NEXT N
30 PRINT "Most nyomjon meg egy gombot"
40 AS=INKEY$:IF A$="" "THEN 40
50 PRINT "Ezt gépelte be:"; A$
```

A 40. sorban várakozik a program a begépelésre.

Lásd: INPUT.

GET

Alakja: GET
vagy GET string-változó
vagy GET periféria: string-változó

A GET utasítással a kijelölt perifériáról, vagy az előzőleg OPEN-nel megnyitott file-ból egy karaktert lehet beolvasni. A változó vagy tömbelem ezt felveszi értékül. A file végén a " " (üres string) értékkel tér vissza.

Az implicit periféria a billentyűzet (1. sorszámú periféria). Billentyűzet olvasása esetén — ellentétben az INPUT utasítással — a GET beolvassa az ASCII kódtábla 0 . . . 255 kódú karaktereit.

Példa:

```
GET          várakozik egy gomb megnyomására a klaviatúrán.
GET A$       egy karakter olvasása a klaviatúráról.
GET #6:A$    olvasás az előzőleg megnyitott file-ból.
```

REM

Alakja: REM megjegyzés szövege

A REM megjegyzés, magyarázó szöveg elhelyezését teszi lehetővé a program tetszőleges helyén. A szöveg csak listázáskor jelenik meg, a programban a BASIC átlátszó utasításnak tekinti a REM-et.

Példa:

```
10 REM A ciklusváltozó értékének kiíratása
20 FOR A=0 TO 100
30 PRINT A
40 NEXT A
```

IF – THEN – ELSE

Alakja: IF feltétel THEN utasítás(ok) vagy sorszám
vagy IF feltétel THEN utasítás(ok) vagy sorszám: ELSE utasítás(ok) vagy sorszám

Az IF feltételes vezérlésátadó utasítás. A feltételt egy kifejezéssel adhatjuk meg, melynek értékét a BASIC kiszámítja. Az eredmény lehet IGAZ (TRUE= -1), vagy HAMIS (FALSE = 0). Ha a feltétel IGAZ, akkor a THEN utáni utasításokat végrehajtja a BASIC. Ha az eredmény HAMIS, úgy az ELSE ág utasításai következnek. Ha nincs ELSE ág, akkor az IF-et követő utasításssoron folytatódik a program.

A THEN utáni utasítás ne legyen újabb IF – THEN – ELSE, mert egymásba ágyazás esetén csak az első ELSE hatásos.

Példa:

```
10 INPUT PROMPT "Gépeljen be egy pozitív számot":A
20 IF A<0 THEN PRINT "Ez negatív szám":GOTO 10
30 PRINT "Köszönöm"
```

Lásd ON.

FOR

Alakja: FOR ciklusváltozó=kezdőérték TO végérték
vagy FOR ciklusváltozó=kezdőérték TO végérték STEP lépésköz

A ciklus fogalmának és szükségességének jobb megértése miatt nézzük meg az alábbi kis példát:

Írassuk ki az egész számokat 10-től kezdve 40-ig tízesével. Az eddigi ismeretek alapján a megoldás a következő lehet:

```
10 A=10:PRINT A
20 A=A+10:PRINT A
30 A=A+10:PRINT A
40 A=A+10:PRINT A
```

vagy

```
10 A=10
20 PRINT A
30 A=A+10
40 IF A>40 THEN STOP
50 GOTO 20
```

Ugyanez ciklus szervezésével:

```
10 FOR A=10 TO 40 STEP 10
20 PRINT A
30 NEXT A
```

Azokat az utasításokat érdemes ciklusba szerveznünk, amelyeket egymás után sokszor kell végrehajtani és a végrehajtásuk száma előre megadható. A ciklus kezdetét a FOR, a végét a NEXT utasítás jelzi.

A ciklus végrehajtását a ciklusváltozó értéke szabályozza. A ciklusváltozó értéke a kezdőértéktől a végértékig a lépésközként megadott lépésekben változik, és minden érték mellett egyszer végrehajtódik a NEXT-ig terjedő utasítássorozat. A FOR utasításban a kezdő- és végérték, valamint a lépésköz numerikus kifejezésekkel adható meg.
A megadott értékektől függetlenül a ciklus legalább egyszer lefut.

Ha nem adunk meg lépésközt, annak értéke automatikusan +1 lesz. A lépésköz negatív is lehet.

Lásd NEXT.

NEXT

Alakja: NEXT
vagy NEXT változó [, változó . . .]

A ciklust a NEXT utasítás zárja le. A NEXT végrehajtásakor a ciklusváltozó aktuális értékéhez hozzáadódik a STEP után megadott érték. Ha az eredmény nagyobb, mint a FOR utasításban megadott végérték, akkor a NEXT utáni utasításon folytatódik a program. Ha az eredmény kisebb, vagy egyenlő, mint a ciklus végértéke, akkor a ciklus a FOR utasítástól kezdve ismét végrehajtódik.
Egy cikluson belül újabb ciklusok szervezhetők, szükség szerint. Több, egymásba ágyazott ciklust egy NEXT is lezárhat, ha a paraméterlistában helyes sorrendben szerepelnek a ciklusváltozók. Ekkor egy belső ciklus befejeződése után a program nem a NEXT utáni utasítással, hanem a NEXT-ben levő következő ciklusváltozó, azaz a következő ciklus kiértékelésével folytatódik.
A ciklusból vezérlésátadó utasítással ki lehet lépni.
A ciklusmagban a ciklusváltozó értéke tetszés szerint módosítható, de ez természetesen befolyásolja a ciklus működését.

Példa:

```
10 INPUT N
20 FOR I=1 TO 5
30 FOR J=1 TO 10
40 PRINT I,J,I*J
50 IF I>N THEN 80
60 NEXT J,I
70 STOP
80 PRINT N:END
```

Lásd FOR, IF.

DATA

Alakja: DATA konstans [, konstans . . .]

A DATA utasítással numerikus vagy string típusú konstansok helyezhetők el a BASIC programon belül. Különösen akkor érdemes alkalmaznunk ezt az utasítást, ha ugyanazt a konstansot — vagy konstansokat — a programban többször, különböző helyeken akarjuk használni. A DATA utasítás a programon belül bárhol elhelyezhető, és tetszőleges számú DATA lehet a programban. Több ilyen utasítás esetén a bennük foglalt konstansok logikailag összefüggő láncot alkotnak (lásd READ). A string típusú konstansot "-ek között kell megadni abban az esetben, ha a konstans tartalmazza a vessző, a felkiáltójel vagy a kettőspont karaktereket, illetve szóközzel kezdődik. Egyébként az " megadása nem szükséges.

Lásd READ, RESTORE.

READ

Alakja: READ változó [, változó . . .]

A DATA utasításokkal a programban elhelyezett adatokat a READ utasítással lehet kiolvasni. A READ paraméterlistájában szereplő változók sorban felveszik értékül az adatterületen elhelyezkedő konstansokat.

Ne feledjük, hogy a több DATA-val elhelyezett konstansok láncot alkotnak! A változó és a konstans típusának meg kell egyeznie!

Példa:

```
10 READ A,B
20 READ C,V$
30 READ D
.
.
.
100 DATA 1,2,3
110 DATA "ABC"
120 DATA 4,5,6
```

A kiolvasás után a változók az alábbi értékeket kapják:

A=1, B=2, C=3, V$="ABC", D=4

Az adatterületen a BASIC egy mutatót kezel, amely megmondja, hogy a következő READ utasításnak melyik adattól kell kezdenie az olvasást. Egy adat kiolvasása eggyel jobbra, előre mozdítja a mutatót.

Ha a READ már valamennyi adatot kiolvasta és ezután újabb olvasást kísérelünk meg, akkor a BASIC hibaüzenetet küld: "*** No DATA". Ezt a problémát a RESTORE utasítás használatával oldhatjuk meg.

Lásd DATA, RESTORE.

RESTORE

Alakja: RESTORE
vagy RESTORE sorszám

Az utasítás segítségével az adatterületen levő mutató – lásd READ – visszaállítható a terület elejére, a programban levő első konstansra, vagy a megadott sorszámú DATA konstans-lista elejére. Így elérhető, hogy ugyanazokat az adatokat többször is kiolvashassuk, szükség esetén különböző sorrendben.
A megadott sorszámon DATA utasításnak kell lennie.

Példa:

```
10 READ A,B
20 READ C$
30 RESTORE
40 READ D,E,F$,G
50 RESTORE 120
60 READ F
```

```
100 DATA 1,2
110 DATA ABC
120 DATA 3
130 DATA 4
```

A kiolvasás utáni értékek:

A=1, B=2, C$="ABC", D=1, E=2, F$="ABC", G=3, F=3

Lásd DATA, READ.

STOP

Alakja: STOP

A STOP utasítás megszakítja a program végrehajtását és a BASIC parancs módba tér át. A program kritikus pontjain elhelyezve hatékonyan segíti a tesztelést és programbelövést. A megszakítás helyéről a program a CONTINUE parancs begépelésével indítható tovább, a CONTINUE parancsnál felsorolt esetek kivételével.

Lásd CONTINUE.

END

Alakja: END

Az END utasítás a program logikai végét jelzi. Hatására az utasítások végrehajtása befejeződik és a BASIC parancs módba tér át.

Ha a program logikai vége megegyezik a fizikai végével, akkor nem szükséges az END-et használni.

DIM

Alakja: DIM változó(deklaráció) [, változó(deklaráció) . . .]
vagy DIM stringváltozó [(deklaráció)] * elem-hossz

ahol deklaráció: dimenzió mérete [, dimenzió mérete . . .]

A DIM utasítás tömböknek foglal helyet a memóriában. A tömb az azonos típusú adatok rendezett halmaza, amely adatok numerikus vagy karakterlánc típusúak lehetnek. Egy aritmetikai vagy string műveletben a tömb egy elemére hivatkozhatunk az elem tömbben elfoglalt helyének pontos meghatározásával. Bármely dimenzió első eleme a 0 sorszámú elem.

String típusú változó és tömb dimenzionálásakor meghatározható a változó(k) mérete az "elem-hossz" paraméter megadásával 1 és 254 karakter között. Alaphelyzetben ez az érték 18.

Elvileg egy tömbnek tetszőleges számú dimenziója lehet, gyakorlatban a DIM utasítás hosszának határt szab az utasítássor hossza (250 karakter). A DIM utasítás végrehajtásakor a numerikus tömb elemei 0 értéket kapnak, a karakterlánc típusú tömb elemei a " " (üres karakterlánc, 0 hossz) értéket veszik fel. Ha olyan tömböt dimenzionálunk, amely előzőleg már létezett, akkor hibajelzést kapunk: *** Variable declared twice.

Példa.

```
10 DIM T (4,4)
20 FOR I=0 TO 4
30 FOR J=0 TO 4
40 READ T(I,J):PRINT T(I,J)
50 NEXT J,I
60 DATA 1,2,3,4,5,6,7,8,9,10,11,12,13
70 DATA 14,15,16,17,18,19,20,21,22,23,24,25
```

GOTO

Alakja: GOTO sorszám

A GOTO utasítás feltétel nélküli ugrást — vezérlésátadást — hajt végre a megadott sorszámú utasítássorra, ellentétben az IF-fel, amelynél egy feltétel teljesülése esetén jött létre a vezérlésátadás. A megadott sorszámú utasítássornak léteznie kell a programban.

Törekednünk kell arra, hogy egy programban minél kevesebb GOTO utasítás legyen. Ennek két oka van:

— A BASIC interpreter belső felépítése és működése miatt a GOTO végrehajtása jelentős időt vesz igénybe. Ezért ahol lehet, inkább szubrutinhívást célszerű használni, két GOTO helyett.
— Minél kevesebb a GOTO, annál áttekinthetőbb lesz a program, ami javításnál, módosításnál lényeges szempont.

Lásd IF, ON.

GOSUB

Alakja: GOSUB sorszám

A programírás során gyakran találkozhatunk azzal a problémával, hogy a program különböző helyein ugyanazokat az utasítássorozatokat kell leírnunk. Ezt a problémát szubrutin — alprogram — szervezésével oldhatjuk meg. A szubrutin olyan utasítássorozat, mely a program különböző pontjairól végrehajtatható a GOSUB utasítással.

A hívó program a változók aktuális értékét adja át paraméterként a szubrutinnak, így a végrehajtás eredménye minden híváskor más és más lehet.

A szubrutinból újabb szubrutin hívható, az egymásba ágyazás mélysége tetszőleges. Egy szubrutin önmagát is hívhatja (rekurzív hívás). A rekurzív hívási lánc befejeződéséről természetesen gondoskodni kell, mivel különben végtelen ciklus hiba keletkezik a programban. A gyakorlatban tehát a rekurzív hívást mindig valamilyen feltételvizsgálat és elágazás (IF) előzi meg.

Egy szubrutin a RETURN utasítással ér véget. Ha hívás, azaz megelőző GOSUB nélkül erre az utasításra fut rá a program, az hibajelzést okoz.

Példa:

```
 10 REM Az elrejtett számot kell kitalálnia!
 20 INPUT A:CLS
 30 INPUT PROMPT "Kérem a számot":B
 40 IF A=B THEN PRINT "Kitalálta":STOP
 50 IF A<B THEN PRINT "Ennél kisebb.": GOSUB 100:GOTO 30
 60 IF A>B THEN PRINT "Ennél nagyobb.": GOSUB 100:GOTO 30
100 PRINT "Próbálja újra!"
110 RETURN
```

Lásd RETURN.

RETURN

Alakja: RETURN

Egy szubrutin logikailag utolsó utasítása a RETURN. Hatására a program végrehajtása az adott szubrutint hívó GOSUB utáni utasításon folytatódik.

Lásd GOSUB.

ON

Alakja: ON kifejezés GOTO sorszám1 [,sorszám2 . . .] :ELSE utasítás vagy sorszám
vagy ON kifejezés GOTO sorszám1 [,sorszám2 . . .]
vagy ON kifejezés GOSUB sorszám1 [,sorszám2 . . .] :ELSE utasítás vagy sorszám
vagy ON kifejezés GOSUB sorszám1 [,sorszám2 . . .]

Az ON speciális vezérlésátadó utasítás. A kifejezés értéke egy mutató, amely rámutat a sorszámok listájának egy elemére és a program végrehajtása ezen a sorszámon fog folytatódni. Azaz, ha a kifejezés értéke 1, akkor az utasítás hatása GOTO (vagy GOSUB) sorszám1; ha az értéke 2, akkor a hatás GOTO (vagy GOSUB) sorszám2, stb.

Ha a kifejezés értéke nagyobb, mint a listaelemek száma, vagy a kifejezés 0 értékű, akkor az ELSE utáni utasítással folytatódik a program; ELSE ág hiányában az ON-t követő utasításon.

Példa:

```
10 INPUT A,B
20 PRINT "Műveletek:"
30 PRINT "1.Összeadás, 2.Kivonás, 3.Szorzás"
40 INPUT PROMPT "Művelet:":C
50 ON C GOTO 60,70,80:ELSE PRINT "Hibás sorszám:":GOTO 40
60 PRINT A+B:GOTO 10
70 PRINT A-B:GOTO 10
80 PRINT A*B:GOTO 10
```

Lásd GOTO, GOSUB.

GRAPHICS

Alakja: GRAPHICS kifejezés

A GRAPHICS utasítással a három grafikus üzemmód valamelyikét lehet kiválasztani; meghatározza az alkalmazható színek, az egy sorba írható karakterek és a grafikus képpontok számát. Az alábbi táblázat összefoglalja ezeket a jellemzőket:

Üzemmód	Karakterek száma	Grafikus képpontok száma
2 színű	64/sor	512/sor
4 színű	32/sor	256/sor
16 színű	16/sor	128/sor

Mindhárom üzemmódban a karaktersorok száma 24, a függőleges irányban levő grafikus képpontok száma 240. A színek természetesen csak megfelelő színes televízión, vagy monitoron jelennek meg, egyébként csak a szürke szín különböző árnyalatai láthatók.

Alaphelyzetben a TV-Computer a négy színű üzemmódban működik. Új üzemmód beállításakor a GRAPHICS utasítás alaphelyzetbe állítja a színeket és törli a képernyőt.

A két- és négyszínű módban a SET PALETTE utasítással lehet kiválasztani a színpalettából az aktuális színeket.

PLOT

Alakja: PLOT x-koordináta, y-koordináta tollvezérlő . . .
vagy PLOT x-koordináta, y-koordináta tollvezérlő . . . ,PAINT

ahol tollvezérlő: , (vessző) vagy ; (pontosvessző)

A TV Computer képes nagy felbontású színes grafikus pontokból kialakított, bonyolult ábrák megjelenítésére a hozzákapcsolt tv-készülék vagy monitor képernyőjén. A PLOT utasítás logikai koordinátákat használ; a képernyőt minden üzemmódban 960 * 1024 képpontra osztja. Az utasítás a logikai koordinátákból az üzemmód figyelembevételével kiszámítja a fizikai koordinátákat, azaz előállítja a 240*512, a 240*256, vagy a 240*128 képpontú képet. Ez a megoldás biztosítja, hogy a PLOT utasításban használt képpont koordináta üzemmódtól függetlenül mindig ugyanazt a helyet jelöli.

A képernyő sarokpontjainak koordinátái:

bal alsó:	0, 0
bal felső:	0, 959
jobb alsó:	1023, 0
jobb felső:	1023, 959

A PLOT utasítás egy "tollat" kezel, amellyel pontokat vagy különböző típusú vonalakat lehet rajzolni. Az x és az y koordinátákkal határozható meg a toll pozíciója. Az utasításban egy vagy több x-y koordinátapár adható meg. Több koordinátapár esetén ezek vesszővel, vagy pontosvesszővel választhatók el. A vessző hatására a képzeletbeli toll felemelkedik és ilyenkor új pozícióra állítható, anélkül, hogy vonalat húzna.
A pontosvesszőre a toll leereszkedik és a kezdő és a végpozíció között vonalat húz.

A rajzolás a SET INK utasítással beállított tintaszínnel és a SET STYLE utasítással adott vonaltípussal történik.

Példák:

```
PLOT,
          a toll felemelkedik az aktuális pozíción
PLOT 0,0;100,200,100,250
       !     !
    kezdő  vég
  pozícióval vonal rajzolása, majd a toll felemelkedik, és a 100,250 pozícióra áll.
```

Lehetőség van zárt alakzatok kifestésére is. A tollat a zárt alakzat belsejébe kell vinni, majd a PAINT paraméter hatására az utasítás az alakzatot az aktuális tinta színére festi.

Példa:

```
 1 REM sinusgörbe rajzolása
10 GRAPHICS 4
20 SET PAPER 0 : SET BORDER 4 : SET INK 3
30 PLOT 0,120,255,120
40 PLOT 19,239;19,0
50 SET INK 1
60 FOR I=0 TO 2*PI STEP 0.02
70 PLOT 19+(30*I),120+(120*SIN(I));
80 NEXT I
90 LIST
```

SET

Alakja: SET paraméterek
a PLOT-tal

A SET a TV-Computer BASIC összetett utasítása. Segítségével beállíthatók az aktuális színek, a PLOT-tal rajzolt vonal típusa, és definiálni lehet felhasználói karaktereket. A paraméterek a következők lehetnek:

PALETTE palettakód0,palettakód1,palettakód2, palettakód3 (négyszínű mód)
PALETTE palettakód0,palettakód1 (kétszínű mód)

A GRAPHICS utasítással beállított üzemmód aktuális színeit választja ki a színpalettából. Csak a két- és a négyszínű üzemmódban hatásos. Az utasítás meghatározza, hogy milyen szín tartozzon a 0,1,2,3; illetve a 0,1 színsorszámokhoz.

A palettakódokat a Függelékben levő táblázat foglalja össze.

SET PALETTE 0,80,81,84 (fekete, zöld, cián, sárga)

INK színsorszám

Az aktuális tintaszínt definiálja a PLOT utasítás számára. Valamennyi színsorszám megengedett, de az aktuális tintaszínt négyszínű módban a színsorszám néggyel, kétszínű módban pedig kettővel történő osztása utáni maradék határozza meg. Így például négyszínű módban a (0,4,8,12), az (1,5,9,13), a (2,6,10,14), és a (3,7,11,15) színsorszámok csoportonként azonos színt állítanak be.

SET INK 2 (a 2-es színsorszámhoz rendelt színnel rajzol a toll)

PAPER színsorszám

A háttér (papír) színét állítja be a színsorszám megadásával. Az aktuális GRAPHICS üzemmódtól függ, hogy hány színből áll a választék. A színsorszám használatára vonatkozó szabályok azonosan az INK-nél leírtakkal.

SET PAPER 14 (a 14-es színsorszámhoz rendelt szín lesz a háttérszín)

BORDER palettakód

A képernyő keretszínét határozza meg. Valamennyi színből lehet választani, függetlenül a beállított GRAPHICS üzemmódtól.

SET BORDER 68 (vörös színű keret)

STYLE vonaltípus-sorszám

A vonaltípust választja ki a PLOT utasítás számára. A táblázat, amely a vonaltípusokat tartalmazza, a Függelékben található.

SET STYLE 1 (folyamatos vonal)

MODE sorszám

Két különböző színű, azonos képpontba kerülő grafikus pont eredő színét határozza meg. Az eredő színt az egymásra kerülő képpontok színeit definiáló palettakódokkal végzett logikai művelet eredménye alakítja ki. A logikai művelet kiértékelése bitenként történik. A logikai műveletet a sorszám határozza meg az alábbiak szerint:

sorszám	művelet	
1	OR	VAGY művelet
2	AND	ÉS művelet
3	XOR	KIZÁRÓ VAGY művelet

Ha a sorszám 0, akkor kereszteződéskor az új szín felülírja az előzőt.

SET MODE 1 (VAGY kapcsolat a színek között)

CHARACTER ASCII-kód, definíció

Felhasználói karaktereket definiál. A karakter képe egy 10 * 8-as pontmátrixból áll, a látható pont értéke 1, a nem láthatóé 0. Tehát a definíció a 10 sornak megfelelően 10 db bináris szám. Az utasításban ezeket decimális számokká alakítva kell megadni.

karakterkép	bináris kód	decimális kód
********	11111111	255
** **	11000011	195
** **	11000011	195
********	11111111	255
**	00011000	24
**	00011000	24
**	00011000	24
****	00011110	30
****	00011110	30
*	00010000	16

SET CHARACTER 129,255,195,195,255,24,24,24,30,30,16

Az ábra a PRINT CHR$(129) utasítással jeleníthető meg.

RATE időállandó

A billentyűzet auto-repeat funkcióját időzíti; meghatározza, hogy folyamatosan nyomva tartott billentyű esetén milyen időközönként kövessék egymást az azonos karakterek.
Az ismétlődés ideje = időállandó / 50 sec.

SET RATE 50 (másodpercenként ismétlődő karakterek)

DELAY időállandó

Az előzőekben ismertetett auto-repeat funkció elindítása előtti késleltetési időt határozza meg.

A késleltetési idő = időállandó / 50 sec.

SET DELAY 300 (A billentyű lenyomása után 6 másodpercig nincs karakterismétlés)

DEF

Alakja: DEF függvénynév (paraméter) = függvénydefinició

A standard BASIC függvények készlete bővíthető saját felhasználói függvényekkel, ezek definiálására szolgál a DEF utasítás. A felhasználói függvény numerikus és string típusú lehet; neve formailag megegyezik egy változó nevével.
A függvény definíció egy értékadás, melynek jobb oldalán tetszőleges – már korábban definiált felhasználói, illetve standard BASIC – függvények és műveletek használhatók.

A paraméter azt a formális változót jelöli, amely az egyenlőségjel jobb oldalán változóként szerepel. Ez a változó a programban máshol, más célra szabadon felhasználható. A felhasználói függvényt első hívása előtt definiálni kell. A definícióban nem szerepelhet a függvény maga (nincsenek rekurziv függvények).

Példa:

```
10 DEF FNC(Y)=SIN(2*Y+Y/3)
20 FOR A=1 TO 5
30 PRINT FNC(A)
40 NEXT A
```

SOUND

Alakja: SOUND ; hangparaméterek . . .
vagy SOUND hangparaméterek . . .

ahol hangparaméterek: PITCH kifejezés,
és/vagy VOLUME kifejezés,
és/vagy DURATION kifejezés

Az utasítás megadott hangmagasságú, időtartamú és erősségű hangot állít elő. A pontosvessző az utasításban azt jelzi, hogy az előző, még folyamatban levő hang képzését be kell fejezni, mielőtt az új hang generálását megkezdenénk.

Példa:

SOUND ; egy hangjelet állít elő, miután az előző hanggeneráló utasítás hatása befejeződött.

SOUND az előző hang megszakad és a hangjelzés azonnal érvényre jut.

Az utasítás paramétereinek implicit értéke:

PITCH	3349	(középső C hang; 261 Hz)
VOLUME	7	(a maximális hangerő fele)
DURATION	100	(a hang időtartama 2 sec)

Egyetlen SOUND utasításban több hang előállítását is előírhatjuk a paraméterek ismétlésével. Ha valamelyik paraméter nem szerepel, akkor a hiányzó paraméter helyett az előző hang adott paraméterét használja az utasítás.

A paraméterként adott PITCH (hangmagasság) és a hang frekvenciája között az alábbi összefüggés teremt kapcsolatot:

195312.5 / (4096 - p) ahol a p az utasításban szereplő PITCH paraméter, az eredményt pedig Hz-ben kapjuk.

Ha a p=4095, akkor nem keletkezik hang. Ezzel a módszerrel lehet egy dallam előállítása közben szünetet beiktatni.

Az időtartam (DURATION) értékét 1 / 50 sec-ban kell megadni.

A hangerő (VOLUME) megadásánál a 0 érték a teljes csendet, a 15 pedig a maximumot jelöli.

A paraméterek lehetséges értékei:

PITCH	0 . . 4094	(kb. 97656.25 Hz) és 4095 (szünet)
DURATION	0 . . 255	(kb. 5 sec)
VOLUME	0 . . 15	

Példa:

```
 5 SOUND                          ! hangjelzés
10 FOR P=1 TO 4000 STEP 10
20 SOUND PITCH P
30 NEXT                          ! gyors skála

100 FOR P=1 TO 4000 STEP 10
110 SOUND ; DURATION 10, PITCH P
120 NEXT                         ! lassú skála
```

A SOUND utasítás PITCH paramétere és a zenei hangok közötti kapcsolatot a Kezelési Útmutató Függelékében levő táblázat tartalmazza. A táblázat felhasználásával viszonylag egyszerűen programozhatók kottájukkal adott dallamok.

OPEN

Alakja: OPEN file-név
vagy OPEN i-o irány file-név

vagy OPEN periféria: i-o irány file-név

ahol: i-o irány: INPUT vagy OUTPUT

Az adatfile-okat írás vagy olvasás előtt meg kell nyitni. Az OPEN utasítás megnyit egy file-t írásra (OUTPUT), illetve olvasásra (INPUT). Ha az i-o irány nincs megadva, akkor a megnyitás olvasásra történik.

A periféria-sorszámot csak abban az esetben kell megadni, ha a megnyitandó file nem az 5-ös eszközön (kazetta) található.

Az INPUT irányra megnyitott file-ból INPUT vagy GET utasítással lehet olvasni, az OUTPUT irányra nyitott file-ba pedig PRINT utasítással lehet írni.

Példa:

```
 10 OPEN OUTPUT "ADATOK"
 20 FOR I=1 TO 10
 30 PRINT#5:I
 40 NEXT
 50 CLOSE OUTPUT
100 OPEN "ADATOK"
110 FOR I=1 TO 10
120 INPUT#5:A
130 PRINT A
140 NEXT
150 CLOSE
```

CLOSE

Alakja: CLOSE

vagy CLOSE i-o irány

vagy CLOSE periféria: i-o irány

ahol i-o irány: INPUT vagy OUTPUT

Egy korábban OPEN-nel megnyitott file-t zár be.

Példák:

CLOSE	kazetta input zárás
10 CLOSE INPUT	ugyanaz
20 CLOSE OUTPUT	kazetta output zárás
CLOSE #6	6-os periféria input zárás
100 CLOSE #6:OUTPUT	6-os periféria output zárás

LOAD

Alakja: LOAD
vagy LOAD file-név
vagy LOAD periféria: file-név

Az utasítás kazettáról, vagy diszkről, illetve a bővítő kártyáról egy programot tölt be a memóriába. A programnév maximum 16 karakter hosszúságú lehet. Ha a programnév nem adott, akkor az elsőként megtalált programot tölti be. Töltés előtt a memóriában levő program és a szimbólumtábla törlődik. Az implicit periféria a kazetta/diszk. Ha az utasításban nemlétező periféria szerepel, akkor a rendszer hibajelzést ad, és minden nyitott file-t lezár.

Példák:

LOAD "PROGRAM"	keresés a kazettán és a PROGRAM betöltése.
LOAD #6:"JÁTÉK"	keresés a 6-os periférián és a JÁTÉK betöltése.
LOAD	az első programfile betöltése a kazettáról.

SAVE

Alakja: SAVE filenév
vagy SAVE periféria: filenév

Az utasítás a memóriában levő teljes programot kazettára, diszkre vagy a bővítő egységre menti. A tárolás úgynevezett belső formátum szerinti; az így tárolt programok LOAD utasítással visszatölthetők.

Az implicit periféria a kazetta/diszk.

Ha az utasítás paraméterezése hibás, akkor hibajelzés után a rendszer minden nyitott file-t lezár.

Példák:

SAVE "PROGRAM"
SAVE #6:"JÁTÉK"

VERIFY

Alakja: VERIFY
vagy VERIFY file-név
vagy VERIFY periféria: file-név

Az utasítás ellenőrzi, hogy egy program helyesen lett-e elmentve egy korábbi SAVE utasítással; a memóriában levő programot hasonlítja össze a megadott file tartalmával.

Célszerű minden programmentés után ellenőrizni — még a program memóriából való kitörlése előtt — a mentés hibátlanságát. Így elkerülhető az esetleges újrabegépelés vagy a fáradságos hibakeresés és javítás.

Példák:

VERIFY az első megtalált programot összehasonlítja a memória tartalmával.
VERIFY "PROGRAM"
VERIFY #6:"JÁTÉK"

EXT

Alakja: EXT kifejezés1,kifejezés2, kifejezés3,kifejezés4

Az utasítás egy felhasználói gépi kódú szubrutin hívását teszi lehetővé. A paraméterként megadott első kifejezés a szubrutin sorszámát határozza meg 0 és 6 között. A további, legfeljebb három kifejezés pedig a processzor HL, DE, és BC regiszterpárjaiba átírandó értékeket jelenti. A gépi kódú szubrutin a HL, DE, BC, AF, DE', AF' és C' regisztereket szabadon használhatja.

Felhasználhatók — nagyon óvatosan — a következő információk is:
IX a BASIC változóterületére mutat,
IY a BASIC stack utolsó byte-jára mutat,
HL' a forrásprogram következő utasítására mutat.

A meghívott gépi kódú szubrutint RET utasítással kell befejezni. Az EXT utasítással hívott rutin nem tud paramétert visszaadni.

A gépi kódú szubrutin kezdőcímet a hívás előtt a USRTAB (címe: 33 decimális) táblázatban kell elhelyezni. A táblázat hét cím tárolására képes, minden kezdőcím két byte-on helyezkedik el. Az első két byte az EXT 0, a második két byte az EXT 1 . . . stb. kezdőcíme. Ha a táblázat 0 címet tartalmaz, akkor a hozzátartozó EXT utasítást a BASIC üres utasításként értelmezi.

LOMEM

Alakja: LOMEM kifejezés

Az utasítás a program memóriabeli áthelyezésére szolgál; a BASIC program kezdőcíme a kifejezéssel megadott kezdőcímre kerül. Az áthelyezés után felszabaduló memóriaterület gépi kódú szubrutinok elhelyezését teszi lehetővé, úgy, hogy azok a BASIC program elé kerülnek. A LOMEM utasítás minden változó értékét törli, ezért csak a programok elején célszerű a használata.

Ha a kifejezés a normál program-kezdőcímnél, azaz a standard kezdőértéknél kisebb vagy 32767-nél nagyobb, akkor hibajelzés keletkezik.

A NEW és a LOAD utasítások a LOMEM értékét visszaállítják a standard értékre, ezért az előzőleg elhelyezett gépi kódú szubrutinok elvesznek.

A LOMEM standard értékét a VLOMEM rendszerváltozó (címe: 5920 decimális) tartalmazza. Ha ennek a változónak az értékét megváltoztatjuk (pl. POKE utasítással), akkor NEW és LOAD után ezen a módosított címen fog kezdődni a BASIC program. Ily módon biztosítható a gépi kódú programok védelme, és megoldható az is, hogy különböző programok ugyanazt – a már memóriában levő – gépi kódú szubrutint használják.

Ha egy program a VLOMEM rendszerváltozót megváltoztatja, akkor ügyelni kell annak visszaállítására is. Ha már nincs szükség a felszabaduló memóriaterületen elhelyezett szubrutinokra, akkor állítsuk vissza a VLOMEM értékét az eredeti standard kezdőcimre.

OUT

Alakja: OUT kifejezés1,kifejezés2

Az utasítás az első kifejezéssel megcímzett fizikai portra beírja a második kifejezés értékét. A kifejezések 1 byte-os értékek lehetnek. Az utasítás használatához ismerni kell a számítógép egyes portjainak funkcióját és fizikai címét; használata nagy körültekintést igényel, mivel figyelmetlenül megválasztott cím, vagy hibás érték esetén a rendszer működésében hiba keletkezhet (programvesztés, stb.).

POKE

Alakja: POKE kifejezés1,kifejezés2

Az utasítás az első kifejezéssel megcímzett memóriarekeszbe a második kifejezés 1 byte-os értékét írja. Ha a cím a BASIC ROM területre mutat, akkor a video-RAM terület lesz kiválasztva.

Az utasítás használatához ismerni kell a rendszer memóriatérképét; hibás paraméterezéssel a rendszer működését megzavarhatjuk.

A POKE tipikus felhasználási területe a gépi kódú szubrutinok elhelyezése a LOMEM utasítással felszabadított memóriaterületre. A másik felhasználási terület a BASIC rendszerváltozók módosítása.

FÜGGVÉNYEK

A függvények ismertetésénél az X numerikus, X$ pedig string kifejezést jelöl.

ABS(X)

Egy kifejezés abszolút értékét adja. A kifejezésnek a
-0.9999999999E63<=X<=0.9999999999E63
tartományba kell esnie.

Az ABS függvény megváltoztatja a negatív érték előjelét, a pozitív kifejezés pedig változatlan marad.

Példa:
```
100 INPUT PROMPT "Két számot kérek: ":a,b
110 LET DIFF=ABS(a-b)
120 PRINT DIFF
```

A példában az ABS függvény lehetővé teszi, hogy a megadott két érték (a és b) különbségét meghatározzuk, függetlenül attól, hogy melyik érték volt a nagyobb.

ATN(X)

Egy kifejezés arcus-tangens függvényértékét adja, radiánban.

A művelet eredménye egy radiánban mért szögérték -pi/2 és pi/2 között. Ha az eredményt 180/pi-vel szorozzuk, akkor a szögértéket fokban kapjuk.

Példa:
```
100 PRINT ATN(1.7)*180/PI
```

A példa 1.7 arcus-tangensét adja fokokban.

CHR$(X)

Egy egykarakteres stringet eredményez, ahol a karaktert az X kifejezés értéke, mint karakterkód határozza meg. Az X kifejezésnek a 0<=X<=255 tartományba kell esnie.

Ha a kifejezés nem egész értéket eredményez, akkor a függvény az érték egész részét veszi figyelembe.

A 32–126 tartományban a függvény a szokásos ASCII karaktereket adja. A függvény a felhasználói grafikus karakterek megjelenítésére is alkalmas.

Példa:
```
FOR N=32 TO 126:PRINT CHR$(N),:NEXT N
```

A példa a 32–126 kódtartományban levő ASCII karaktereket írja ki.

COS(X)

Az X kifejezés koszinuszát eredményezi, ahol X radián értéket jelent. A kifejezésnek a -0.1E+13<X<0.1E+13 tartományba kell esnie, különben a függvény 0 értéket ad vissza.

Ha az X kifejezést fokokban mérjük, akkor a kifejezést szorozni kell pi/180-nal.

Példa: PRINT COS(67*PI/180)

A példa 67 fok koszinuszát számítja ki.

EXP(X)

Az e^x függvény értékét adja, tehát az e konstanst (2.71828 . . .) az x kifejezésben megadott hatványra emeli.

Példa: PRINT EXP(3.5),EXP(1)

A példa $e^{3,5}$, illetve az e^1 értékét írja ki.

FREE

A függvény azt az értéket adja, mely a program számára a RAM memóriában még rendelkezésre állt a FREE függvény meghívása előtti pillanatban.

Példa: PRINT FREE

IN(X)

A számítógép X kifejezéssel megcímzett fizikai portján található adat értékét adja meg. A függvény használatához a port címek és az egyes portok funkciójának ismerete feltétlenül szükséges.

INT(X)

Az X kifejezés értékénél nem nagyobb egész számot adja meg.

Az eredmény az a legnagyobb egész szám, amelyik kisebb, vagy egyenlő X kifejezés értékével. Tehát INT(9.87) 9-et, INT(-14.5) pedig -15-öt eredményez.

Példa:
```
100 INPUT PROMPT "Kérek egy egész számot:":A
120 IF A>INT(A) THEN PRINT "Nem jó": GOTO 100
```

LEN(X$)

Az X$ stringben levő karakterek darabszámát adja meg.

A string hossza max. 254 karakter lehet.

Példa:
```
100 INPUT PROMPT "KÉREK EGY SZÓT: ":A$
120 PRINT "A középső karakter: ";
130 PRINT A$((LEN(A$)+1)/2)
```

A példa a beírt szó középső karakterét írja ki, ha a szó páratlan számú karakterből állt. Páros számú karakter esetén az előző karakter az eredmény.

LOG(X)

Az X kifejezés természetes logaritmusát adja. A kifejezésnek

0<X<=0.9999999999E63

értéktartományba kell esnie.

A természetes logaritmus az e alapú logaritmust jelenti.

Példa:
```
PRINT LOG(15.8),LOG(1)
```

ORD(X$)

Az X$ string első karakterének kódtáblabeli kódját adja. A függvény ASCII kódokat generál.

Példa:
```
100 A$="+9N":PRINT ORD(A$)
```

A példa a + karakter ASCII kódját adja meg.

Lásd még a CHR$ függvényt is.

PEEK(X)

A függvény az X kifejezéssel megcímzett memóriacella pillanatnyi tartalmát adja meg. Ha a cím a BASIC ROM területet címzi, akkor a függvény a video RAM területről fog olvasni.

PI

A függvény a pi konstans értékét adja meg, azaz pi=3.141592654.

Példa: 100 INPUT PROMPT "a kör sugara? ":R
110 PRINT "a kör kerülete ";2*PI*R

RND(X) és RANDOMIZE utasítás

A függvény paramétere opcionális. Ha paraméter nem szerepel, akkor egy 0 és 1 közé eső pszeudo-véletlen számot ad a függvény.

Ha az X kifejezés adott, akkor a pszeudo-véletlen szám 0 és X-1 tartományban keletkezik. Ez tehát hatásában azonos az INT(RND*X) összetett függvényhívás hatásával, de az RND(X) lényegesen gyorsabb működésű.

A függvény által előállított legkisebb szám (a nullán kívül) 1/65536*X, a legnagyobb pedig 65535/65536*X.

A függvény csak 65535 féle értéket szolgáltat, de ismétlődés csak az 524287 érték után következik be az értékek sorozatában.

A RANDOMIZE utasítás a véletlenszám-sorozat kezdőértékét határozza meg, ami a RANDOMIZE hatására szintén véletlenszerűen keletkezik. A RANDOMIZE előzetes használatával biztosítható, hogy egy program minden egyes futtatása más és más RND sorozatokat állítson elő.

Példa: 100 PRINT RND(10)
120 PRINT RND(6)+1

A 100-as sor egy 0 és 9 közötti véletlen számot generál, a 120-as pedig a dobókockát modellezi.

SIN(X)

Az X kifejezés szinuszát adja meg. Az X kifejezés a szöget radiánokban határozza meg. X-nek -0.1E+13<X<0.1E+13 tartományba kell esnie, különben a függvény 0-át eredményez.

Példa: PRINT SIN(45*PI/180)

A példa 45 fok szinuszát fogja kiírni.

SGN(X)

A függvény az X kifejezés előjele alapján -1,0,+1 értéket eredményez.

Ha X pozitív, akkor SGN(X)= +1,
ha X nulla, akkor SGN(X)= 0,
ha X negatív, akkor SGN(X)= -1.

X-nek -0.9999999999E+63<=X<=0.9999999999E+63 tartományba kell esnie.

Példa:
```
100 INPUT PROMPT "kérek egy számot: ":N
110 s=SGN(N)
120 IF s=1 THEN PRINT "pozitív": ELSE IF s=0 THEN PRINT "nulla":
                                        ELSE PRINT "negatív"
```

STR$(X)

Az X kifejezés nyomtatási képét adja, tehát azt a karaktersorozatot, amely az adott numerikus kifejezés kiírásakor a képernyőn megjelenne. A kifejezésnek a

-0.9999999999E+63<=X<=0.9999999999E+63

tartományba kell esnie.

Példa:
```
100 a=12+(23*31)^2
110 a$=STR$(a)
120 PRINT a$(LEN(a$))
```

A példa a 100-as sorban adott kifejezés számértékének utolsó számjegyét fogja kiírni.

STRING$(n, x) vagy STRING$(n, x$)

A függvény a CHR$ láncolását végzi; n számú azonos karakterből álló stringet vesz fel értékül.

A paraméterek értelmezése:

x	– a karakter ASCII kódja, ahol 0<=X<=255
x$	– string, amelynek első karakterét veszi figyelembe a függvény
n	– ismétlési tényező, ahol 0<=n<=254

Példák:

```
PRINT STRING$(15,32)
```
15 db. space nyomtatása.

```
PRINT STRING$(20,"A")
```
20 db. A betű nyomtatása.

SQR(X)

A függvény az X kifejezés pozitív négyzetgyökét adja. Az X kifejezésnek

0<=X<=0.9999999999E+63

tartományba kell esnie.
Ha X negatív, akkor hibajelzés keletkezik.

Példa:
```
100 INPUT PROMPT "Együtthatók: ":a,b,c
110 d=b^2-4*a*c
120 if d<0 THEN PRINT "nem jó": goto 100
130 ds=SQR(d)
140 x1=-b+ds/(2*a)
150 x2=-b-ds/(2*a)
160 PRINT x1,x2
```

A példa az együtthatóival adott másodfokú egyenlet megoldását számítja ki.

TAN(X)

Az X kifejezés tangensének értékét adja meg. Az X szöget radiánban kell megadni. Ha X fokokban mért szög, akkor szorozni kell pi/180-al.

Példa:
```
PRINT TAN(15*PI/180)
```

A példa 15 fok tangensét írja ki.

USR(X, Y)

Egy felhasználói gépi kódú szubrutint hívó függvény. A felhasználói gépi kódú szubrutin kezdőcímét az X kifejezésben kell megadni. Az Y paraméter feltételes. Ha szerepel, akkor az Y értéke a szubrutin meghívása előtt a processzor HL regiszterpárjába kerül, ahol a gépi kódú rutin felhasználhatja. A függvény által visszaadott érték a szubrutin lefutása után a HL regiszterpár legutolsó tartalma, amit előjeles egésznek értelmez a függvény.

Lásd még a Felhasználói gépi kódú szubrutinok című alfejezetet.

VAL(X$)

Az X$ stringben levő karaktersorozatot numerikus értékké alakítja. Az átalakítás a következő szabályok szerint történik:

a) a függvény az X$-ban esetlegesen szereplő bevezető szóközöket figyelmen kívül hagyja;
b) az átalakítás az első, nem számjegyként értelmezhető karakterig tart;
c) ha X$ első karaktere nem számjegy, akkor a függvény 0-át ad eredményül.

Példa:
```
100 PRINT VAL("-13+256")
110 Dátum$="1985 március 1"
120 PRINT VAL(Dátum$)
130 PRINT VAL("PI*2")
```

A példában a 100-as sor -13-at, a 120-as sor 1985-öt, a 130-as pedig 0-át ír ki.

VARPTR(X) vagy VARPTR(X$)

A függvény a megadott változó vagy tömbelem pillanatnyi memóriabeli címét veszi fel értékül.

A VARPTR-el kapott értéket egy gépi kódú szubrutinban használhatjuk fel egy változó megcímzésére. A VARPTR hívásakor az adott változó típusát a TYPE (címe: 5896 decimális) rendszerváltozó tartalmazza.

Vigyázat! A program módosítása után a változók címe megváltozhat, ezért a VARPTR is más értéket ad.

VERNUM

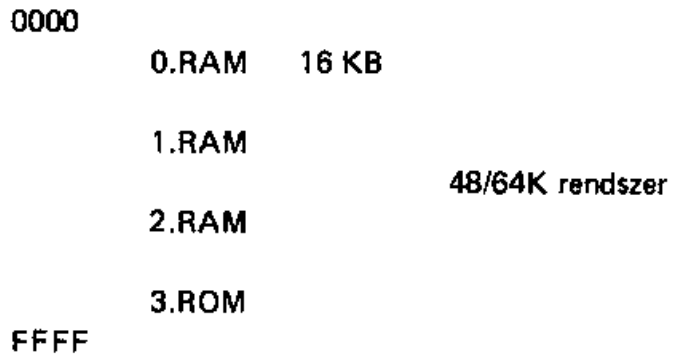
A BASIC azonosítószámát adja meg.

A legnagyobb helyiértékű számjegy a verziószám, az ezutáni számjegyek az alváltozatot jelölik.

MEMÓRIAFELOSZTÁS

Memóriatérkép

A BASIC rendszer használatakor a memória az alábbi felépítésű:

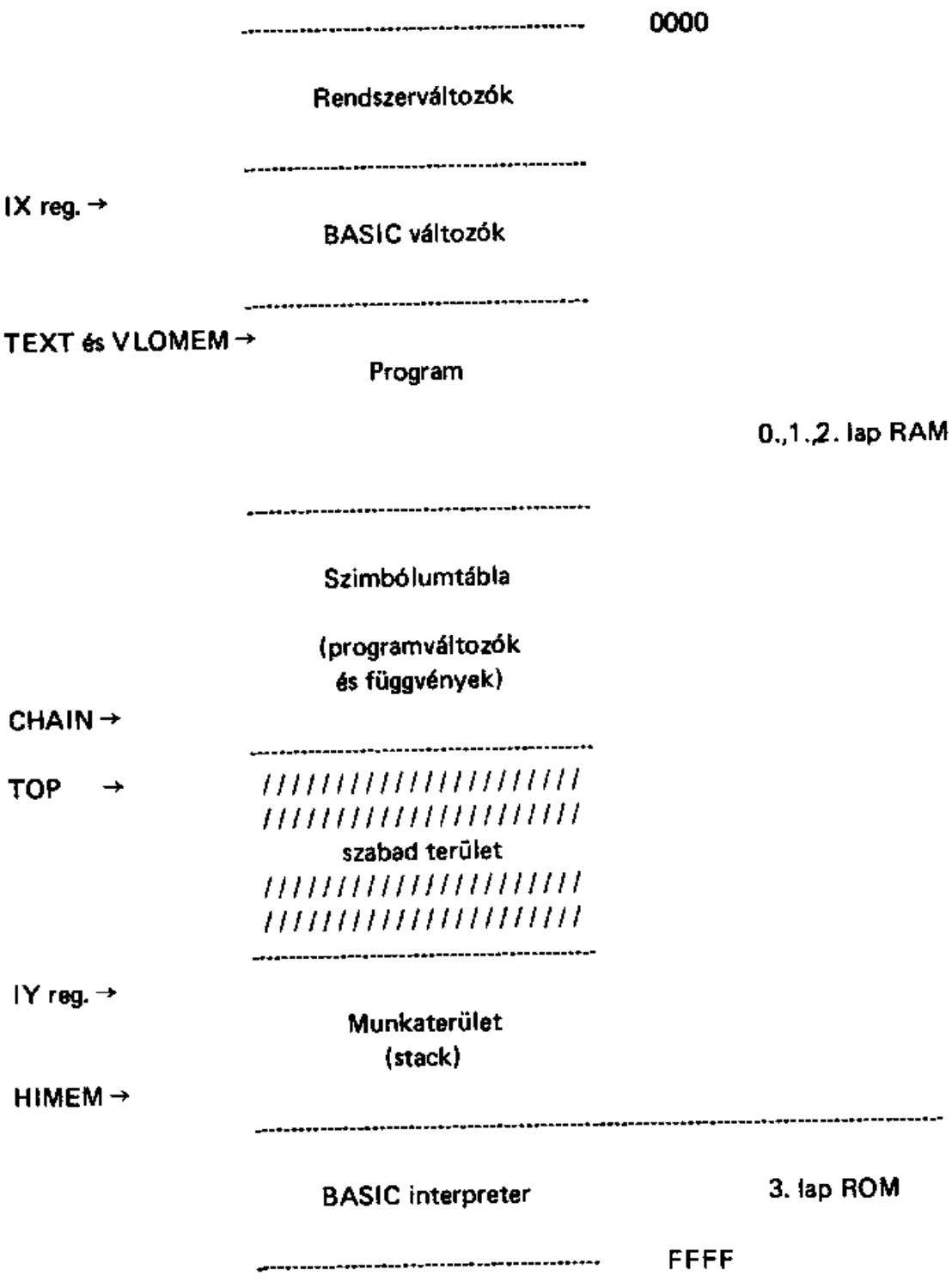


A BASIC és az operációs rendszer ROM memóriája mindig a 3-as lapon található, a 0-ás lap pedig minden konfigurációban RAM memória. A 32K-s vagy annál nagyobb rendszerekben az 1-es lap, 48K-s vagy nagyobb rendszerekben pedig a 2-es lap is RAM memória. 64K méretű rendszerekben a BASIC sajnos nem tudja kihasználni a legfelső 16K területet, mivel a 3-as lap ROM memória.

A 0-ás lapon a BASIC és az operációs rendszer a rendszerváltozókat tárolja. Efelett a BASIC változók és a memóriában levő BASIC program találhatók, amit a szimbólum tábla követ.

A BASIC rendszer munkaterülete (stack a FOR ... NEXT,GOSUB számára) a RAM legnagyobb címétől a kisebb címek felé terjeszkedik.

Az IX regiszter mindig a BASIC változók kezdőcímére, az IY pedig a BASIC stack tetejére mutat.


0000
Rendszerváltozók
IX reg. →
BASIC változók
TEXT és VLOMEM →
Program
0.,1.,2. lap RAM
Szimbólumtábla
(programváltozók
és függvények)
CHAIN →
TOP →
szabad terület
IY reg. →
Munkaterület
(stack)
HIMEM →
BASIC interpreter
3. lap ROM
FFFF

BASIC változók és munkaterületek

A BASIC interpreter több olyan belső változót és munkaterületet használ, amelyek a felhasználói programból is elérhetők a POKE utasítás, ill. a PEEk függvény segítségével.

Név	Cím	Hossz(byte)	Funkció
AUTO	5895(1707H)	1	Automatikus programindítás a betöltés után, ha mentéskor az értéke 255
FILENAME	6606(19CEH)	17	Filenév-puffer. 1.byte = filenév hossza 2—17 = filenév
VLOMEM	5920(1720H)	2	A BASIC terület kezdete
USRTAB	33(21H)	7*2	A gépi kódú szubrutinok címeit tartalmazó táblázat
START	5900(170CH)	2	Az aktuális BASIC sor kezdőcíme
TEXT	5922(1722H)	2	A BASIC program kezdőcíme
TYPE	5896(1708H)	1	A szimbólumtábla aktuális elemének típusa
CHAIN'	5924(1724H)	2	A szimbólumtábla utolsó elemének címe
TOP	5926(1726H)	2	A szimbólumtábla következő szabad byte-jának címe
COMMAND	5938(1732H)	255	Az aktuális BASIC utasítássort tartalmazó puffer
BUFFER	6193(1831H)	255	Input puffer a klaviatúra számára
X	6592(19C0H)	7	Lebegőpontos aritmetikai regiszter
Y	6599(19C7H)	7	Lebegőpontos aritmetikai regiszter
STOPFL	2838(0B16H)	1	= FF — CTRL ESC megnyomása a klaviatúrán =0 — nem volt CTRL ESC

Név	Cím	Hossz(byte)	Funkció
HIMEM	2841(0B19H)	2	A legmagasabb RAM cím
P3RAM	2843(0B1BH)	1	= 0 — a 3.RAM lap jó = FF — hiba a 3.RAM lapon
INTINC	2845(0B1DH)	2	Számláló: értéke 20 ms-onként eggyel nő
COLD FLAG	2850(0B22H)	1	= 0,WARM RESET engedélyezése = FF,WARM RESET tiltása
MODE	2891(0B4BH)	1	Grafikus pont felülírási módja
STYLE	2892(0B4CH)	1	Vonaltípus
INK	2893(0B4DH)	1	Tintaszín
PAPER	2894(0B4EH)	1	Papírszín
BORDER	2895(0B4FH)	1	Keretszín
VFLAG	2896(0B50H)	1	Karakter felülírási flag
PICTURE	2897(0B51H)	10	A klaviatúráról utoljára beolvasott karakter mátrixa
DELAYKEY	2917(0B65H)	1	Auto-repeat késleltetési idő
LOCK KEY	2918(0B66H)	1	CTRL/SHIFT/ALT állapota b1 = CAPS LOCK b2 = SHIFT LOCK b8 = ALT LOCK
RATE KEY	2919(0B67H)	1	Auto-repeat időzítés
HOLD DIS	2920(0B68H)	1	= 0 — a HOLD mód engedélyezése; a CTRL—P használható = FF — a HOLD mód tiltása
EOF	2926(0B6EH)	1	= 0 — nincs file vége <>0 — file vége
PROGRAM	6639(19EFH)	—	A BASIC program előre definiált kezdete

ADATÁBRÁZOLÁS

Programok a memóriában

A program a TEXT rendszerváltozó által meghatározott címen kezdődik. A program növekvő sorszámok szerint van tárolva. Minden sor az alábbi felépítésű:

1 byte – a következő sor távolsága, azaz az adott sor hossza + 1
2 byte – a sor sorszáma binárisan ábrázolva
n byte – maga a sor tömörített formában, $0 < n <= 250$
1 byte – OFFH sorvégjel

A program végét null karakter jelzi. Az ezutáni byte-on a szimbólumtábla kezdődik.

A tömörített sor a programozó által beírt utasítássort tartalmazza, de a kulcsszavak helyett 1 byte-os jelek állnak, melyeknél a legfelső bit 1-re van állítva.

A BASIC az utasítássorban szereplő DATA adatokat, stringkonstansokat és megjegyzéseket nem tömöríti. A nem tömörített karakterek normál ASCII karakterként szerepelnek.

A szimbólumtábla

A szimbólumtábla azonosítók láncolt listájából áll. A táblázat alaphelyzetben a BASIC beépített azonosítóit tartalmazza.

Minden bejegyzés első két byte-ja az előző bejegyzés első byte-jára mutat; illetve 0-át tartalmaz, ha ez az első bejegyzés a táblázatban.

A CHAIN rendszerváltozó az utolsó bejegyzésre mutat.

Ha egy felhasználói azonosító kerül a táblázatba, akkor a CHAIN tartalma beíródik az új bejegyzés első két byte-jába, a CHAIN pedig felveszi az új bejegyzés címét.

A keresés a szimbólumtáblában mindig a CHAIN-től indul, ezért egy felhasználói szimbólum felülírhatja, újra definiálhatja a beépített azonosítókat.

Ha egy azonosító függvény-definíciós utasítás paramétere, akkor az adott azonosítónak lokálisnak kell lennie az adott utasításra nézve. Ezt a BASIC interpreter biztosítja a CHAIN régi értékének elmentésével és a függvény kiértékelése utáni visszaállításával, azaz a függvény kiértékelése közben csak ideiglenesen hoz létre új bejegyzést a függvényparaméter számára a szimbólumtáblában.

Minden szimbólumbejegyzés az alábbi felépítésű:

2 byte – az előző bejegyzésre mutató cím
1 byte – az azonosító hossza
n byte – azonosítónév
1 byte – típusbyte
d byte – adat

A típusbyte a szimbólum típusát írja le. Ez határozza meg, hogy az adatbyte-okat hogyan kell értelmezni.

A típusbyte 0.bitje – tömb esetén 1
1.bitje – numerikus változó esetén 1, stringnél 0
2.bitje – DEF utasításhoz tartozó azonosítónál 1
3.bitje – beépített függvénynél 1

Egyszerű változó esetén az adatbyte-ok kezdetben nullát tartalmaznak.

Numerikus változóknál hat adatbyte tartalmazza az értéket, tömörített BCD formátumban. Az első byte a két legkisebb helyiértékű számjegypárt, a hatodik byte pedig a 64-el eltolva ábrázolt kitevőt tartalmazza. A kitevő legfelső bitje a szám előjele. A szám mindig normalizált.

Stringek esetén az első adatbyte a string maximális hosszát (alaphelyzetben 18-at), a második byte pedig az aktuális hosszt tartalmazza. A további byte-ok a string karaktereit tárolják karakterenként egy byte-on.

Tömbök esetén az első adatbyte a dimenziószám (n). A következő 2*n byte dimenziónként a maximális index értéket tartalmazza, 16 bites bináris számmal ábrázolva. Ezt követik a tömb egyes elemeinek értékei úgy, mintha egy-egy egyszerű változó állna ott.

Felhasználó által definiált függvény esetén az első két adatbyte a DEF utasítást tartalmazó utasítássor kezdetére mutat, a következő két byte pedig az aktuális programsorban a függvény nevét követő első karakterre (tehát "=" jelre, vagy a "(" jelre a paraméter előtt).

Beépített függvények esetén az adatbyte-ok gépi kódú utasítások, melyek a függvényt megvalósítják.

A BASIC stack

A kifejezések kiértékelése közben a BASIC munkaterületet igényel a közbülső részeredmények tárolására. Ugyanez a munkaterület szolgál a FOR . . . NEXT és GOSUB utasításokban a hívás helyének tárolására.

A stack a HIMEM rendszerváltozóban tárolt értéktől kezdődve a kisebb memóriacímek felé terjed. A legutoljára használt byte-ra az IY regiszter mutat.

A stack-ben levő minden elemet megelőz egy byte, melynek a jelentése a következő:

ha a byte = 0 – munkaterület vége
byte = 1 – string következik
byte = 6 – GOSUB-hoz tartozó cím következik
byte = 9 – szám következik
byte = 43 – FOR-hoz tartozó cím következik

A string esetét kivéve a byte értéke egyúttal az elem hosszát is megadja.

Stringek esetében a következő byte a string hossza, amit a karakterbyte-ok követnek. A string a munkaterületen mindig csak az aktuális hosszának megfelelő hosszban foglal helyet.

GOSUB cím esetén a következő byte-ok felépítése az alábbi:

2 byte – a GOSUB után következő utasítás címe
2 byte – a GOSUB-ot tartalmazó sor kezdőcíme
1 byte – jelzi, hogy ON GOSUB-nál ELSE ág adott

Szám esetén a 8 byte felépítése a következő:

1 byte – két további számjegy BCD alakban
6 byte – a szimbólumtáblában megadottal azonos ábrázolással a szám
1 byte – két túlcsorduló számjegy
A kibővített 8 byte-os ábrázolás a számítások nagyobb pontosságát biztosítja.

FOR cím esetén a következő byte-ok felépítése az alábbi:

9 byte – TO értéke
9 byte – a NEXT utasítás használja
9 byte – a NEXT utasítás használja
9 byte – a STEP értéke
2 byte – a ciklusváltozó címe az adatterületen
2 byte – a FOR-t tartalmazó sor kezdőcíme
2 byte – a FOR-t követő utasítás címe

File formátum

A SAVE utasítás a program pontos memóriaképét írja a file-ba, beleértve a befejező 0 byte-ot is. Ezt megelőzi azonban egy 16 byte-os fejléc, mely a későbbiekben lehetővé teszi a program kezdetének felismerését, és ezen kívül a betöltés kezdetén a BASIC megtudhatja a program méretét.

A fejléc formátuma:

1 byte – 0
1 byte – típusbyte
2 byte – méret
1 byte – auto-run üzemmód
10 byte – jelenleg nem használt (0)
1 byte – változatszám (0)

A programfile-ok esetén a típusbyte értéke 1, minden más esetben más érték. ASCII file-ok esetén a típusbyte értéke 0.
A méret a program hossza byte-okban, beleértve a befejező nulla byte-ot is.
Az auto-run byte nem nulla értéke jelzi, ha a programnak töltés után azonnal (RUN nélkül) indulni kell.
A fennmaradó byte-ok jelenleg 0-ra vannak állítva.

FELHASZNÁLÓI GÉPI KÓDÚ SZUBRUTINOK

A felhasználó az USR függvénnyel és az EXT utasítással hívhat saját gépi kódú szubrutinokat.

Az USR függvény egy paramétert adhat át a HL regiszterpárba, és ugyanonnan paraméter érték kapható vissza. A rutinnak RET utasítással kell befejeződnie.

A szubrutin felhasználhatja a HL, BC, DE, AF, DE', AF' és C' regisztereket Felhasználható ezen kívül az alábbi információ:

IX – a BASIC változó területére mutat
IY – a BASIC munkaterület utolsó elemére mutat
HL' – a forrásprogram következő elemére mutat

A gépi kódú szubrutin elhelyezésének szokásos módja az, hogy a program DATA mezőkben tartalmazza a gépi kódokat és felhasználás előtt READ és POKE utasításokkal a program elhelyezi a gépi utasításkódokat a memória megfelelő helyén. A program kezdőcím LOMEM-el történő átállítása után a gépi kódú program a régi programkezdőcímen kezdődhet és az aktuális LOMEM-el beállított új programkezdőcímig terjedhet.

A felhasználói szubrutin BASIC függvényként is megvalósítható. Ekkor a LOMEM utasítással védett területre egy szimbólumtábla elemet POKE-olunk be, a típusbyte-ban jelezve, hogy beépített függvényről van szó. Természetesen ekkor a szimbólumtáblánál leírtak szerint a CHAIN változó módosítása is szükséges. A szimbólumtábla bejegyzést követi a gépi kódú szubrutin. Ilyen megoldásnál a gépi kódú szubrutin úgy fog működni, mint a BASIC beépített függvényei. A szubrutin ekkor a választ a BASIC stack-ben adja vissza, az IY regiszter megfelelő állítása mellett.

A gépi kódú rutinok hívásának harmadik lehetősége az EXT utasítás. Itt az első paraméter határozza meg, hogy hányadik szubrutint kell elindítani, a további paraméterek pedig a HL, DE, BC regiszterekben adhatnak át értékeket. A szubrutin ugyanazokat a regisztereket használhatja, mint az USR függvénnyel hívott szubrutin, de az EXT utasítás mellett nincs lehetőség paraméter visszaadásra.

A szubrutin kezdőcímét az USRTAB táblázat tartalmazza. A táblázatban az első két byte az EXT 0, a második két byte az EXT 1 . . . stb. szubrutin kezdőcímet tartalmazza. Ha címként nulla szerepel, akkor a BASIC a hozzá tartozó EXT utasítást üres utasításként hajtja végre.

FÜGGELÉK

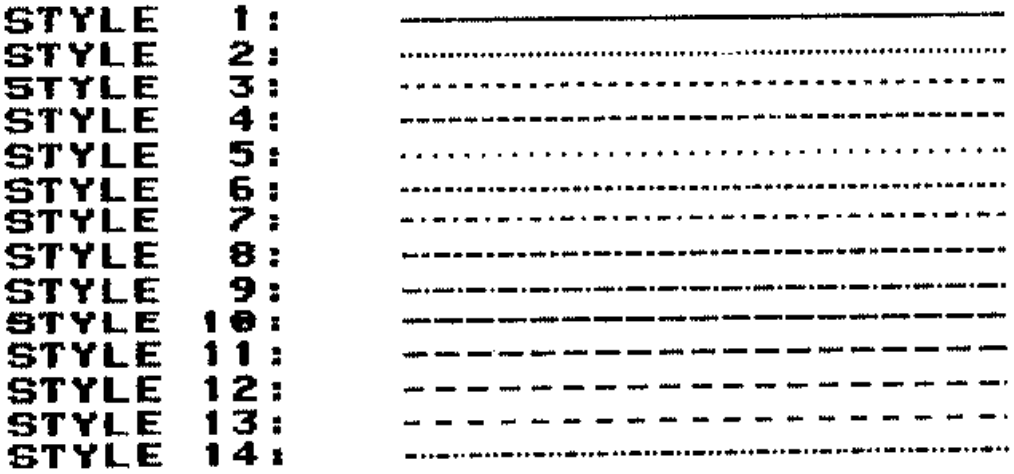
Színsorszámok és palettakódok

színsorszám	palettakód	
0	0	fekete
1	1	sötétkék
2	4	sötétvörös
3	5	sötétlila
4	16	sötétzöld
5	17	sötét kékeszöld (cián)
6	20	sötétsárga
7	21	szürke
8	64	fekete
9	65	kék
10	68	vörös
11	69	lila
12	80	zöld
13	81	kékeszöld (cián)
14	84	sárga
15	85	fehér

Palettakód színbitek:

128	64	32	16	8	4	2	1
	0: sötét 1: világos		zöld		vörös		kék

Vonaltípusok



Gépi kódú utasítások

Kód	Hex	Z80 Assembly	CB után	ED után
0	0	NOP	RLC B	
1	1	LD BC,NN	RLC C	
2	2	LD (BC),A	RLC D	
3	3	INC BC	RLC E	
4	4	INC B	RLC H	
5	5	DEC B	RLC L	
6	6	LD B,N	RLC (HL)	
7	7	RLCA	RLC A	
8	8	EX AF,AF'	RRC B	
9	9	ADD HL,BC	RRC C	
10	A	LD A, (BC)	RRC D	
11	B	DEC BC	RRC E	
12	C	INC C	RRC H	
13	D	DEC C	RRC L	
14	E	LD C,N	RRC (HL)	
15	F	RRCA	RRC A	
16	10	DJNZ DIS	RL B	
17	11	LD DE,NN	RL C	
18	12	LD (DE),A	RL D	
19	13	INC DE	RL E	
20	14	INC D	RL H	

Kód	Hex	Z80 Assembly	CB után	ED után
21	15	DEC D	RL L	
22	16	LD D,N	RL (HL)	
23	17	RLA	RL A	
24	18	JR DIS	RR B	
25	19	ADD HL,BC	RR C	
26	1A	LD A,(DE)	RR D	
27	1B	DEC DE	RR E	
28	1C	INC E	RR H	
29	1D	DEC E	RR L	
30	1E	LD E,N	RR (HL)	
31	1F	RRA	RR A	
32	20	JR NZ,DIS	SLA B	
33	21	LD HL,NN	SLA C	
34	22	LD (NN),HL	SLA D	
35	23	INC HL	SLA E	
36	24	INC H	SLA H	
37	25	DEC H	SLA L	
38	26	LD H,H	SLA (HL)	
39	27	DAA	SLA A	
40	28	JR Z,DIS	SRA B	
41	29	ADD HL,HL	SRA C	
42	2A	LD HL,(NN)	SRA D	
43	2B	DEC HL	SRA E	
44	2C	INC L	SRA H	
45	2D	DEC L	SRA L	
46	2E	LD L,N	SRA (HL)	
47	2F	CPL	SRA A	
48	30	JR NC,DIS		
49	31	LD SP,NN		
50	32	LD (NN),A		
51	33	INC SP		
52	34	INC (HL)		
53	35	DEC (HL)		
54	36	LD (HL),N		
55	37	SCF		
56	38	JR C,DIS	SRL B	
57	39	ADD HL,SP	SRL C	
58	3A	LD A,(NN)	SRL D	
59	3B	DEC SP	SRL E	
60	3C	INC A	SRL H	
61	3D	DEC A	SRL L	
62	3E	LD A,N	SRL (HL)	
63	3F	CCF	SRL A	

Kód	Hex	Z80 Assembly	CB után	ED után
64	40	LD B,B	BIT 0,B	IN B,(C)
65	41	LD B,C	BIT 0,C	OUT (C),B
66	42	LD B,D	BIT 0,D	SBC HL,BC
67	43	LD B,E	BIT 0,E	LD (NN),BC
68	44	LD B,H	BIT 0,H	NEG
69	45	LD B,L	BIT 0,L	RETN
70	46	LD B,(HL)	BIT 0,(HL)	IM 0
71	47	LD B,A	BIT 0,A	LD I,A
72	48	LD C,B	BIT 1,B	IN C,(C)
73	49	LD C,C	BIT 1,C	OUT (C),C
74	4A	LD C,D	BIT 1,D	ADC HL,BC
75	4B	LD C,E	BIT 1,E	LD BC,(NN)
76	4C	LD C,H	BIT 1,H	
77	4D	LD C,L	BIT 1,L	RETI
78	4E	LD C,(HL)	BIT 1,(HL)	
79	4F	LD C,A	BIT 1,A	LD R,A
80	50	LD D,B	BIT 2,B	IN D,(C)
81	51	LD D,C	BIT 2,C	OUT (C),D
82	52	LD D,D	BIT 2,D	SBC HL,DE
83	53	LD D,E	BIT 2,E	LD (NN),DE
84	54	LD D,H	BIT 2,H	
85	55	LD D,L	BIT 2,L	
86	56	LD D,(HL)	BIT 2,(HL)	IM 1
87	57	LD D,A	BIT 2,A	LD A,I
88	58	LD E,B	BIT 3,B	IN E,(C)
89	59	LD E,C	BIT 3,C	OUT (C),E
90	5A	LD E,D	BIT 3,D	ADC HL,DE
91	5B	LD E,E	BIT 3,E	LD DE,(NN)
92	5C	LD E,H	BIT 3,H	
93	5D	LD E,L	BIT 3,L	
94	5E	LD E,(HL)	BIT 3,(HL)	IM 2
95	5F	LD E,A	BIT 3,A	LD A,R
96	60	LD H,B	BIT 4,B	IN H,(C)
97	61	LD H,C	BIT 4,C	OUT (C),H
98	62	LD H,D	BIT 4,D	SBC HL,HL
99	63	LD H,E	BIT 4,E	LD (NN),HL
100	64	LD H,H	BIT 4,H	
101	65	LD H,L	BIT 4,L	
102	66	LD H,(HL)	BIT 4,(HL)	
103	67	LD H,A	BIT 4,A	RRD
104	68	LD L,B	BIT 5,B	IN L,(C)
105	69	LD L,C	BIT 5,C	OUT (C),L
106	6A	LD L,D	BIT 5,D	ADC HL,HL

Kód	Hex	Z80 Assembly	CB után	ED után
107	6B	LD L,E	BIT 5,E	LD HL,(NN)
108	6C	LD L,H	BIT 5,H	
109	6D	LD L,L	BIT 5,L	
110	6E	LD L,(HL)	BIT 5,(HL)	
111	6F	LD L,A	BIT 5,A	RLD
112	70	LD (HL),B	BIT 6,B	
113	71	LD (HL),C	BIT 6,C	
114	72	LD (HL),D	BIT B,D	SBC HL,SP
115	73	LD (HL),E	BIT 6,E	LD (NN),SP
116	74	LD (HL), H	BIT 6,H	
117	75	LD (HL),L	BIT 6,L	
118	76	HALT	BIT 6,(HL)	
119	77	LD (HL),A	BIT 6,A	
120	78	LD A,B	BIT 7,B	IN A,(C)
121	79	LD A,C	BIT 7,C	OUT (C),A
122	7A	LD A,D	BIT 7,D	ADC HL,SP
123	7B	LD A,E	BIT 7,E	LD SP,(NN)
124	7C	LD A,H	BIT 7,H	
125	7D	LD A,L	BIT 7,L	
126	7E	LD A,(HL)	BIT 7,(HL)	
127	7F	LD A,A	BIT 7,A	
128	80	ADD A,B	RES 0,B	
129	81	ADD A,C	RES 0,C	
130	82	ADD A,D	RES 0,D	
131	83	ADD A,E	RES 0,E	
132	84	ADD A,H	RES 0,H	
133	85	ADD A,L	RES 0,L	
134	86	ADD A,(HL)	RES 0,(HL)	
135	87	ADD A,A	RES 0,A	
136	88	ADC A,B	RES 1,B	
137	89	ADC A,C	RES 1,C	
138	8A	ADC A,D	RES 1,D	
139	8B	ADC A,E	RES 1,E	
140	8C	ADC A,H	RES 1,H	
141	8D	ADC A,L	RES 1,L	
142	8E	ADC A,(HL)	RES 1,(HL)	
143	8F	ADC A,A	RES 1,A	
144	90	SUB B	RES 2,B	
145	91	SUB C	RES 2,C	
146	92	SUB D	RES 2,D	
147	93	SUB E	RES 2,E	
148	94	SUB H	RES 2,H	
149	95	SUB L	RES 2,L	

Kód	Hex	Z80 Assembly	CB után	ED után
150	96	SUB (HL)	RES 2,(HL)	
151	97	SUB A	RES 2,A	
152	98	SBC A,B	RES 3,B	
153	99	SBC A,C	RES 3,C	
154	9A	SBC A,D	RES 3,D	
155	9B	SBC A,E	RES 3,E	
156	9C	SBC A,H	RES 3,H	
157	9D	SBC A,L	RES 3,L	
158	9E	SBC A,(HL)	RES 3,(HL)	
159	9F	SBC A,A	RES 3,A	
160	A0	AND B	RES 4,B	LDI
161	A1	AND C	RES 4,C	CPI
162	A2	AND D	RES 4,D	INI
163	A3	AND E	RES 4,E	OUTI
164	A4	AND H	RES 4,H	
165	A5	AND L	RES 4,L	
166	A6	AND(HL)	RES 4,(HL)	
167	A7	AND A	RES 4,A	
168	A8	XOR B	RES 5,B	LDD
169	A9	XOR C	RES 5,C	CPD
170	AA	XOR D	RES 5,D	IND
171	AB	XOR E	RES 5,E	OUTD
172	AC	XOR H	RES 5,H	
173	AD	XOR L	RES 5,L	
174	AE	XOR (HL)	RES 5,(HL)	
175	AF	XOR A	RES 5,A	
176	B0	OR B	RES 6,B	LDIR
177	B1	OR C	RES 6,C	CPIR
178	B2	OR D	RES 6,D	INIR
179	B3	OR E	RES 6,E	OTIR
180	B4	OR H	RES 6,H	
181	B5	OR L	RES 6,L	
182	B6	OR (HL)	RES 6,(HL)	
183	B7	OR A	RES 6,A	
184	B8	CP B	RES 7,B	LDDR
185	B9	CP C	RES 7,C	CPDR
186	BA	CP D	RES 7,D	INDR
187	BB	CP E	RES 7,E	OTDR
188	BC	CP H	RES 7,H	
189	BD	CP L	RES 7,L	
190	BE	CP (HL)	RES 7,(HL)	
191	BF	CP A	RES 7,A	
192	C0	RET NZ	SET 0,B	

Kód	Hex	Z80 Assembly	CB után	ED után
193	C1	POP BC	SET 0,C	
194	C2	JP NZ,NN	SET 0,D	
195	C3	JP NN	SET 0,E	
196	C4	CALL NZ,NN	SET 0,H	
197	C5	PUSH BC	SET 0,L	
198	C6	ADD A,N	SET 0,(HL)	
199	C7	RST 0	SET 0,A	
200	C8	RET Z	SET 1,B	
201	C9	RET	SET 1,C	
202	CA	JP Z,NN	SET 1,D	
203	CB		SET 1,E	
204	CC	CALL Z,NN	SET 1,H	
205	CD	CALL NN	SET 1,L	
206	CE	ADC A,N	SET 1,(HL)	
207	CF	RST 8	SET 1,A	
208	D0	RET NC	SET 2,B	
209	D1	POP DE	SET 2,C	
210	D2	JP NC,NN	SET 2,D	
211	D3	OUT (N),A	SET 2,E	
212	D4	CALL NC,NN	SET 2,H	
213	D5	PUSH DE	SET 2,L	
214	D6	SUB N	SET 2,(HL)	
215	D7	RST 16	SET 2,A	
216	D8	RET C	SET 3,B	
217	D9	EXX	SET 3,C	
218	DA	JP C,NN	SET 3,D	
219	DB	IN A,(N)	SET 3,E	
220	DC	CALL C,NN	SET 3,H	
221	DD	IX-el indexelt utasítások*	SET 3,L	
222	DE	SBC A,N	SET 3,(HL)	
223	DF	RST 24	SET 3,A	
224	E0	RET PO	SET 4,B	
225	E1	POP HL	SET 4,C	
226	E2	JP PO,NN	SET 4,D	
227	E3	EX (SP),HL	SET 4,E	
228	E4	CALL PO,NN	SET 4,H	
229	E5	PUSH HL	SET 4,L	
230	E6	AND N	SET 4,(HL)	
231	E7	RST 32	SET 4,A	
232	E8	RET PE	SET 5,B	
233	E9	JP (HL)	SET 5,C	
234	EA	JP PE,NN	SET 5,D	

Kód	Hex	Z80 Assembly	CB után	ED után
235	EB	EX DE,HL	SET 5,E	
236	EC	CALL PE,NN	SET 5,H	
237	ED		SET 5,L	
238	EE	XR N	SET 5,(HL)	
239	EF	RST 40	SET 5,A	
240	F0	RET P	SET 6,B	
241	F1	POP AF	SET 6,C	
242	F2	JP P,NN	SET 6,D	
243	F3	DI	SET 6,E	
244	F4	CALL P,NN	SET 6,H	
245	F5	PUSH AF	SET 6,L	
246	F6	OR N	SET 6,(HL)	
247	F7	RST 48	SET 6,A	
248	F8	RET M	SET 7,B	
249	F9	LD SP,HL	SET 7,C	
250	FA	JP M,NN	SET 7,D	
251	FB	EI	SET 7,E	
252	FC	CALL M,NN	SET 7,H	
253	FD	IY-nal indexelt utasítások**	SET 7,L	
254	FE	CP N	SET 7,(HL)	
255	FF	RST 56	SET 7,A	

*A DD után azok az utasítások következhetnek, amelyekben a HL regiszterpár címzi meg a memóriabyte-ot, ill. ez a regiszterpár az operandus.

**Az FD után azok az utasítások állhatnak, amelyekben a HL regiszterpár címzi meg a memóriabyte-ot, ill. ez a regiszterpár az operandus.

A BASIC-ben nem definiált matematikai függvények meghatározása az alapfüggvényekkel

Függvénynév	Függvény
szekáns	SEC(X)=1/COS(X)
koszekáns	CSC(X)=1/SIN(X)
kotangens	COT(X)=1/TAN(X)
arkusz szinusz	ARCSIN(X)=ATN(X/SQR(-X*X+1))
arkusz koszinusz	ARCCOS(X)=ABS(ATN(X/SQR(-X*X+1))−1.5708)
szinusz hiperbolikusz	SINH(X)=(EXP(X)-EXP(-X))/2
koszinusz hiperbolikusz	COSH(X)=(EXP(X)+EXP(-X))/2
tangens hiperbolikusz	TANH(X)=EXP(-X)/(EXP(X)-EXP(-X))*(-2)+1
kotangens hiperbolikusz	COTH(X)=EXP(-X)/(EXP(X)-EXP(-X))*2+1
area szinusz hiperbolikusz	ARSINH(X)=LOG(X+SQR(X*X+1))
area koszinusz hiperbolikusz	ARCOSH(X)=LOG(X+SQR(X*X-1))
area tangens hiperbolikusz	ARTANH(X)=LOG((1+X)/(1-X))/2
area kotangens hiperbolikusz	ARCOTH(X)=LOG((X+1)/(X-1))/2
tizes alapú logaritmus	TIZLOG(X)=LOG(X)/LOG(10)

Bináris–Hexadecimális konverziós táblázat

Bin	Hex	Dec
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Hexadecimális-decimális konverziós táblázat

Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec
1000	4096	100	256	10	16	1	1
2000	8192	200	512	20	32	2	2
3000	12288	300	768	30	48	3	3
4000	16384	400	1024	40	64	4	4
5000	20480	500	1280	50	80	5	5
6000	24576	600	1536	60	96	6	6
7000	28672	700	1792	70	112	7	7
8000	32768	800	2048	80	128	8	8
9000	36864	900	2304	90	144	9	9
A000	40960	A00	2560	A0	160	A	10
B000	45056	B00	2816	B0	176	B	11
C000	49152	C00	3072	C0	192	C	12
D000	53248	C00	3328	D0	208	D	13
E000	57344	E00	3584	E0	224	E	14
F000	61440	F00	3840	F0	240	F	15

Pl.:

FC4AH	F000H	61440
	C00H	3072
	40H	64
	AH	+ 10
	FC4AH	64586

Mintafeladatok

1

```
  1 REM Színes vonalak
 10 GRAPHICS 16
 20 RANDOMIZE
 30 FOR I=1 TO 50:REM 50 vonal
 40 PLOT RND(1023),RND(959);RND
(1023),RND(959)
 50 SET INK RND(15)
 60 SET STYLE RND(15)
 70 NEXT I
 75 REM Késleltetés
 80 FOR J=1 TO  999:NEXT J
 90 GOTO 30
```

2

```
  1 REM Óra program
 10 GRAPHICS 4
 20 SET PALETTE 80,68,0,85
 30 SET PAPER 12:SET BORDER 80
 40 CLS
 45 REM A számlap felrajzolása
 50 FOR N=1 TO 12
 60 PRINT AT 11-10*COS(N/6*3.14
),16+10*SIN(N/6*3.14):N
 70 NEXT N
 80 FOR T=0 TO 200000:REM Másod
percek
 90 A=T/30*PI
100 KX=240*SIN(A):KY=250*COS(A)
110 SET MODE 0
120 PLOT 512,560;512+KX,560+KY
125 REM Késleltetés kb. 1 mp.
130 FOR N=0 TO 400:NEXT
140 SET MODE 3
150 PLOT 512,560;512+KX,560+KY
160 NEXT T
```

```
    1 REM Számok rendezése
   10 CLS
   20 PRINT"Gépeljen be számokat,
"
   30 PRINT"a program növekvö sor
rendbe"
   40 PRINT"rendezi azokat"
   50 INPUT PROMPT"Hány számot ki
ván rendezni?";S
   60 DIM SZAM(S-1):CLS
   70 FOR N=0 TO S-1
   80 PRINT USING"##. ";N+1;
   90 INPUT A
  100 SZAM(N)=A:REM a SZAM tömb
tartalmazza a számokat
  110 NEXT N
  115 REM Rendezes
  120 FOR M=1 TO S-1
  130 FOR N=0 TO S-2
  140 IF SZAM(N)<=SZAM(N+1) THEN
170
  150 B=SZAM(N):SZAM(N)=SZAM(N+1)
  160 SZAM(N+1)=B
  170 NEXT N,M
  180 CLS:PRINT"A rendezett számo
k:"
  190 FOR N=0 TO S-1
  200 PRINT USING "##. ";N+1;
  210 PRINT SZAM(N)
  220 NEXT N
  230 END
```

4

```
  1 REM Decimális-hexadecimális
    konverzió
 10 GRAPHICS 4:DIM H(4)
 20 SET PALETTE 81,68,64,1
 30 SET BORDER 1:SET PAPER 13
 40 SET INK 8:CLS
 50 PRINT"Decimális szám  :";
 60 INPUT D
 70 IF D>65535 THEN 50
 80 IF D=0 THEN END
 90 REM D=0 program vége
100 H(1)=INT(D/4096)
110 D=D-H(1)*4096
120 H(2)=INT(D/256)
130 D=D-H(2)*256
140 H(3)=INT(D/16)
150 D=D-H(3)*16
160 H(4)=D
170 FOR A=1 TO 4
180 H$(A)=CHR$(H(A)+48+7*ABS((H
(A)>9)))
190 NEXT A
200 PRINT"Hexadecimális szám:";
210 SET INK 10
220 PRINT H$
230 SET INK 8:GOTO 50
```

```
    1 REM Master-mind játék
   10 GRAPHICS 2
   20 PRINT"A gép 4 szint választ
   az alábbi 7 közül:"
   30 PRINT"fekete,kék,piros,lila
"
   40 PRINT"zöld,cián,sárga"
   50 PRINT"Önnek ki kell találni
a,mit rejtett el a gép"
   60 PRINT"A színeket a nevük ke
zdőbetűjével adja meg!"
   70 PRINT"Ha megnézte a szabály
okat,nyomjon meg egy tetszőleges
gombot!"
   80 A$=INKEY$:IF A$="" THEN 80
   90 SET CHARACTER 129,0,24,60,6
0,60,60,60,60,24,0
  100 GRAPHICS 16:SET BORDER 84
  110 SET PAPER 4:SET INK 10:CLS
  120 DIM T(7):RESTORE:RANDOMIZE
  130 S$="fkpłzcs"
  140 FOR N=1 TO 7
  150 READ T(N)
  160 NEXT N
  170 C=1
  180 REM 4 szín választása
  190 R$="":FOR N=1 TO 4
  200 A=INT(RND(8)):IF A=0 THEN 2
00:ELSE X$=S$(A)
  210 R$=R$&X$:NEXT N
  220 GOSUB 1500:REM fejléc nyomt
atása
  230 PT=0:ST=0:SET INK 15
  240 PRINT USING"##":C;
  250 PRINT TAB(5);
  260 GOSUB 1000:REM Válasz beolv
asása
  265 REM Poziciótalálat vizsgála
t
  270 FOR N=1 TO 4
  280 IF V$(N)<>R$(N) THEN 290
  282 PT=PT+1
  284 Y$=V$(N):FOR Q=1 TO 4
  286 IF V$(Q)=Y$ THEN V$(Q)="*"
  288 NEXT Q
  290 NEXT N
  295 REM Színtalálat vizsgálata
  300 FOR N=1 TO 4
```

```
 310 FOR M=1 TO 4
 320 IF V$(M)<>R$(N) THEN 370
 330 ST=ST+1:Y$=V$(M)
 340 FOR Q=1 TO 4
 350 IF V$(Q)=Y$ THEN V$(Q)="*"
 360 NEXT Q
 370 NEXT M,N
 380 PRINT TAB(11);
 390 IF PT=4 THEN PRINT AT 23,0:
"";:PRINT"Kitalálta":GOTO 490
 400 IF PT=0 THEN 430
 410 SET INK 0:FOR N=1 TO PT
 420 PRINT CHR$(129);:NEXT N
 430 IF ST=0 THEN 460
 440 SET INK 15:FOR N=1 TO ST
 450 PRINT CHR$(129);:NEXT N
 460 PRINT"":C=C+1:SET INK 10
 470 IF C<15 THEN 230
 480 PRINT AT 23,0:"Nincs több l
ehetösége!"
 490 PRINT"Akar újabb játékot? (
i/n)";
 500 INPUT X$:IF X$="i" THEN 100
 :ELSE END
 995 REM Válasz beolvasása
1000 V$="":FOR N=1 TO 4
1010 X$=INKEY$:IF X$=""THEN 1010
1020 F=0:GOSUB 1100
1030 IF F=1 THEN 1010
1040 V$=V$&X$
1050 NEXT N:RETURN
1100 FOR M=1 TO 7
1110 IF X$=S$(M) THEN SET INK T(
M):PRINT CHR$(129);:RETURN
1120 NEXT M
1130 F=1:RETURN
1495 REM Fejlèc
1500 PRINT TAB(4);"válasz";
1510 PRINT TAB(11);"talál."
1520 RETURN
2000 DATA 0,9,10,11,12,13,14
```

VIDEOTON

ELEKTRONIKAI VÁLLALAT
SZÁMÍTÁSTECHNIKAI GYÁRA