HOMELAB 4

KEZELÉSI UTASÍTÁS

TARTALOMJEGYZÉK

1. Műszaki adatok:

Mikroprocesszor: Z 80 A

Memória: Homelab 4/16 16 kbyte RAM
Homelab 4/64 64 kbyte RAM

Szoftver: 8 + 2 kbyte Basic interpreter
2 kbyte Assembler
4 kbyte duplapontos aritmetika (opcionális)

Perifériák:

Monitor:
- Video- és VHF TV csatlakozás (VHF–6–12 csatornán)
- 32 sor, soronként 32, 64 karakter átkapcsolási lehetőség
- Z 80 Busz csatlakozási lehetőség
- Párhuzamos port (16 bit) csatlakozási lehetőség-
  - egyben Centronics interface
- Kazettás magnó csatlakozási lehetőség
  (ajánlott típus: Junó MK 29).

Billentyűzet: 59 standard kiosztású, mechanikus működésű szilikon gumiérintkezős tasztatúra.

Tápegység: Homelab 4/16 típusnál +5 V 600 mA, +12 V/55 mA, –5 V/30 mA.
Homelab 4/64 típusnál +5 V/650 mA.

Hálózat: 220 V ±10% 50 Hz 20 VA.

Biztosíték: 100 mA.

Érintésvédelem: kettős szigetelés (II. oszt.)

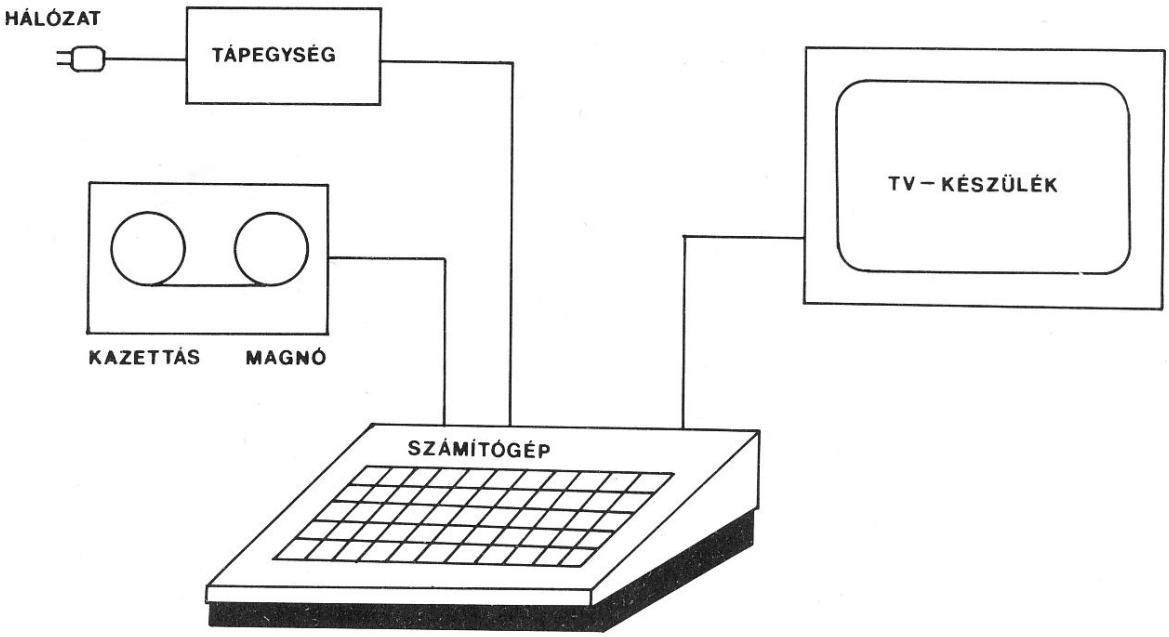
Fejlesztő: HOMELAB HARD-SOFT
ELEKTRÓNIKAI GMK

Gyártó: COLOR IPARI SZÖVETKEZET
7200 DOMBÓVÁR
Engels u. 7.
Telefon: 15–14, 15–15, 12–03.
Telex: 13–271.

Szervíz: RAMOVILL SZERVIZEK

2. Üzembehelyezés:

A Homelab 4 személyi számítógéppel az 1. ábrán látható alapkonfiguráció építhető ki.



1.ábra

A TV készüléket a géphez mellékelt 75 Ohm impedonciájú koaxiális kábellel kapcsolhatjuk össze a számítógéppel.

Lehetőség van az antennacsatlakozás helyett a TV video bemenetére csatlakozni. Ehhez megfelelően átalakított TV szükséges. Az átalakítást az ajánlások alapján szakemberrel végeztesse el, ha szükséges.

A magnetofont a hozzá tartozó átjátszókábel segítségével csatlakoztassuk a számítógéphez.

A tápegységen levő csatlakozót a tápegység feliratú aljzatba dugjuk.

Ezzel összeállt a rendszerünk.

A rendszer bekapcsolásának sorrendje a következő:

- TV
- magnetofon
- számítógép.

A számítógép bejelentkezését sipoló hang jelzi.

Amennyiben a számítógép önmagától nem jelentkezik be, úgy a Reset gomb benyomásával a gép működésbe hozható. Ha a számítógéphez VHF szinten csatlakozik a TV készülék, akkor állítsuk be a gép hátulján levő kapcsolót 32 karakteres állásba, és nyomjuk meg a Reset gombot. 64 karakteres kapcsolóállás esetén csak a Video szintű csatlakozáskor kapunk megfelelő minőségű TV képet.

A TV készülék folyamatos hangolócsavarjával 6—12 csatorna tartományában be tudjuk állítani a képet.

Ha a kép minősége nem megfelelő pl.: világos, fut, torz stb., a TV készülék kezelőszerveivel tudjuk beállítani a képet.

A TV kontrasztját és fényerősségét úgy szabályozzuk, hogy a szemünknek kellemes legyen, a betűk jól olvashatóan, tisztán látszódjanak. A TV hangerőszabályozóját célszerű min. állásba állítani.

Miután beállítottuk TV készülékünket a következő felirat jelenik meg a TV képernyőn (Homelab 4/16 gép esetén):

HOMELAB 4
16115 bájt szabad memória
HOMELAB BASIC 4.1.

Az Ok üzenet alatt a képernyő szélén egy kis négyszög alakú jel, a Cursor villog, ami jelzi, hogy a számítógép üzemkész állapotban van.

A Homelab 4 két változatban kerül forgalomba:

16 és 64 jelzéssel.

Az első 16 K RAM (memória) területtel rendelkezik, a másodikban pedig 64 K RAM van.

Ez utóbbinál a Basic számára felhasználható terület 48 K, és így a 64-es gép 48883 bájt szabadmemóriát jelez.

A géphez bármilyen típusú kazettás magnó használható, mely rendelkezik 5 pólusú átjátszócsatlakozóval, valamint képes legalább 500 mV-os jelet kiadni.

Lehetőleg számlálóberendezéssel ellátott magnót használjunk, hogy a rajta levő felvételeket könnyen visszakereshessük (ajánlott típus: Juno MK 29).

Az ajánlott magnótípus esetén felvételkor és lejátszáskor a hangerő és a hangszín szabályzó középállásban legyen.

A felvételt ALC (automata) állásban készítsük el.

3. Karbantartás

A számítógép tisztántartáson kívül egyéb karbantartást nem igényel.

A gépet száraz vagy enyhén nedves puha ronggyal tisztogassuk. Óvjuk a gépet ütődésektől, a szélsőséges hőmérsékletektől, valamint a nedvességtől.

Ne tegyük ki a tűző nap hatásának sem.

A tasztatúra nyomásra érzékeny, ne könyököljünk vagy tenyereljünk rá!

A TV-re és az egyéb perifériákra vonatkozóan tartsuk be azok karbantartási utasítását.

Biztonsági rendszabályok:

A számítógép és tápegysége II. érintésvédelmi osztályba tartozik, kettős szigetelésű. Védőföldelt hálózatba nem köthető! Csak a névleges 220 V eff. feszültségű aljzatról üzemeltessük a berendezést.

A számítógépet ne szedjük szét, ne nyúljunk a belsejébe!

A számítógép megbontása a garancia megszüntét vonja maga után.

A TV-re magnóra és a perifériákra vonatkozóan a gyártó által előírt biztonsági rendszabályok a mérvadóak.

A gép üzemszerű használatához minimum három konnektor szükséges. Célszerű több lyukú csatlakozó aljzatot használni.

Ne használjunk T dugót, vagy toldott vezetékeket.

Bekapcsoláskor mindig előbb a hálózati dugót csatlakoztassuk a hálózatba, majd utána kapcsoljuk be a berendezést.

A számítógép bármi nemű meghibásodása esetén forduljon a gyártóhoz, illetve a gyártó által megadott szervízhez.

4. A Homelab 4 Basic nyelve

4.1. Általános ismertető

A Homelab 4 személyi számítógépben levő Basic (HOMELAB BASIC 4.1.) alapjában véve kompatibilis a szabványokban lerögzített nyelvvel.

Azok az apróbb eltérések, amelyek mégis megvannak, bárki számára első olvasásra megtanulhatók.

Ezek a kiegészítések – kevés kivételtől eltekintve – a szabványos Basic-en túl, annak szintaxisát megtartva, új lehetőségeket nyújtanak a felhasználó számára.

A Homelab 4 Basic nyelve felülről kompatibilis a Homelab 3 és a Homelab 2. (Aircomp 16) elnevezésű gépek Basic-jével. Ez azt jelenti, hogy a korábbi gépeken készült programok lefuttathatók a Homelab 4-en, illetve az új gépen készült programok is ugyanígy átvihetők a régibe. Persze a régi gép nem fogja megérteni a Homelab 4 új utasításait.

A gépben ROM-ban elhelyzett Basic nyelv interpreter jellegű. Ez azt jelenti, hogy a programot csak futtatáskor értelmezi. Ebből következik, hogy a hibásan beírt programot éppúgy eltárolja, mint a jót.

A hiba csak akkor derül ki, ha a vezérlés rossz sorra adódott.

A programsorok hossza nincs korlátozva, több TV-soron keresztül is folytatódhat. Maximális hossza a képernyő méretének megfelelően 1023 vagy 2047 karakter.

A gépben a teljes képernyőre kiterjedő editor (szerkesztő) működik. Ez azt jelenti, hogy a képernyőn bárhol bármi átírható, törölhető, beszúrható. Ennek részletesebb működését a speciális karaktereket leíró fejezet tárgyalja.

A billenytűzet segítségével a karakterkészletben előforduló legtöbb jel megjeleníthető (F1, F2, jobb és bal Shift és egy betűgomb valamilyen kombinációjával. Lásd a karaktertáblázatot). A gép a lenyomott gombot hangjelzéssel nyugtázza.

A nyomvatartott gombot a gép folyamatosan ismétli!

Mielőtt rátérnénk a Homelab Basic részletesebb ismertetésére, felhívjuk a figyelmet, hogy ebből a gépkönyvből a Basic nyelv nem tanulható meg! Az alapok elsajátítására már sok jó szakkönyv megjelent, a gépkönyv ezeknek csak kiegészítése, innen csak a gép tájszólása tanulható meg.

Másik jótanácsunk, hogy a gépben semmi nem romlik el egy hibásan bevitt utasítás hatására (legföljebb a gép hibát üzen!). Ezért ha valami véletlenül még a gépkönyvből sem világos, akkor azt bátran próbálják ki!

A Homelab 4-ben a 10K Basic-en kívül megtalálható még egy egyszerű ASSEMBLER-DISASSEMBLER program is (2 k), és opcionálisan betehető a dupla pontosságú aritmetika (4 k) is. A gépkönyv ezek leírását is tartalmazza!

Megemlítjük még, hogy a Homelab 4 számítógép 8K Basic-kel is működőképes. Ekkor a *-gal jelzett utasítások és függvények nem használhatók, és a gép angol hibaüzeneteket ad.

A Basic ismertetését egy összefoglaló táblázattal kezdjük.

Az utasítások és parancsok kulcsszókészlete

BEEP	CALL	CONT	CUR	DATA
DELETE*	DIM	EDIT*	END	EXT
FOR	GOSUB	GOSUB#*	GOTO	GOTO#*
IF	INPUT	KEY	LIST	LOAD
MERGE*	MON	NEXT	NEW	ON
PLOT	POKE	POP	PRINT	READ
REM	REPEAT*	RESTORE	RETURN	RUN
SAVE	STEP	THEN	TO	UNTIL*
VERIFY*				

A felsorolt kulcsszavak a CONT kivételével utasításként és parancsként egyaránt kiadhatók. A CONT nem lehet utasítás, csak parancs.

Valsós függvények:

ABS(X)	LOG(X)	SIN(X)
ATN(X)	MAX(J1, J2 . . . . Jn)*	SGN(X)
COS(X)	MIN(J1, J2 . . . . Jn)*	SQR(X)
EXP(X)	MOD(X, Y)*	TAN(X)
FRA(X)*	PEEK(X)	USR(X)
FRE(X)	POINT(X, Y)	VAR(X)*
FSW(X, Y, Z)*	RND(X)	
INT(X)	ROUND(X, Y)*	

Stringfüggvények:

ASC (X$)	LFT (X$, I)
CHR $ (I1, I2 . . . . Jn)	MID $ (X$, I, J)
DEC (X$)*	RGH $ (X$, I)
FORM $ (X, Y, Z)*	STR $ (X)*
HEX $ (X)*	STRING $ (X, Y)*
LEN (X$)	VAL (X$)

Logikai műveletek:

AND	OR	NOT

Relációk:

>	=	<
>=	><	<=

Speciális változók:

INKEY	INKEY $	PRG*
CR	HM	PI

A Basic jelkészlete:

Számok: ø 1 2 3 4 5 6 7 8 9
31 db magyar nagy- és kisbetű
Műveleti jelek: + – / * ∧
Reláció jelek: > = <
Egyéb jelek: ! " # $ & % ' ( ) . , : ; ?

Egyéb grafikus jelek: lásd a karakterkészlet leírásánál a függelékben.
A *-gal jelölt utasítások és függvények csak a 10 k-s (bővített) Basic-ben értelmesek.

4.3. A Basic speciális karakterei

CR — CARRIAGE REUTRN. A begépelt sort zárja le.
Kódja: 13 ($ØD)

CLS — Clear. Shift CR-rel érhető el, a képernyőt törli.
Kódja: 12 ($ØC)

HOME — HOME. Shift Space-vel érhető el: a cursort a kép tetejére teszi.
Kódja: 15 ($ØF)

TAB — Tabulátor. Shift/↑ érhető el. Hatására a cursor új zóna elejére lép, és az útjában levő karaktereket törli. 1 zóna 8 vagy 16 karakterből áll.
Kódja: 14 ($ØE)

CURSOR — Mozgó karakterek. A CURSOR segítségével a képernyő bármely pontja elérhető, ott javítás végezhető vagy a szöveg újraírható. A CR hatására mindig az a sor kerül bevételre, amelyikben a CURSOR éppen volt. Itt logikai sor értendő, nem fizikai!
Figyelem! A javításról a gép csak a *CR megnyomásakor* értesül, anélkül a javítás hatástalan marad.
Kódok: ↓ 8 ($Ø8) ↑ 9 ($Ø9)
←10 ($ØA) →11 ($ØB)

INS — INSERT. A Shift/→ segítségével érhető el, új karakter beszúrására alkalmas. A CURSOR-tól jobbra eső szövegrészt eggyel jobbra lépteti, így a CURSOR helyén egy szabad karakter marad. Ismételt megnyomással újabb betűkhöz karakterek keletkeznek, amelyek helyébe új szöveg írható be.
Kódja: 6 ($Ø6)

DEL — DELETE. A Shift/← segítségével érhető el, s egy karakter törlésére szolgál. A CURSOR-tól jobbra eső szövegrészt eggyel balra lépteti úgy, hogy a CURSOR által kijelölt karakter elvész. Ha a CURSOR a sor végén van, az utolsó karaktert törli.
Kódja: 7 ($Ø7)

BELL — BELL. Shift/↓ érhető el. Hatására a gép rövid hangjelzést ad. Stringbe beépítve így fütty is kiprintelhető.
Kódja: 5 ($Ø5)

Egy Basic sorban tetszőleges számú utasítás lehet, ezek között kettőspont az elválasztó karakter.
Kódja: 58 ($3A)

Vessző. Lista elemeket választ el vagy PRINT utasítás formátumát határozza meg.
Kódja: 44 ($2C)

#	PRINT, INPUT és LIST utasításoknál OUTPUT, illetve INPUT eszközt jelöl ki. Kódja: 35 ($23)
'	Aposztrof. Feltételes listák elválasztó karaktere (lásd ON, IF), vagy a Basic-sor hátralevő részének letiltására használható. Az ' mögött álló utasítások – az IF és az ON kivételével – sohasem hajtódnak végre. IF vagy ON után ' nem szerepelhet szövegkonstansban! Kódja: 39 ($27)
"	Basic-ban a stringkonstansok elejét és végét jelzik. Az idézőjelek között levő , ' , ; : ? karakterek elvesztik előbb felsorolt jelentésüket, ilyenkor ezek is hozzátartoznak a szövegkonstanshoz. Kódja: 34 ($22)
?	Basic programokban helyettesíti a Print kulcsszót. Kódja: 63 ($3F)
%*	A mögötte álló *számkonstanst* binárisan értelmezi % mögött ezért csak Ø vagy 1 állhat. A bináris számkonstans hossza legföljebb 16 jegy lehet. Pl.: A = % 1ØØ11 (A = 19) Kódja: 37 ($25)
$*	A mögötte álló *számkonstanst* hexadecimálisan értelmezi. A $ mögött ezért csak Ø . . . 9 és A B C D E F állhat. A $ után álló jelsorozatból a gép az utolsó 4 jegyet értelmezi. Pl.: A = $ 3F39 (A = 16185) Kódja: 36 ($24)

4.4. Egyéb billentyűfunkciók

F1, F2	Funkciógombok. A többi gombbal egyszerre lenyomva a karaktertáblázatban felsorolt jeleket eredményezi.
F1/F2	A funkciógombok egyidejű megnyomása megállítja a Basic program futását. (Break) Csak Basic program futása közben hatásos!
ALT	ALTER. Ez a gomb a nagybetűs és kisbetűs írásmód felcserélésére szolgál, csak a betűkre van hatással. Ha cursor teli négyzet alakjában villog, akkor a leütött gombok nagybetűt adnak, és a shift vált kisbetűre. Az ALT ezt úgy változtatja meg, hogy egy üres négyzet fog villogni, és alapesetben kisbetű, shift-tel pedig nagybetű lesz. Ujabb ALT visszaváltja az előbbi üzemmódra és így tovább.
SPACE	Soremelésnél, ha teli a kép és ezt a gombot nyomva tartjuk, egyenként lassan lépteti a sorokat. Ha shift-tel együtt nyomjuk meg, teljesen leáll a kiírás, egészen addig, míg újra Shift/Spece-t nem nyomunk. Amíg áll a kép, addig is van lehetőség a sorokat egyenként léptetni a SPACE nyomogatásával. A Shift/CR ebben az állapotban egy teljes új képet hoz be. Mindezeknek listázásnál és folyamatosan keletkező eredményeknél van szerepe.

*Csak 10 k-s Basic esetén használható

4.5. Számábrázolás

A gép belső számábrázolása 4 bájtos lebegőpontos. Ennek megfelelően $+10^{38}$ és $+10^{-38}$ közé eső abszolút értékű számokat tud ábrázolni, 10^{-7} relatív hibakorláttal. A gép a számokat 6 jegyre kerekítve írja ki 999999 és 0.1 tartományban törtalakban, azon kívül pedig normál alakban.

A mantissza és a karakterisztika + előjele, valamint a tizedespont előtti érvénytelen nullák elhagyhatók.

Függvények vagy hosszabb számítások során a hiba halmozódik, ilyenkor a hibakorlát a megadottnál nagyobb is lehet!

A duplapontos aritmetika használatakor (ez külön bővítés!) a számok belső ábrázolása 16 jegyre nő és a gép 14 jegyet ír ki.

4.6. Változók

A Basic-ben használt változók azonosítója — típustól függetlenül — legfeljebb két betű lehet. A 30 nagybetűn kívül más karakter nem használható. (Az Á nem lehet változónév). Több betűből álló azonosítók esetében csak az első két betűt jegyzi meg és veszi figyelembe. A változók neve nem tartalmazhatja a kulcsszavak karaktersorozatát. A változók tartalmuk szerint valós vagy STRING típusúak lehetnek. Ez utóbbinál az azonosító után $ áll.

Szervezésük szerint a változók lehetnek egyszerű változók, vektorok és mátrixok. A vektorok és mátrixok (tömbök) azonosítója után zárójelben áll az egy-, ill. kéttagú indexkifejezés. A tömböket az első előfordulás előtt mindig dimenzionálni kell, függetlenül a tömb méretétől (nincs autodimenzió!)

Ugyanolyan típusú vektorok és mátrixok azonosítására nem használhatók azonos betűkombinációk, mert A (X, Ø) = A (X)! A legnagyobb tömbindex 255 lehet!

4.7. Stringek

Egy STRING hossza nem lehet nagyobb 255-nél és összesen 256 féle karakter fordulhat elő benne. Ezek zöme beírható a billentyűzetről is (a SHIFT, F1, F2 és egy jelgomb valamilyen kombinációjával), vagy a CHR függvény segítségével is. Van néhány karakter, amelynek speciális „nyomtatási képe" van. Ilyen, a karakterként is kiadható: INS, DEL, BELL, TAB, HOME, CURSOR mozgatások. Ezek a karakterek a billentyűzetről is beírhatók és beépíthetők a stringkonstansokba, mert az F1 vagy F2 hatására elvesztik speciális tulajdonságaikat, és csak a jelük látszik. Kinyomtatáskor természetesen mint speciális karakterek iródnak ki.

A stringekre értelmezve vannak a relációk és az összeadás is. Az összeadás egyszerű egymás után írást jelent.

A$ = B$, ha hosszuk egyenlő és elemeik rendre megegyeznek.

A$ < B$, ha B$-ben előbb van nagyobb kódu karakter, mint A$-ben, illetve A$ kezdőstringje B$-nek, (B$ az elején tartalmazza A$-et).

Pl.: „BÉLA" = „BÉLA"
„BÉNA" > „BÉLA"
„PAD" < „PADLÓ"

4.8. Kifejezések

A kifejezésekre ugyanez vonatkozik, mint a standard Basic-ben. A prioritási elv és a balról-jobbra szabály érvényesül.

Kifejezésekben használhatók az AND, OR, NOT műveletek és az összes relációk is. Az AND, OR, NOT logikai műveletek a számok egészrészének 2 bájtos fixpontos kettes komplemens alakjának bitjeire vonatkoznak. Pl.: 63 AND 16 = 16; 4 OR 2 = 6; NOT 1 = –2. A relációknak szintén értékük van. Ha egy reláció hamis, akkor az értéke Ø, ha igaz értéke 1. (A = 2 esetén; (A = 2) = 1; (A = 57) = Ø; (A > Ø) = 1).

4.9. Speciális Basic változók

PI	Csak olvasható változó. Értéke a π. PI = 3.14159
INKEY INKEY $	Ez a változó lehet valós és STRING típusú is. Mindkettő az éppen lenyomott billentyűt adja: INKEY a lenyomott gomb ASCII kódját, az INKEY $ pedig magát a karaktert. Ha nincs lenyomva semmi, INKEY = Ø és INKEY $ üres string. Az F1, F2 és SHIFT gombok lenyomását a gép külön nem veszi észre.
CR	COLOR. A képernyőn PLOT utasítással megjeleníteni kívánt pont színét határozza meg. Ha CR = Ø fekete, ha CR = 1 fehér a pont. A CR = 2 invertálást jelent. CR-t csak akkor kell állítani, ha az éppen benne levő szín nem megfelelő.
HM	HIGHEST MEMORY. Az ide beírt decimális szám a Basic számára felhasználható legmagasabb memóriahely címét jelenti. Ezzel a változóval gépikódú programkészletek és adatmezők számára rezerválható terület.
PRG*	Csak olvasható változó. Azt adja meg, hogy a lehetséges Basic lapok (lásd EDIT) közül éppen melyikben vagyunk.

4.10. Az utasítások ismertetése

4.10.1. BEEP A$

Hangkeltő utasítás. Elfütyüli a mögötte álló stringet, stringkifejezést vagy stringkonstansot. A stringben az egyes karakterek különböző ritmusokat és hangmagasságokat jelentenek.

ASC II kód	jelentés
Ø – 31	érvénytelen (mély hangokat ad)
32	Space. Ez a zárókarakter. Feltételenül szükséges az elfütyülendő string végén. Ha ez nincs, nem áll le a fütyülés!
33–63	A számok és a jelek. Ezek a ritmust határozzák meg, azt, hogy egy hang mennyi ideig szól. A 33 (!) a leggyorsabb, a 63 (?) a leglassabb ritmus.
64–255	A betűk és a grafikus karakterek. Ezek a hang magasságát határozzák meg. 64 a legmagasabb, 255 a legmélyebb hang.

Példák: 1. A$ = „#"
FOR I = Ø TO 9 : A$ = A$ + CHR$ (64 + RND(192)):
NEXT : A$ = A$ + „ _ " : BEEP A$.

2. BEEP CHR$ (40, 253, 209, 253, 209, 164, 164, 32)

3. BEEP „+ABCDEFGHIJKLMNO_"

A zenei hangokra vonatkozó táblázat a függelékben megtalálható. (8.2. fejezet).
A kiadott hangok frekvenciája

$57692/K\ [H_z]$, ahol $64 < K < 255$

A hangok hossza a

$4 \cdot (K-32) \cdot 10.24$ [ms], ahol $33 < K < 63$

képlet alapján határozható meg.

4.10.2. CALL $X_1, X_2, \ldots$

Gépikódú szubrutin hívások az X_1, X_2 stb. (decimális) címekre. A szubrutinhívások a címsorrendnek megfelelően egymás után következnek. A címek között vessző az elválasztó jel. Címnek nem csak szám, hanem aritmetikai kifejezés is írható. A szubrutin gépikódú RET utasítással térhet vissza a Basic-be.

ACPU regiszterei IY kivételével felhasználhatók.

Példa: CALL 416 ($1AØ monitorrutin meghívása).

4.10.3. CONT

BRK-val (F1 és F2 egyidejű megnyomása) megszakított program folytatását eredményezi. Ez csak akkor lehetséges, ha a BREAK üzenet óta még nem történt programsorbeírás, vagy bármilyen hiba üzenet.

Ellenkező esetben CN ERROR (NEM FOLYTATHATÓ!) üzenet keletkezik.

4.10.4. CUR X, Y

Pozícionáló utasítás. A PRINT és INPUT utasításokban helyezhető el a cursor pozíciójának beállítására, hatására, még mielőtt a kiírás megtörténne, a cursor az Y. sor X. karakterére áll be. Ha Y. hiányzik, akkor az adott soron belül áll be az X. pozícióra.
Y = Ø a legfelső sor, X = Ø a baloldali szélső karakterhely!

Néhány példa a használatára:

```
PRINT CUR 1Ø, 12 ; „GÉZA" , CUR 2Ø, 24; „BÉLA"
PRINT CUR 12; „ÉVA" , CUR 29, „JENŐ"
INPUR CUR 19, 2Ø; „MI VAN" , A$
```

4.10.5. DATA $X_1, X_2, \ldots$

Adatlista a READ utasításhoz. DATA után vesszővel elválasztva tetszőleges típusú kifejezések állhatnak. Ezen kifejezések felhasználásáról a READ-nél szólunk. DATA utasítás a programban bárhol elhelyezhető, de célszerű a program végére tenni.

*4.10.6. DELETE A–B**

A kulcsszó után egy címintervallum áll. A címintervallum a LIST-tel azonos módon értendő.

A DELETE a címintervallum által meghatározott programsorokat kitörli. Ha a DELETE után nem áll semmi, az összes sort törli.

Példa: DELETE 10 DELETE 100– DELETE 100–200

4.10.7. DIM A, B, . . .

Dimenzionálás. A, B, . . . stb. – tetszőleges típusú tömbváltozók. Az azonosító után zárójelben egy ill. két aritmetikai kifejezés áll, ami megadja a tömb maximális indexét (indexeit). Az alsó index mindig 0. Ugyanaz a változó egy programon belül nem szerepelhet kétszer DIM utasításban. Ebben a Basic-ben minden tömböt először dimenzionálni kell, automatikus dimenzionálás nincs!

A maximális tömbméretet a memória korlátozza.

Ha túl nagy tömböt dimenzionálunk, akkor OM ERROR (TUL KICSI A TÁR!) keletkezik. De a tártól függetlenül a maximálisan felhasználható index 255!

Példák: DIM A (19), B (24, 3), C (4), U$ (22)

*4.10.8. EDIT N**

A gépben egyszerre több – egymástól független – Basic program elhelyezése lehetséges. Ennek megfelelően parancsmód is több lehet. Ezeket Basic lapoknak nevezzük. Az EDIT N ezek között biztosít átmenetet.

Alap esetben a gép a 0 lapon van. Tegyük fel, hogy oda már írtunk egy programot.

Ha most kiadjuk az EDIT 1 parancsot, a gép kitörli a változótáblát és átlép egy másik lapra. Ott új, az előzőtől független program írható. (Tehát újra lehet használni ugyanazokat a sorszámokat). Ha itt listázunk, vagy itt adunk RUN parancsot, akkor az csak ezen a lapon levő programra lesz érvényes.

EDIT 2-vel újabb lapot nyithatunk, míg EDIT 0-val visszatérhetünk a 0 lapra (alaplap).

Tehát az EDIT N parancsmódban átteszi a vezérlést az N lapra. Ha már korábban megnyitottuk ezt a lapot, akkor nem töröl változótáblát. Ha ilyen számú lap még nem volt, akkor megnyitja a következő lapot, és törli a változótáblát.

(Figyelem! ha például 5 lap van nyitva, és EDIT 8-at adunk, akkor új lapot nyit, de az 6 lesz).

Megnyitott lapok megszüntetése nem lehetséges. Ha egy lap szükségtelen, egyszerűen csak törölni kell belőle a programot.

Hogy éppen melyik lapon dolgozik a gép, azt a PRG rendszerváltozó mutatja meg. Az alaplapon a PRG = 0.

A különböző lapokon elhelyezett programok közös változótáblával dolgoznak, de a programok egymástól függetlenek, és egyszerre csak az egyik lapon futhat program.

Az egyik lapon futó program átugorhat egy másik lapra, vagy meghívhat egy másik lapon levő szubrutint. Ezekre külön utasítások szolgálnak. (GOTO #, GOSUB #)

Magnóra tároláskor az összes megnyitott lap eltárolódik függetlenül attól, hogy melyik lapon adtuk ki a SAVE parancsot.

Ugyanígy a LOAD is beveszi az összes kitárolt lapot. A programban kiadott EDIT N megállítja a programot és a gép parancsmódra megy az N lapon.

4.10.9. END

Megállító utasítás. Ezt végrehajtva a programfutás befejeződik. Nem feltétlenül szükséges beírni a programba.

A program akkor is megáll, ha nincs több programsor.

Példa:
```
10 GOSUB 100
20 END
100 PRINT „NA LÁTOD" : REUTRN
```

4.10.10. EXT

Szoftver bővítéseket elsősegítő utasítás. Hatására BASIC hex 2800-ra ugrik, ahol a gép szoftverjének bővítései kezdődnek. Általában parancsként használatos.

Alapesetben 2800-ra az ASSEMBLER kerül.

4.10.11. FOR X = A TO B STEP C : . . . . . : NEXT

Ciklusképző utasítás. X a ciklus-változó, ami lehet vektor, vagy mátrix-elem is. A, B, C tetszőleges aritmetikai kifejezések. A ciklus-változó kezdőértéke A, B pedig a ciklus végét adja. C a növekmény. STEP C hiányozhat, ekkor a növekmény 1. Ezután az ún. deklaratív rész után következik a ciklus-mag, majd a NEXT cikluszáró utasítás. A ciklus-változó értéke mindig a NEXT utasítás alatt módosul a növekménnyel.

Ha C>0, a ciklus addig tart, mint X≥B. Ha C<0 a ciklus X<B-ig jut. Tehát a ciklus-mag legalább egyszer mindig végrehajtódik! Ha a ciklus lejárt, a NEXT-et követő utasítás hajtódik végre.

Ciklusok tetszőleges mélységig egymásba skatulyázhatók.

Figyelem! A NEXT utasítás után *nem szabad* odaírni a ciklus-változót. Ezt más Basic-ek megengedik, vagy megkövetelik, de itt ez hibás! A NEXT mindig azt a ciklust zárja, amit utoljára nyitottak meg.

Viszont egy NEXT-tel több ciklus is lezárható. A NEXT utasítás mögé írt minden vessző újabb ciklust zár le.

Példa: NEXT , , ekvivalens a NEXT : NEXT : NEXT-el.

FOR ciklus nélkül használt NEXT esetén Pp Error-t (STACK HIBA!) üzen a gép.

4.10.12. GOSUB X

Szubrutinhibás az X címre. X-re ugyanaz érvényes, mint a GOTO utasításnál. Lehetőség van egymás után több szubrutin meghívására is, ekkor a címeket vesszővel elválasztva kell felsorolni. A szubrutinok értelemszerűen a címek sorrendjében hajtódnak végre.

Példák: GOSUB 100 GOSUB 100, 200, 300

GOSUB #N, A.

Ez az autasítás az N. lapon levő program A. soránál levő szubrutint hívja. N. is és A. is lehet kifejezés (lásd GOTO). A szubrutin végén a RETURN utasítás hatására a futás visszatér az eredeti lap megfelelő sorába.

Példa: GOSUB #2, 100 a második lap 100-as szubrutinja.

4.10.13. GOTO X

Feltétlen vezérlést átadó utasítás. A program végrehajtása, az X sornál folytatódik. X lehet egy sorszám is, de lehet aritmetikai kifejezés is. Ezzel kiszámított GOTO képezhető. Ha X aritmetikai kifejezés, akkor nem kezdődhet számmal, vagy ha ez elkerülhetetlen, akkor zárójelbe kell tenni.

Példa: GOTO 100 GOTO a + 100 GOTO (2 /\ A)

GOTO #N, A

Ez az utasítás az N. lapon levő program A. sorára ugrik. N is A is lehet kifejezés a GOTO-nál megismert szabályok szerint.

Példa: GOTO #PRG + 2,1000 relatív két lapváltás után az 1000-dik sorra ugrik a program.

4.10.14. IF

A relációkról (lásd kifejezések) elmondottak alapján nem meglepő, hogy az IF tulajdonképpen az ON kétértékű megfelelője (lásd előbb ON). Hogy a BASIC kompatibilitása megmaradjon, a THEN szócskát megtartottuk, bár jelentése itt megegyezik az aposztroffal. További ' segítségével az ELSE funkciója valósítható meg. (Ekkor a szigorúan vett ON-ban a reláció 0 értéke helyett 2 kellene legyen. Ez az egyetlen különbség!) Mivel az IF és az ON logikailag azonos, ezért az IF-ben is kiemelhető utasítás, vagy írható utasítás csoport is. Az alábbiakban példák láthatók helyesen felírt IF utasításokra:

```
IF  A > D  THEN A = – A:B = A
IF  A < D  THEN A = Q' B = P
IF  A < D  'B = O : PRINT Z' S = ü : GOTO 100
IF  A = D  GOTO' 10' 100
IF  A = D  PRINT' A 'B
IF  A = D  PRINT' A : C = ?'B : E = B + 6
```

4.10.15. INPUT

Adatbeviteli utasítás. Legegyszerűbben talán úgy foglalható össze, hogy mindent, ami nem változó kiír, a változókat pedig megkérdezi, és a válasszal értéket ad nekik. De lássuk precízebben!

„A" legyen egy változó, „B" pedig vesszőt, pontvesszőt, CUR-t vagy nem változóval kezdődő tetszőleges kifejezést jelentsen. Az INPUT szó után A-knak és B-knek tetszőleges sorozata állhat. Ha a gép kiértékelés közben B-t talál, azt kiírja (pontosan ugyanúgy, mint a PRINT). Ha A-t talál, azt megkérdezi és az értéket a változónak adja. A „B", mint

említettük, nem változóval kezdődő kifejezés is lehet. Ha „B" aritmetikai, a gép kiszámolja a kifejezés értékét, és azt írja ki. Ha „B" string, akkor természetesen a string kerül ki. A pontosvessző és a vessző a PRINT-tel azonosan működik. Lássunk egy példát:

INPUT „GÉZA" ; G ; 12 + 8 – U ; " = " ; H , , "ÉVA", E

Ez a következőt csinálja: kiírja, hogy Géza, és megkérdezi G változó értéket. Utána kiszámítja a kifejezést és kiírja egy egyenlőségjellel együtt, majd beveszi H értékét.

A következő szöveg, két zónával beljebb kezdődik és az E változó értékét is egy következő zónában kérdezi meg.

Általános szabály, hogy minden változó bevételekor a gép a teljes beírt sort feldolgozza. Ezért nem lehetséges egy kérdésre több értéket felsorolni, és ezért van az is, hogy a gép minden változó bevétele után új sort is kezd.

Ha mégis szükséges, hogy két kérdés ugyanabban a sorban legyen, akkor a CUR utasítást kell használni, vagy bele kell építeni a kiirandó stringbe a cursormozgató karaktert is.

Példa: INPUT „Hónap:"; H „ " ↑ Nap: ";N
INPUT CUR 5, 12; "X értéke: "; X; CUR 25; 12; "Y értéke: "; Y
INPUT "ÁR = → → → → → Ft ← ← ← ← ← "; A

A cursormozgatások és a CUR segítségével tehát nem csak az adatbevitel elé, hanem mögé is tehetünk feliratokat!

Még egy szabály: az INPUT csak akkor tesz kérdőjelet, ha a bevétel előtt nem volt nyomtatás. Ha volt nyomtatás, akkor csak a cursor jelenik meg a kiírt szöveg után.

Az INPUT kérdésére üres sort válaszolva — stringváltozó esetén — üres string keletkezik, amíg aritmetikai változó esetén a gép újra kérdez.

Alapvető tulajdonság az is, hogy kifejezést is lehet válaszolni az input „kérdésére". Ilyenkor előfordulhat, hogy a begépelt kifejezés hibás és ezért kiértékelhetetlen. Ekkor a gép szintén megismétli a bevételt.

INPUT # X; A, B . . . .

Ez a INPUT-tal azonos módon működik, mindössze az a különbség, hogy az input-periféria nem a billentyűzet lesz. X szám vagy kifejezés azonosítja az input eszközt.

Ø-a billentyűzet. (Ez, mint az előbb is láttuk, lehagyható). További input perifériák kiszolgálására való rutinok a későbbi fejlesztések során készülnek el.

*4.10.16. KEY**

A KEY utasítás megváltoztatja a funkciós gombok jelentését. KEY utasítás után a betűk és az F1/F2 gombok kombinációjával a Basic legfontosabb kulcsszavai gombnyomásra bevihetők.

Ujabb KEY utasítás visszaállítja az eredeti állapotot, ahol a funkciós gombokkal a grafikus karakterek érhetők el.

A kulcsszavak elhelyezését a függelékben megtalálja (8.1. fejezet).

4.10.17. LIST

Kilistázza a programot. Ha utána egy szám áll, akkor csak azt a sort írja ki. Ha két szám áll kötőjellel, akkor a két szorszám közti területet írja ki. Itt a kettő közül bárme-

lyik szám hiányozhat, ekkor értelemszerűen az adott sortól ill. az adott sorig listáz. Ha a LIST után # áll, akkor az előzővel azonos módon, de a printerre listáz. (A printer csatlakoztatása a függelékben szerepel!)

Példa:	LIST 10–20	LIST # 10–20
	LIST 10–	LIST # 10–
	LIST –20	LIST # –20
	LIST 10	LIST # 10
	LIST	LIST #

4.10.18. LOAD „NÉV"

Kazettás magnóról programot olvas. A rekord nevét idéző jelek közé kell tenni. Ha a név hiányzik, akkor az első érvényes rekordot veszi be. Ha érvényes rekordot talál, kiírja a nevet. Ha a név nem egyezett, tovább keres. Ha nem volt név, vagy a név egyezett, beveszi a rekordot. A LOAD után a CR-t még azelőtt meg kell nyomni, hogy a magnón a fütty megszólal.

Ha a beolvasás hibás volt, de csak a program szövegében van a hiba, akkor ERROR üzenettel tér vissza.

Ha máshol volt a hiba, és a beolvasás teljesen használhatatlan, újra bejelentkezik a gép, és ekkor alapállapotba kerül a Basic is.

A LOAD parancsot lehetőleg a kép tetején adjuk ki, mert az utolsó sorban a soremelés miatt hibás lesz a név.

4.10.19. MERGE „NÉV" MERGE X, „NÉV"**

Uj lapot nyit, és kazettán eltárolt programot a Loaddal azonos módon betölti erre a lapra. Ha X hiányzik, vagy értéke 1, a kazettás magnóról olvas. X többi értéke további fejlesztésekhez van fenntartva.

*4.10.20. MON A$**

Ez az utasítás a mögötte álló stringkifejezésben megadott Monitor vagy Assembler parancsot hajtja végre.

A string szövegének pontosan úgy kell kinéznie, mintha a Monitorban adták volna ki. A Monitor-parancs végrehajtása után visszatér a Basic-ba.

Példák: MON "T 3000"
MON "T" + HEX$ (A) + " _ "
A$ = "D 3000" : MON A$

4.10.21. NEW NEW A

Alapállapotba hozza a Basic-et. Ez azonos lesz a bekapcsolás utáni állapottal. Ha utána egy aritmetikai kifejezés is áll, akkor ez annak a memóriahelynek a decimális címe lesz, ahol a Basic-program kezdődni fog. Ekkor $40A0 és a megadott cím közötti memó-

riaterület szabadon marad gépi kódu felhasználások számára. Az így beágyazott gépi programok SAVE-nél a BASIC programmal együtt felkerülnek a kazettára, és Load-dal egyszerre betölthetők.

4.10.22. ON X

Szokásos alakja ON X GOTO 10, 20, 30 MÁS GÉPEKEN!!!

Az eddig használt listáknál a „ , " elválasztó jel mindig minden listaelem figyelembevételét előírta. Pl.: PRINT A, B, A(X,Y) GOSUB 10, 20. Az ON utasításnál viszont a listának csakis egyetlen (persze X-től függő) eleme számít. Az ON utáni lista tehát egy exkluzív lista, amely elemeinek elválasztására új jelet érdemes bevezetni.

Ez az aposztrof. E szerint a helyes formátum a HOMELAB–BASIC-ben ON X GOTO' 10' 20' 30 lesz. (Itt az első listaelem előtt is kell aposztrof.) Ez az utasítás az X-ediknek felsorolt címre ugrik el. Itt most $X < 1$ és $X \geqslant 4$ esetén a következő sorban folytatja a végrehajtást. X értéke lehet tört is, akkor a gép először egészrészt képez belőle.

Ha alaposan megnézzük az ON után a GOTO utasítás szinte "ki van emelve", és az utána következő exkluzív lista az argumentuma. Ennek analógiája bármely utasítás – kivéve az értékadás – kiemelhető ON után.

Példa: ON X PRINT' A' B' C

X értéknek megfelelően A, B vagy C értéket nyomtatja ki. Ha X különböző értékeire különböző utasításokat kell végrehajtani, tehát az utasítás nem emelhető ki, akkor az ON X után egy exkluzív utasítás-lista is állhat.

Példa: 10 ON X ' PRINT A : C = 3 ' PRINT D ' FOR J = 1 TO 7 : U (J) = 0 : NEXT

20 GOSUB 100 . . .

Ezt a hagyományos módon a következő programrészlet oldja meg:

```
10   ON X GOTO' 1000' 1100' 1200
20   GOSUB 100 . . .
1000 PRINT A : C = : GOTO 20
1100 PRINT D : GOTO 20
1200 FOR J = 1 TO 7 : U (J) = 0 : NEXT : GOTO 20
```

Az exkluzív utasítás listában az X-nek megfelelő utasítás-csoport hajtódik végre, de az aposztrofhoz érve a program-végrehajtás abbamarad, és a következő sornál folytatódik. (Az előző példában a program minden X. mellett a 20. sorban folytatódik).

Mivel a programsor hossza nincs korlátozva, az ' bevezetésével az ON utasítás határtalan lehetőségeket nyit a programozó számára. Komplett eljárások írhatók meg egyetlen sorban.

Egyetlen kikötés, hogy az ON utáni utasítás-lista nem tartalmazhat újabb ON utasítást, vagy '-os IF-et, (lásd IF-nél), és a benne előforduló szövegkonstansokban sem szerepelhet aposztrof.

4.10.23. PLOT X,Y

A CR változó által meghatározott "színű" pontot tesz ki a képernyő (x,y) koordinátájú pontjára. A (0, 0) pont a bal alsó sarokban van. X a vízszintes, y a függőleges koor-

dináta. Az utasítás CR értékét nem változtatja meg. A koordinátákra nézve:

$0 \leqslant X \leqslant 127$ vagy 63; $0 \leqslant Y \leqslant 95$

4.10.24. PKE I, J, K . . .

J értékét az I című memóriahelyre teszi, K értékét az I + 1 helyre és így tovább. I (a cím) 0 és 131072 közé eső szám lehet. Ha I kisebb mint 8192, akkor a POKE nem memóriahelyre tesz, hanem az I címmel gépikódu OUT utasítást végez (lásd a Z 80 processzor belső rendszerét!). Igy output perifériák kezelhetők le. Ha I 65536 és 131072 között van, akkor a POKE a második memórialapra szól. (lásd 7.1. fejezet) 16 k-s gép esetén ez azonos az elsővel, de 64 k-s gépnél további 16 k Ram, a Videoram és a keyboard érhető el itt. Ebben az esetben a valódi memóriacím I—65536 lesz. (Tehát pl.: a második lap 40 000 memóriahelyét I = 105536 Poke-címen lehet beállítani. J, K . . . . stb. értéke 0 és 255 közötti szám lehet.

A POKE utasítás J, K . . . stb. értékeiből először egészrészt von.

4.10.25. POP

Ha FOR ciklusból, vagy Basic szubrutinból NEXT vagy RETURN utasítás nélkül akarunk kilépni, akkor ezt POP-utasítás után tehetjük meg.

A ciklus és a szubrutin hívás eltárol egy visszatérési címet a stack memóriába. NEXT és RETURN utasításnál ezeket a gép mindig előveszi a stackból, és innen tudja, hova kell visszatérni. Ha tehát többször kilépünk NEXT vagy RETURN nélkül, a stack lassanként megtelik. Az is előfordulhatna, hogy egy félbehagyott ciklus keletkezik, és egy későbbi NEXT nem azt a ciklust zárja le, amit kellene.

Ezt akadályozza meg a POP utasítás. Eliminálja a stack legfelső értékét.

Példa:
```
FOR I = 0 TO10 : IF A (I) > 100 THEN POP : GOTO 100
.
.
.
NEX
```

A (I) > 100 esetén nem folytatja tovább a FOR ciklust

4.10.26. PRINT

Kiíratő utasítás. A kulcsszó után a nyomtatási lista áll. Ez kifejezésekből, változókból, a CUR-ból, szám ill. stringkonstansokból állhat. A nyomtatási lista elemei között vessző és pontosvessző az elválasztó jel. Ezek egyben a kiírási formátumot is meghatározzák.

A pontosvessző a listaelemek egyszerű egymás után írását eredményezi. A vessző a következő zóna elejére állítja a kiírót. Ebben a gépben 1 sor 4 zónából áll, amelyek egyenként 8 vagy 16 karaktert tartalmaznak.

Ha a lista után nem áll semmi, az a kiirás után egy soremelést eredményez.

Üres PRINT utasítás egy sort emel. A PRINT kulcsszót a kérdőjel teljes mértékben helyettesíti, elég tehát csak azt írni PRINT helyett, a gép viszont listázáskor lefordítja a kérdőjelet PRINT-té.

PRINT # X;

Az OUTPUT perifériák kezelésére szolgáló utasítás.
Az X szám vagy kifejezés, az Output-eszköz száma.
Ø a TV disply-t jelenti (ua. mint a normál PRINT utasítás).
1 a Printert-t jelenti (a gépbe beépített centronics interfacen keresztül).
A további output eszközök kezelését az INPUT-hoz hasonlóan szintén külön kell megírni.
A CUR hatása nem jelenik meg a printeren, a CUR mindig a dispay-en érvényes!

4.10.27. READ $X_1, X_2, \ldots$

A DATA utasítás párja. READ hatására X_1, X_2 stb. változók rendre felveszik a DATA utasítások után felsorolt értékeket. Ezt úgy lehet elképzelni, hogy az összes DATA utáni adatokat sorban betesszük egy táblázatba, és a READ utasítással egyenként olvassuk őket. A READ-nak van egy „mutatója", amit minden értékadás után eggyel tovább állít. Ujabb READ utasítás onnan folytatja a táblázat olvasását, ahol az előző abbahagyta.
Ha a DATA után álló kifejezés hibás, akkor a hibaüzenet a READ sorában keletkezik!
Ha kevesebb az adat, mint a READ, akkor OD Error-t (több READ mint DATA) üzen.
Pl. egy tömb feltöltése DATA-ban megadott értékekkel:

```
FOR I = Ø   TO1Ø : READ A (I) : NEXT
```

4.10.28. REM

A REM után megjegyzések helyezhetők el a programban. Végrehajtáskor a gép ezeket a szövegeket figyelmen kívül hagyja.

Példa: 1 REM A programot készítette: Kovács Béla

*4.10.29. REPEAT: . . . . . : UNTIL R**

Ez a FOR- NEXT-hez hasonló ciklusszervező utasításpár.
A REPEAT és UNTIL közötti programrészletet (ciklusmag) addig ismétli, amíg az UNTIL után megadott R reláció értéke hamis. Ha a reláció értéke igaz, akkor az UNTUL után folytatja a végrehajtást.

Példa:
```
1Ø A = Ø
2Ø REPEAT : PRINT A : A = A + 1 : UNTIL A = 1Ø
```
ez a programrészlet ekvivalens a következővel:
```
1Ø A = Ø
2Ø PRINT A : A = A + 1 : IF A 1Ø THEN GOTO 2Ø
```
(Ez a példa csak formailag mutatja a REPEAT—UNTIL működését. Valódi haszna csak bonyolultabb esetben látszik).

4.10.30. RESTORE RESTORE A

A DATA-val definiált adatmező elejére állítja a READ „olvasó-mutatóját". Ha a RESTORE után egy kifejezés áll, akkor azt kiszámolja, és READ „olvasó-mutatóját" ennek a sornak az első DATA-jára állítja. Ha nincs DATA a sorban, akkor az ezután következő első DATA-ra áll.

4.10.31. RETURN

Visszatérés szubrutinból, GOSUB-utasítás párja.
GOSUB nélküli RETURN esetén Pop ERROR (STACK hiba!) keletkezik.

4.10.32. RUN RUN A

Elindítja a programot az A. sornál. Ha A hiányzik, akkor a legelső sornál indul a program.

Figyelem! A RUN törli a változótáblát! Ha ez nem engedhető meg (pl.: program vagy adathiba miatt leállt program továbbfuttatása esetén), akkor a GOTO utasítást kell használni.

4.10.33. SAVE „NÉV"

Program eltárolása kazettás magnóra. Idézőjelek között a rekord nevét kell beírni. CR megnyomása előtt a magnót kell elindítani felvétel állásban. Tárolás alatt a gép fütyül.

A tárolás $4016-tól a BASIC TEXT végéig tart.

4.10.34. VERIFY „NÉV" VERIFY X, „NÉV"*

Ezzel az utasítással kazettára felvett programokat lehet ellenőrizni. (Gépi kódu felvételt is!).

A VERIFY a LOAD-hoz hasonlóan működik. „Elolvassa" a kazettát, de nem tölti be a memóriába, csak összehasonlítja azzal. Ok üzenettel tér vissza, ha nem talál különbséget. Ha volt hiba, „értelmetlen" hibaüzenetet ad.

A VERIFY után a név hiányozhat, akkor az éppen következő programot ellenőrzi.

4.10.35. LET

Ilyen utasítás nincs a Basic-ban, az értékadás egyszerűen A = x alakú, ahol A egy változó, x pedig egy kifejezés. A LET szó nem is szerepel a kulcsszavak között!

Beírása hibát eredményez!

ABS (X) — X kifejezés abszolút értékét adja.
ABS (1) = 1; ABS (–2) = 2

ATN (X) — X kifejezés arcus tangensét adja.
A függvény értéke radiánban adja a szöget.

COS (X) — X kifejezés cosinusát adja; X értéke radiánban értendő.

EXP (X) — e-t, a természetes logaritmus alapját; az X kifejezés által adott hatványra emeli.

FRA (X)* — X törtrészét adja. FRA (X) = X–INT (X)
Pl.: FRA (PI) = .141593
FRA (–PI) = .858407

FRE (X) — Megadja a szabad memóriaterületek nagyságát. X értéke közömbös, de nem hiányozhat.

FSW (X, Y, Z)* — Az első forma X függvényében Y vagy Z kifejezés.

FSW (X, Y, Z, U)* — értékét adja.
Ha X = Ø akkor az FSW értéke Y lesz
Ha X = Ø akkor az FSW értéke Z lesz

a második formában ugyancsak X függvényében ad értéket:
Ha X<Ø akkor az FSW értéke U lesz
Ha X = Ø akkor az FSW értéke Y lesz
Ha X>Ø akkor az FSW értéke Z lesz

Példa: a következő program az A tömb legnagyobb elemét keresi meg

```
T = Ø : FOR I = Ø TO 1ØØ
T = FSW (A(T) > A(I), I, T) : NEXT
```

T-ben a legnagyobb értékű elem indexe lesz.

INT (X) — X kifejezés egész értékét adja.
INT (4,5) = 4; INT (4) = 4; INT (–3.2) = –4

LOG (X) . — X kifejezés természetes alapú logaritmusát képezi.

MIN (11, 12, . . . In)* Ezek a függvények az argomentumban felsorolt
MAX (11, 12, . . . In)*kifejezések értékei közül a legkisebbet ill. a legnagyobbat adják.
Pl.: MIN (4, 7, 3, 9) = 3

MOD (X, Y)* — X, Y = Ø tetszőleges számok. Ekkor MOD (X, Y) = X–ABS(Y) * INT(X)ABS(Y)) vagyis X „maradékát" adja Y-nal osztva.
Pl.: MOD (26,3) = 2

PEEK (X) — Megadja az argumentumába írt kifejezés (X) számértékének megfelelő című memóriahely tartalmát.
Ha X < 8192 ($2ØØØ), akkor a PEEK nem memóriaterületet szólít meg, hanem az adott címmel IN gépikódu utasítást hajt végre (lásd a Z 80 belső rendszerét).
Ily módon INPUT perifférák kezelhetők (több mint 8ØØØ darab!)

Ha X nagyobb mint 65536, akkor X értékéből az a szám levonódik, és a PEEK a második lap megfelelő címét olvassa (mint a POKE utasításnál!)

POINT (X, Y) — Az X, Y koordinátájú képpont színét adja meg.
Értéke Ø, ha a pont fekete, 1, ha fehér, A (Ø, Ø) pont a képernyő bal alsó sarka, és Ø ⩽ X 127 ⩽ vagy 63; Ø ⩽ Y⩽ 95.

RND (X) — Véletlen számokat előállító függvény.
Ha X > Ø akkor a véletlen generátor a (Ø, x) intervallumba eső véletlen számértéket ad.
Pl.: INT (RND (6) + 1) a dobókocka dobásait szimulálja.
Ha X < Ø akkor ez a negatív szám a véletlen generátor kezdőértéke lesz. Ugyanarra a számra a gép mindig ugyanazt a véletlen sorozatot generálja.
A kezdőértéknél ajánlatos minél több jegyű számot (pl. egy törtet) beírni!
Pl.: A = RND (–1/PI)

ROUND (X, Y)* — Az X kifejezés értékét a 10^{-Y} helyiértékre kerekíti.
Pl.: ROUND (PI, +4) = 3.1416
ROUND (1532, –2) = 1500

SGN (X) — X kifejezés előjelét adja. Értéke + 1,Ø,–1 lehet.
SGN (–3)=–1, SGN(Ø)=Ø, SGN (4.1)=1

SIN (X) — S kifejezés sinusát adja. X értéke radiánban értendő.

SQR (X) — X kifejezés négyzetgyökét adja. X értéke nem lehet negatív szám.

TAN (X) — X kifejezés tangensét adja. X értéke radiánban értendő.

USR (X) — Gépikódu szubrutinhívást tesz lehetővé adatátvitellel. A szubrutin címét előzőleg POKE utasítással kell beállítani az USR-vektorba.
A gépikódu szubrutin magas címbájtjának helye 16385, az alacsonyé pedig 16384.
X egészrésze 2 bájtos fixpontos alakban a HL regiszterpárba kerül.
RET gépikódu utasítás hatására a gépikódu rutin visszatér a Basic-ba, és USR (X) értéke a HL-ben levő szám lesz.
A CPU összes regisztere felhasználható, IY kivételével.
Pl.: a $ 4567 kezdőcímű rutinhoz a beállítás:
POKE 16384, $67, $45 : A = USR(B)

VAR (X)* — X tetszőleges típusú változó azonosítója.
A VAR függvény megadja, hogy az X változó értéke hol van tárolva memóriában.
Ha X még nem kapott értéket, akkor a VAR értéke Ø lesz. REAL típusú változó esetén a VAR által adott címen kezdődő 4 bájt a változó értéke.
Ez a BASIC rendszeréről szóló fejezetben leírtak szerint értendő.
String változónál csak 3 bájt értékes.
Az első a string hosszát adja, a második kettő pedig azt mutatja meg, hogy hol kezdődik a string a memóriában.
Figyelem! Parancsmódban a változók helye állandóan változik, ezért a VAR függvényt csak programban érdemes használni!
Példa: Az itt leírt kifejezés Basic-ben reprodukálja a V változó

értékét W-ben:

```
C = VAR (V)
W = ((( PEEK (X+1) or 128)* 256 + PEEK(C+3))+
PEEK (C+2)(256)*Sgn (127.5–PEEK(X+1))
* 2 /\ (PEEK(C)–144)
```

4.12. String függvények

ASC (X$)

Megadja a X$ első karakterének kódját. Ha ez az karakter az ASCII készlet eleme, akkor a kód egyben ASCII kód is.

CHR$ (I_1, I_2 . . . . In)

Azt a stringent adja meg, amely karaktereinek kódjai rendje: I_k .I lehet speciális karakterek kódja is.

Pl.: ASC$ (CHR$ (I)) = I.

DEC (A$)*

Ha A$ tartalma egy HEX szám ASCII-képe, akkor DEC megadja ennek decimális értékét. (A HEX szám kettes komplemens kódban értendő).

Az A$ elején levő betűközöket a DEC figyelmen kívül hagyja. Ha a HEX szám 4 karakternél hosszabb, csak az utolsó 4 jegyet veszi figyelembe.

Példa: DEC ("EFE5") = –4123
DEC ("F") = 15

FORM$ (X, Y, Z)*

A Z kifejezés értékének formatált képét adja meg.

X az egészrész, Y pedig a törtrész számjegyeinek számát adja.

Az egészrészt és a törtrészt a tizedesjel választja el, és további karakter a szám előtt az előjel. Igy a FORM$ egy X + Y + 2 hosszúságú stringet ad.

Ha a Z értéke már nem fér el ebben az ábrázolási tartományban, akkor az előjel függvényében csupa + vagy csupa – string keletkezik.

Figyelem! A FORM$ nem kerekít!

A FORM$ szerkezetét az X és az Y értékeken kívül egy további paraméter is szabályozza. Ez a HEX 4042 (16450) címen levő rendszerváltozó, ami 6 bitjében a következőket jelenti:

x	x	5.	4.	3.	2.	1.	Ø.

$ 4042
dec 16450Ø

bit

Ø Ha Ø, akkor a + előjelet nem írja ki (helyette betűköz áll).
Ha 1, akkor a + előjelet is kiírja.

1 Ha Ø,,akkor tizedespontot ír.
Ha 1, akkor tizedesvesszőt ír.

2 Ha Ø, akkor az értéktelen nullákat nem írja ki a törtrészben.

3 Ha 1, akkor Ø törtrész esetén kiírja a tizedesjelet, és egy nullát.
Ha Ø, akkor sem tizedesjelet, sem nullát nem ír.
A 2. bit prioritása nagyobb!

4 Ha Ø, akkor a nulla egészrészt nem írja ki (csak egy tizedesjelet)
Ha 1, akkor egy nullát ír a tizedesjel elé.

5. Ha 0, akkor az előjelet a bal szélső helyre írja.
   Ha 1, akkor az előjelet közvetlenül az egészrész elé írja.
   Példa:POKE$4042,%00001011 esetén
   FORM$ (3,3,5) = "+ _ _ 5,0 _ _ "
   POKE$4042,%00110001 esetén
   FORM$ (3,3,1/2) = " _ _ +0.5 _ _ "

HEX$ (X)*

Ez a függvény előállít egy 4 karakteres stringet, ami az X kifejezés értékének hexadecimális ábrázolású képe lesz $-2^{15} < X < 2^{16}$

LEN (X$)

X$ hosszának számértékét adja meg.

LFT$ (X$, I)

X$ első I karakteréből álló stringet adja meg.

MIND$ (X$, I, J)

X$ I-edik karakterénél kezdődő és J darab karakterből álló stringet adja meg.

RGH$ (X$, I)

X$ utolsó I karakteréből álló stringent adja meg.

STR$ (X)

Azt a stringet adja meg, amelyik az X kifejezés értékének nyomtatási képe.

Pl.: X = 3.1 STR$ (X) = "__ 3.1"

STRING$ (X, Y)*

Ez a függvény egy X hosszúságú stringet ad, ami csupa Y kódu karakterből áll.
Ha Y hiányzik, akkor az X hosszúságú string a betűhöz karakterből fog állni.

VAL (X$)

Ha X$ egy számot ábrázol, vagy aritmetikai kifejezés stringképe, akkor a VAL (X$) ennek a számértékét adja.

Pl.: A = 3 és B = 4 esetén VAL ("A * 4+64 * B") = 268, vagy
VAL ("238.45") = 238.45 , VAL ("") = 0

4.13. Hibaüzenetek

BS BAD SUBSCRIPT: Méreten kívül eső tömbindex
CN CONTINUE ERROR: Nem folytatható a futás
DD DOUBLE DIENSIONED VAR: Kétszer dimenzionált változó
IQ ILLEGAL QUANTITY: Az argumentum kívül esik az értelmezési tartományon
OD OUT OF DATA: Több a READ utasítás, mint a DATA
OM OUT OF MEMORY: Elfogyott az engedélyezett memóriaterület
OV OVERFLOW: Túlcsordulás, ábrázolhatatlanul nagy eredmény keletkezett
PP POP ERROR: NEXT FOR nélkül, RETURN GOSUB nélkül vagy POP FOR ill. GOSUB nélkül
SL STRING TOO LONG: 255-nél hosszabb STRING keletkezett
SN SINTAX ERROR: Szintaktikus hiba
TM TYPE MISMATCH: Az előfordult kifejezés típusa nem megfelelő

US UNDEFINED STATEMENT: Nem létező sorra történt hivatkozás
/Ø DIVISION BY ZERO: Ø-val történt osztás.

*Magyar hibaüzenetek**

BS Rossz tömbindex!
CN Nem folytatható!
DD Ujra méretezett tömb!
IO Nem értelmezhető szám!
OD Több READ, mint DATA!
OM Túl kicsi a tár!
OV Túl nagy szám!
PP Stack hiba!
SL Túl hosszú szöveg!
SN Értelmetlen!
TM Rossz adattípus!
US Hivatkozás nem létező sorra!
/Ø Ø-val való osztás!

5. Gépikódu monitor és Assembler

A gépikódu programozás elősegítésére két szint van a Homelab 4 számítógépben. Az egyik az úgynevezett MONITOR, amely minden gépben benne van.

Ez csupán az alapvető funkciókat látja el:

Memóriaterületet *listáz,* memóriába *adatot tölt,* memóriaterületet kazettára *kitárol* és a programot a megadott címtől *elindítja.*

A monitor Basic-ből CALL 892 (decimális) hívható be.

A másik szint az ún. ASSEMBLER. Ez gépi kódu programok egyszerű oda- és visszafordítását teszi lehetővé, és kiegészül még néhány hasznos memóriakezelő paranccsal.

Az ASSEMBLER csak a 12 k ROM-ot tartalmazó gépekben van benne!

(A továbbiakban ezeket az utasításokat csillaggal jelöljük meg).

Az ASSEMBLER $ 2800 címen kezdődik, behívása legegyszerűbben a BASIC EXT utasításával történhet.

Az ASSEMBLER szint természetesen tartalmazza a MONITOR parancsait is. Mindkettőben minden szám Hexadecimálisan értendő, és az egyes parancsokat egy-egy betű jelenti. Természetesen itt is használható a Basic-ben megismert képernyőeditor. Ezt itt, mint majd látni fogjuk, különösen jól lehet használni.

A Basic-be való visszatérés mindkét szintről a GØ paranccsal történhet.

Figyelem! Gépi szinten nagyon könnyű „megbolondítani a számítógépet. Ezért alaposan át kell gondolni, hogy a memóriaterület mely részét szabad használni. Ebben ad eligazítást a gép belső rendszeréről szóló fejezet.

Természetesen akkor sincs komoly baj ha a gép „megbolondul", legföljebb elveszik belőle a beírt program.

A gépikódu programozás rejtelmeivel itt természetesen nem foglalkozhatunk, erre vonatkozóan jó néhány szakkönyv eligazít. Ugyanígy nem térhetünk ki a Z 80 mikroprocesszor utasítás-készletének ismertetésére sem. Ezeket ismertnek tételezzük fel, tehát aki nem tud valamit, itt álljon meg és nézze át a szakirodalmat.

5.1. A parancsok:

D HEX formában kiír egy memóriaterületet a képernyőre.
: Memóriatartalmat lehet beírni HEX formában.
S Memóriaterületet lehet kitárolni kazettára.
G Programot indít el.
F Egy adott területet feltölt egy megadott karakterrel.
M Egy memóriaterületet átmásol egy megadott helyre.
C Két memóriaterületet összehasonlít.
T Assembly formában kiír egy memóriaterületet. (visszafordítható)
; Gépikódu programot lehet beírni Assembly formában. (odafordítható)

Ha # áll egy parancs előtt, amelynek van ouputja, akkor az a printerre fog vonatkozni. Ilyenek a D, a C és a T parancsok.

Az itt felsoroltaktól eltérő parancs-betűkre a gép ERR hibát üzen.

Visszatérés Basic-be GØ paranccsal; vagy a RESET gomb megnyomásával.

5.1.1. D X Y vagy #D X Y

Memóriaterületet listáz X címtől Y címig.

Hogyha a végcím (Y) hiányzik, akkor a D hatására – a képmérettől függően – 128 vagy 256 bájt iródik ki, soronként 8 vagy 16 bájt. A sorok elején a kettőspont után a sor első bájtjának címe van.

Ha D után nem áll szám, onnan folytatódik a kiírás, ahol legutóbb abbamaradt. A Ø kezdőcímű listázás nem lehetséges, ekkor a gép nem veszi figyelembe az X címet.

Ha a D előtt # áll, akkor az előbbi szabályoknak megfelelően a printerre listáz.

5.1.2. :X P Q R . . .

Memóriaterületet tölt be. X az a cím, ahová az első számot (P) teszi. A Q R . . . stb. számok egymás után a soronkövetkező címekre kerülnek be. A CR megnyomásakor mindig kiírja a következő címet és új adatot vár. Üres sorra visszatér a monitorba.

A cím és az első adat között egy betűköz áll. További betűköz már nem szükséges, a számok sorban, kétjegyenként töltődnek be. Idegen (nem space vagy hex szám) karakterre a betöltés abbamarad. Mivel a D parancs a lista elé kettőspontot ír, ezért az általa listázott sorok cursor-ral javíthatók. CR-re a javított sor töltődik be!

5.1.3. S X Y Z

Kitárolja kazettás magnetofonra az X-től Y-ig terjedő memória-területet. A rekord neve Z lesz. A név a második címet követő első nem space karaktertől a sor végéig tart.

Kitárolás közben a beépített hangszóróból a magnetofonra felvett hang hallható. A monitorból kitárolt rekord formátuma azonos a Basic-évei, ezért a beolvasás mindig Basic-ben LOAD-dal történik.

5.1.4. G X

Programot indít el az X címtől.

A program végén, ha csak valami különös ok nincs, a monitorba célszerű visszatérni egy RET utasítással.

*5.1.5. F X Y Z **

X kezdőcímtől Y-ig feltölti a memóriát (még Y-t is) Z-vel. Z tehát egy kétjegyű szám. Ha ennél többet írunk, csak az utolsó két jegyet értelmezi és tölt be.

*5.1.6. M X Y Z **

X-től Y-ig (Y-t is!) terjedő részt átmásolja Z helyre. Átmásolás után Z tartalma ugyanaz lesz, mint ami az X-é. Z + 1-é, min X + 1-é, stb.

Z-re nincs semmilyen megkötés, bárhová át lehet másolni. (A gép „megnézi" Z viszonyát X-hez, ill. Y-hoz és ennek megfelelően felülről lefelé vagy alulról fölfelé haladva másol. Ez biztosítja, hogy másolás közben az eredeti tartalom nem vész el).

5.1.7. *C X Y Z vagy #C X Y Z **

Az X-től Y-ig terjedő területet (y-t is!) összehasonlítja a Z-nél kezdődő ugyanilyen hosszú memóriaterülettel.

Az eltérő tartalmú címeket felsorolja tartalmukkal együtt.

Az első (négyjegyű) szám a címet jelenti, a második az azon található adatot. A harmadik az a szám, ami a Z-től kezdődő területben a megfelelő helyen áll. A különbségek felsorolását a gép zónánként és folyamatosan végzi el.

Az összehasonlítás a jelzett terület végén, vagy az X gomb megnyomásakor fejeződik be.

A parancs elé #-et írva a különbségek felsorolása a printerre megy.

5.1.8. *T X Y vagy #T X Y **

X-től Y-ig terjedő részt Z 80 mnemónikus kódba fordítja és kiírja. X-re és Y-ra pontosan ugyanaz vonatkozik, mint a D parancsnál.

T a lista elé pontosvesszőt ír, utána jön az utasítás címe, majd a mnemónikus kód.

Ez a mnemónikus kód néhány kivételtől eltekintve a Zilog jelölésével azonos. A kivételek a következők:

A két byte-ot mozgató, vagy abszolút címre hivatkozó LD utasítások helyett *MV használatos.* (Az LD a szimpla regiszterek közti átvitelre van korlátozva!)

Zilog standard	Homelab 4
ADC HL, regiszterpár	ADC regiszterpár
ADC regiszter, szám	ADC regiszter, $ szám
ADD A, (HL)	ADD (HL)
SBC (IX + d), (IY + d)	SBC (IX + d), (IY + d)
A többi ilyen utasításra ugyanígy!	
BIT O, regiszter	BITO regiszter
SET 1, regiszter	SET 1 regiszter
RES 2, regiszter	RES 2 regiszter
A többi ilyen utasításra ugyanígy!	
CALL C cím	CALLC cím
JP NZ cím	JPNZ cím
RET NC	RETNC
JR Z cím	JRZ cím
A többi ilyen utasításra ugyanígy!	
IM Ø	IM Ø
IM 1	IM 1
IM 2	IM 2
IN A, (cím)	IN (cím)
IN A, (C)	IN A

A többi ilyen utasításra és az OUT-ra ugyanígy.

OTIR	OUTIR
OTDR	OUTDR
RETN	RETR

*5.1.9. ; X MNEMONIC**

Ez a parancs a kettősponthoz hasonlóan működik. X címre berakja annak az utasításnak a kódját és operandusait, amit utána mnemonikus formában beírtak.

A mnemonikus kódokkal kapcsolatban ugyanaz érvényes, mint T-nél.

A relatív ugrásoknál (JR) be lehet írni relatív címet, vagy azt az abszolút címet, ahová az ugrás szükséges.

Ha címnek FF-nél kisebb szám áll az relatív, ha ennél nagyobb, akkor az abszolút cím lesz.

Formailag a visszafordító úgy működik, hogy a beírt sort a CR megnyomása után letörli és visszaírja a Standard formátumot. Ez kettős célt szolgál:

Mint majd látni fogjuk, a felhasználónak nem kell mindig mindent precízen beírni az assembly formából, így az esetleges beírási mód helyett mindig egy tiszta, áttekinthető formátum látszik. Másrészt azonnal meg lehet győződni arról, hogy a gép valóban azt az utasítást vette be, amit szerettünk volna. Ha a gép nem tudta értelmezni a beírt szöveget nem történik meg a felülírás. A cursor a helyén marad és újra kell írni az utasítást.

Minden sor bevétele után a gép pontosvesszőt tesz, és kiírja a soron következő címet. Ide újabb utasítást lehet beírni, vagy üres sorral alaphelyzetbe visszatérni.

A D—: párhoz hasonlóan módosítani kell a T által kiírt listát. A cursorral visszalépkedve bármi — cím is — átjavítható, CR-re újra betölthető.

Figyelem! Ha a javítás során kezdő-címek megváltoznak, ügyelni kell, hogy az abszolút és relatív ugrások helyesek maradjanak. A gép nem végzi el a címek módosítását!

Mint már említettük a felhasználónak nem kell beírni azt a teljes formátumot, amit a gép kiír.

Első könnyítés, hogy egy komplett utasítás után bármi állhat, az nem zavarja a visszafordítást. Átjavított soroknál tehát szükségtelen letörölni a sor hátralevő részét, azt a gép már figyelmen kívül hagyja.

Ugyanígy nem kell ügyelni az utasítás opareandusai közötti betűközökre, vagy a beírt utasítás helyére a soron belül. Második, hogy a számot jelölő $-jel elmaradhat, ha egyértelmű, hogy az adott helyen csak szám állhat (pl. ugrásoknál).

Ahol regiszter is állhat ott ki kell tenni a $-t, mert a HEX számok jegyei egyben regisztereket is jelölnek. A végzárójel helyett lehet betűközt írni, a kezdő zárójel pedig csak akkor szükséges, ha az egyértelműség azt megkívánja.

6. Duplapontos aritmetika

A Homelab 4 Basic-jének számábrázolási pontossága 6 számjegy.

Ez sok esetben kevésnek bizonyul. Ennél lényegesen többre van szükség statisztikai, illetve adatfeldolgozási feladatoknál és olyan matematikai eljárásoknál, ahol a hibák összegződnek. Ezért kifejlesztettünk egy olyan *aritmetikai bővítést,* amely a számkijelzés pontosságát 14 számjegyre növeli.

Ez a program további 4 k ROM-ot igényel, tehát duplapontos aritmetika csak a 16 k ROM-ot tartalmazó gépekben van!

6.1. A Duplapontos számábrázolás

A duplapontos csomagban minden szám 8 bájtos lebegőpontos alakban van eltárolva. Ez egyben azt is jelenti, hogy a belső alapműveletek relatív pontossága 2E−17, míg az ábrázolható számok tartománya 1E−32-től 1E+32-ig terjed.

Függvények esetén ez a relatív hiba valamivel nagyobb, de mindenképpen 1E−15 alatt marad.

A gép a számokat 14 jegyre kerekítve írja ki Ø.1-től 99999999999999-ig törtalakban, azonkívül pedig normál alakban. A szám, illetve a tizes kitevő + előjele elhagyható (kiíráskor a gép is betűközt ír a helyére), míg a negatív előjelet mindig ki kell tenni.

6.2. A Duplapontos változók

A duplapontos bővítésben természetesen érvényben maradnak a korábbi változótípusok.

Az új, dupla pontosságú változó (ezután rövidítve D-változó) jele a felkiáltójel (!). D-változó neve egy betűből és a felkiáltójelből áll. Kétbetűs név és a magyar ABC betűi itt nem használhatók.

D-tömbök azonosítására szintén egy betű használható, ami után a felkiáltójel és a zárójeles indexkifejezés áll.

Vektor és mátrix azonosítója nem lehet azonos betű, a tömbök indexkifejezésében nem szerepelhet D-változó. Az index maximális értéke 255. Igy összesen 26 D-változó és 26 D-tömb szerepelhet egy programban.

Az egyszerű D-változókat az első hivatkozás által a gép automatikusan definiálja. Ekkor mind a 26 lehetséges változónak rezervál területet, tehát lefoglal 214 byte-ot a memóriából.

A D-tömböket mindig definiálni kell, nincsen automatikus definiálás a 10 alatti tömbökre sem. A definiálást a DIM utasításnál ismertetjük. Itt említjük még meg, hogy egy n * mfelső indexhatárú D-tömb definiálására 6+8 * (n+1) * (m+1) bájtot használ fel a gép.

Példák: A! B! . . . Z! A!(5) B!(X, Y) Q!(5, =)

Ezen kívül, ha duplapontos utasításban szerepel a PI változó, értékét az aritmetika 16 jegyre adja!

6.3. Duplapontos kifejezések

Duplapontos kifejezés az, ami duplapontos utasításokban szerepel (lásd később).
D-kifejezésekben használhatók a zárójelek, az összes aritmetikai művelet és reláció.
A logikai műveletek (AND OR NOT) D-kifejezésekhez nem használhatók!
D-kifejezésekben természetesen szerepelhetnek D és valós változók és konstansok is.
A függvények közül a következők állhatnak D-kifejezésekben:

ABS	ATN	COS	EXP
INT	LOG	SIN	SGN
SQR	TAN	VAL	

Nem használhatók az RND, PEEK, FRE függvények! Ha mégis ezek értékére volna szükség, egy D-kifejezésben, akkor előbb egy valós áltozónak kell értéket adni ezekkel a függvényekkel és utána ezt a változót kell beírni a D-kifejezésbe.

Ha a D-kifejezésben valós típusú érték szerepel, akkor azt a gép D-vé alakítja és azon végzi el a műveleteket és függvényeket. Igy számolás közben is mindig D-típusú részeredmény keletkezik.

A D-kifejezések további tulajdonságai azonosak az alap Basic-ben megismert real kifejezésekkel.

6.4. Duplapontos utasítások

Egy utasítás akkor duplapontos, ha *előtte felkiáltójel áll.* Ez azt jelenti a gép számára, hogy ebben az utasításban minden számolást dupla pontossággal kell elvégezni. Az összes valós típusú szám és változó D-típusúvá konvertálódik, égy így megy végbe a kiértékelés.

Három D-utasítás lehetséges, nevezetesen

= (értékadás) !DIM !PRINT

ÉRTÉKADÁS

Alakja az alap Basic-en túl négyféle lehet:

a) !A! = D-kifejezés értékadás D változónak.
A D-kifejezés kiértékelés közben minden valós típusú változó vagy konstans D típusúvá konvertálódik, és ezzel az értékkel folytatódik a számolás.

b) !A = D-kifejezés értékadás valós áltozónak.
D-kifejezés itt a számolások legvégén valóssá konvertálódik, és ezt az értéket kapja a valós változó.

c) !A! = VAL (A$) D-értékadás stringből.
Mivel duplapontosságú INPUT nincs, ez a módja D-értékek bevitelének.
INPUT „Szög:" A$: !A! = VAL (A$)

d) !A$ = STR$ (D-kifejezés) String értékadás.
A$ a D-kifejezés értékének stringképét kapja.
Az STR az egyetlen string fv. amelyben előfordulhat D-változó, ezért utána már csak a bezárójel állhat. (További stringműveletek nem!)

Mivel az értékadások különböző fajtáival az esetleges valós → D, illetve D → valós konverziók megoldhatók, külön konvertáló utasítások nincsenek.

Valós → D:	D → valós:
!A! = B	!A = B!

Dimenzionálás

Alakja !DIM A!(6, 2), B!(5), ...

A DIM előtt felkiáltójel áll, utána pedig a változók listája vesszővel elválasztva. !DIM után csak D-tömböt lehet definiálni, valós és D nem keverhető!

A D-tömbök indexkifejezésben valós változók használhatók, D változók nem!

Minden D-tömböt definiálni kell, a 10 alatti maximális indexűeket is!

Kiírás

Alakja !PRINT A!, B!, C

A PRINT előtt felkiáltójel áll, utána pedig a nyomtatási lista, amely azonos a PRINT utasításban megismerttel.

Ebben valós és D változók és kifejezések vegyesen szerepelhetnek. Nem keverhető viszont a stringek kiírásával. Ahol ez szükséges, ott új PRINT utasítást kell kiadni.

Összefoglalva

A duplapontos változók:

1 betűsek plusz felkiáltójel

PI a π D-értéke

Duplapontos műveletek:

+ – * / ∧

Duplapontos függvények:

ABS ATN COS EXP INT LOG SIN
SGN SIN TAN VAL

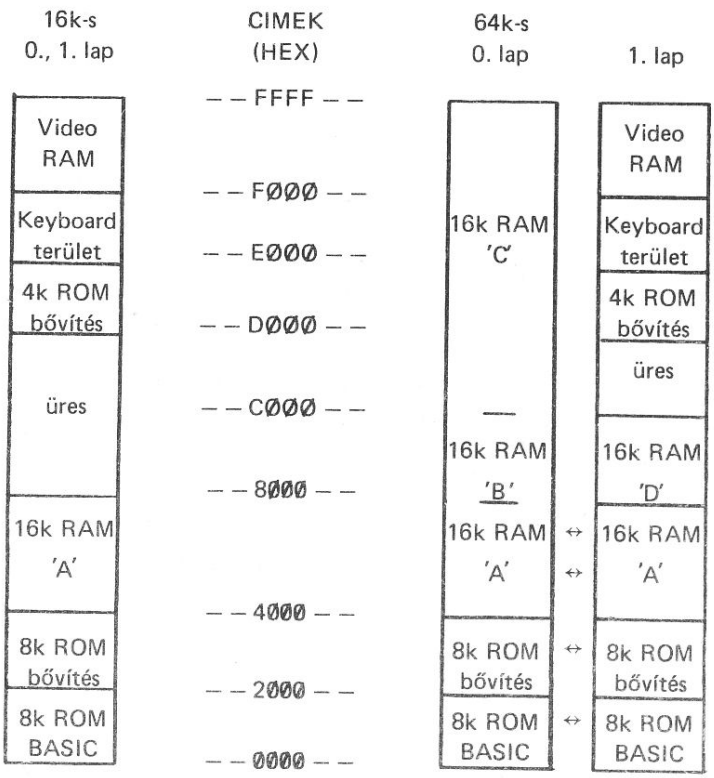
Duplapontos utasítások:

= (értékadás) !PRINT !DIM

ezek előtt felkiáltójel áll, ha utána bárhol az utasításban van legalább egy felkiáltójel!

7. A gép belső rendszere

7.1. A memóriatérkép



A HOMELAB 4 címkiosztását egy TM 188 típusú PROM IC végzi, így elvileg tetszőleges címkiosztás megvalósítható.

Ebben a részben azonban csak a 16k-s, ill. 64k-s gépekben levő „A" és „C" jelű címgenerátorok által meghatározott címkiosztással foglalkozunk.

A Z–80 mikroprocesszor közvetlenül csak 64k memóriát tud kicímezni, ami a 64k-s gépeknél nem elég (64k RAM + rendszer ROM-ok + VIDEO RAM + KEYBOARD terület összesen meghaladja a 64k-t). Ezért gépünk két 64k-s memórialappal (0. és 1. lap) dolgozik. 16k-s gépeknél mindkét lapon ugyan az található.

A 0. lap alsó 16k-ján (HEX ØØØØ-3FFF) a rendszer ROM-ok (BASIC) és azok bővítései (BŐVITETT BASIC, ASSEMBLER, DUPLAPONTOS ARITMERIKA) látszanak. A címgenerátor ide az alapnyákon levő 1., 2., 3. és 4. EPROM tokokat címezi.

A 0. lapon ezután HEX 4ØØØ-től a rendszer RAM következik. Ezt használja a BASIC, ide tölthetők az adatok, itt futhatnak a gépikódu programok stb. (A rendszer -RAM felosztását lásd később). A 16 k-s gépeknél ez a memória HEX 7FFF-ig tart, míg a 64k-s gépeknél 48k-n keresztül a lap tetejéig: HEX FFFF-ig.

A 64k-s gépeknél az 1. lap alsó 32k-ja megegyezik a 0. lap 32k-jával. Erre azért van szükség, hogy – függetlenül attól, hogy a processzor melyik lapon dolgozik – a rendszerprogramok és a rendszerváltozók mindig elérhetők legyenek.

Ezután következik (HEX 8000–BFFF) a 64k-s memória IC maradék 16k-s területe, majd a HEX C000–DFFF területen 8k hely EPROM bővítéseknek.

Az 1. lap felső 8k-s területe (16k-s gépnél mind a két lapon) a perifériáknak van fenntartva: HEX E000–EFFF (4k) a keyboard-terület, HEX F000–FFFF (4k) a VIDEO RAM (ezeket lásd később).

A 16k-s gépeken a HEX 8000–DFFF, a 64k-s gépeken az 1. lap HEX C000–DFFF terület üres így ide további fejlesztések kerülhetnek. A HEX D000–DFFF-re kerülő bővítés az alapnyákon levő 5. EPROM tokba helyezhető, a többi üres címet csak külső bővítéssel lehet használni.

A BASIC és a rendszerprogramok a memórialapozást automatikusan végzik, így alapesetben a felhasználónak ezzel nem kell törődnie. A memórialapozás csak gépikódu programok esetén indokolt a következő helyzetekben:

- Ha el akarjuk érni a 64k-s gép 1. lapján levő 16k RAM-ot
- Ha a VIDEO RAM-hoz, a billentyűzethez kell hozzáférni
- Hangefektusoknál és más rendszerű magnókezelésnél
- EPROM, vagy más rendszerbővítéseknél

A memórialapozás módjai:

Minden gépikódu IN, OUT utasítás átkapcsol a 0. vagy 1. lapra annak megfelelően, hogy a meghatározott I/O cím legnagyobb helyiértékű bitje 0 vagy 1. Igy az

OUT (FF) utasítás az 1. lapra, míg az

OUT (7F) pedig a 0. lapra kapcsol.

Az átkapcsolásra célszerű ezeket a címeket használni, mert más címek esetleg az átkapcsoláson kívül még megszólíthatnak valamilyen I/O eszközt is. Nagyon fontos, hogy I/O-t használó programok esetén az I/O címek megfeleljenek az éppen aktuális lapnak. Gyakori hibaforrás lehet az, hogy I/O kezelés közben lapváltás is történik.

Mint említettük, lapváltáskor csak a címtartomány alsó 32k-ja marad változatlan. Ez újabb két hiba forrása lehet. Az első az, hogy lapváltás után a processzor PC-je (program counter) esetleg nem a helyes programra fog mutatni. Ezért a lapozásban résztvevő programrésznek mindig a címtartomány alsó 32k-jában kell lennie.

A második hiba az lehet, hogy a lapváltás után a SP (stack pointer) nem jó helyre mutat. 64k-s gépeknél a stack általában a memória tetején (HEX FFEØ-tól lefele) szokott működni. Viszont átkapcsolás után ez éppen a VIDEO RAM-ba mutatna. Emiatt a program könnyen elszállhat, vagy „beleszemetelhet" a képernyőbe. Ezért átkapcsolás előtt gondoskodni kell arról, ha van stackhasználat, hogy az új lapon is ott legyen a stack (HEX 8000 alatti címen).

Egyszerűbb esetekben lapozásra használható a következő két szubrutin:

A CALL 0533 utasítás hatására a processzor elmenti a SP tartalmát a HEX 4033 címen kezdődő 2 bájtra és beállítja a SP értékét HEX 402B-re. Ezután átkapcsol az 1. lapra.

HEX 402B-től lefele van 14 szabad bájt HEX 401E-ig, így ez használható stack-nak 7 mélységben.

A CALL 05BE utasítás hatására a SP visszanyeri a régi értékét és a memória visszakapcsolódik a 0. lapra.

7.2. Inicializálás, RESET

Ebben a részben azzal foglalkozunk, hogy mi történik a gép bekpapcsolása után illetve a RESET gomb megnyomásakor.

Bekapcsoláskor egy egyszerű áramkör rövid ideig lehúzza a Z–80-as RESET nevű bemenetét 0 logikai szintre. Erre a processzor alapállapotba kerül és a HEX **0000** címre ugrik. Ez az inicializáló program kezdőcíme.

A RESET gomb megnyomásakor a processzor kap egy NMI-t (nem tiltható interrupt). Ennek hatására elugrik a HEX 0066 címre. Ott a következő található:

```
; 0066   POP   HL          ; az NMI visszatérési címe
; 0067   JP    0000        ; elugrik az inicializálásra.
```

Igy a RESET gomb megnyomásakor pontosan ugyanaz történik, mint a bekapcsoláskor, de mint később látni fogjuk lényeges különbséget okoz a BASIC számára az, hogy a RESET megnyomása előtt a gép már be volt kapcsolva, esetleg program is volt benne.

Lehetőség van arra, hogy speciális felhasználásoknál az NMI-t más funkcióra használjuk. Ekkor a HEX 0066 tartalmát át kell égetni az EPROM-ban egy olyan JP utasításra, amelyik az interrupt lekezelő program címére ugrik. Ezzel azonban elveszítjük annak a lehetőségét, hogy végtelen gépi ciklusba jutott programokat megállítsunk stb.

Most pedig lássuk sorban, mi történik az inicializálás alatt:

– A HEX 4000–4015 területen levő rendszerváltozók megkapják a kezdeti értéküket.

– Teszteli, hogy hány karakter fér egy sorban a képernyőre (32 v. 64) és ennek megfelelően beállítja a HEX 400A bájtot.

– Meghatározza, hogy mennyi RAM van a 0. lapon (közben a RAM tartalma nem változik meg). A legmagasabb RAM címet eltárolja HEX 4016-ra és a stack-et is erre a címre állítja be.

– A képernyőt törli és kiírja a bejelentkező szöveget.

– Ha a HEX 2000-es cím tartalma 00, akkor elugrik HEX 2000-re. Ez a rendszerbővítő EPROM-ok címe.

– Ha az alap BASIC ROM-ok benne vannak a rendszerben akkor beleugrik az alap BASIC inicializálásába (HEX 1D4D).

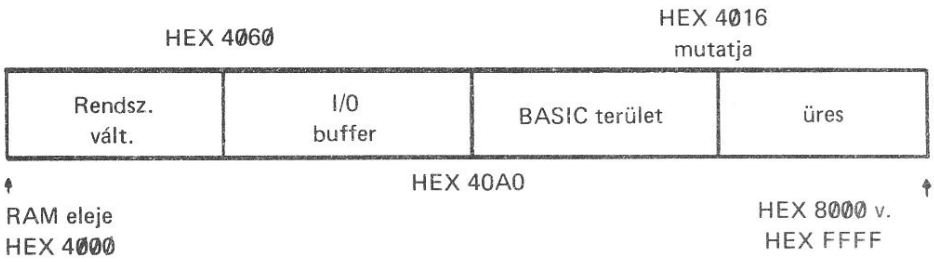
– Ha nincs BASIC ROM, akkor beugrik a gépikódu monitorba (HEX 037C).

Speciális felhasználásoknál szükség lehet arra, hogy a gép ne a BASIC-ban „ébredjen", hanem valamilyen célprogramban. Ez úgy érhető el, hogy pl. HEX 2000-re tesszük azt az EPROM-ot, amelyben a célprogram van úgy, hogy a HEX 2000-en egy 00 bájt van. Az is megtehető, hogy a rendszerben az eredeti rendszerprogramok közül csak az első 2k-t hagyjuk meg (HEX 0000–07FF területen vannak a perifériakezelő programok), a maradék 14k-ra pedig tetszőleges felhasználói programot teszünk. Ezekhez a lehetőségekhez kell tudni a következő ugrási címeket:

```
; 007D   JPZ   2000        ; Ez ugrik a bővítésbe
; 0510   JPZ   1D4D        ; Ez ugrik a BASIC-ba
; 0513   JP    037C        ; Ez ugrik a monitorba
```

7.3.1. A memóriafelosztás

A memóriafelosztás jobb megértéséhez nyújt segítséget a következő ábra:

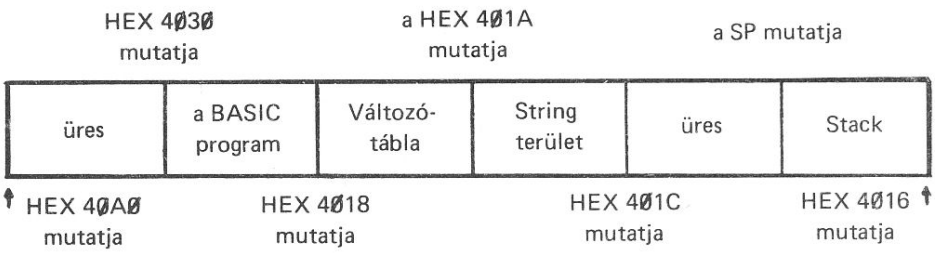


A HEX 4000–405F területen vannak a rendszerváltozók (lásd később). Utána, HEX 4060–409F között van az a buffer, amit a képernyőkezelő és a sorbevevő program használ.

Eztán kezdődik a HEX 40A0 címtől a BASIC által használt terület, aminek a végét a HEX 4016-on levő kétbájtos változó mutatja. Ez a BASIC HM nevű változójával ekvivalens. Ha BASIC-ben megváltoztatjuk a HM értékét, akkor a HEX 4016 tartalma megváltozik és a stack is lejjebb kerül. A HEX 4016-os változó kezdőértéke a memória utolsó bájtjának címe (az inicializálás állítja be).

A memória végéig hátralevő terület üres, ide jöhetnek pl. gépi programok, vagy adatterületek.

A BASIC terület térképe a következő:



A HEX 40A0-tól a HEX 4030 által mutatott címig levő terület üres, ezt a BASIC nem használja. Ide azokat a gépikódu programokat célszerű tenni, melyek a BASIC programhoz tartoznak, ugyanis ez a terület is kitárolódik a BASIC SAVE parancsának végrehajtásakor. A BASIC program kezdőcímét (HEX 4030-at) a NEW parancs után írt számmal lehet meghatározni, így ezzel lehet beállítani az üres terület nagyságát.

Ezután jön az a terület, ahol maga a BASIC program szövege helyezkedik el (tömörített formában). Ennek a területnek a végét a HEX 4018-on levő kétbájtos rendszerváltozó határozza meg. A BASIC SAVE utasítása HEX 4016-tól eddig a címig tárolja ki a memóriát.

7.3.2. A BASIC változótáblája

A változótáblában a változók és aktuális értékeik az első értékadásuk ill. dimenzionálásuk sorrendjében vannak eltárolva. Most felsoroljuk, hogy a különböző típusú változókat hogyan tárolja el a BASIC.

Az N1 ill. N2 a változónév első ill. második betűjének kódját jelenti. Ha a név egybetűs, akkor az N2 értéke HEX 5F. Az I_1 és I_2 a tömbök méreteit jelöli.

– Valós típusú egyszerű:

6 bájt sorrendben: N1, N2, + 4 bájt az értékének: EX, M1, M3, M2 (jelentésüket lásd később).

– String típusú egyszerű:

5 bájt sorrendben: N1 + 32, N2, Hossz, Cím (2 bájt).

A Hossz a string hossza, a Cím pedig annak a memóriának a címe, ahol a string kezdődik. A Címnél az alsó bájt van előbb!

– Valós típusú tömb:

4 * (I_1+1) * (I_2+1) +6 darab bájt sorrendben: N1, N2–64, Link (2 bájt), I_1, I_2, és 4 * (I_1+1) * (I_2+1) bájt a tömb elemeinek.

A Link értéke 4 * (I_1+1) * (I_2+1) +2. Ha ezt a számot hozzáadjuk az I_1 címéhez, akkor a változótábla következő elemének címét kapjuk. A tömb egy elemének értéke pontosan úgy tárolódik, mint az egyszerű típusnál. A tömb elemei az első index szerint folytonosan vannak eltárolva.

– String típusú tömb:

3 * (I_1+1) * (I_2+1) +6 darab bájt sorrendben: N1+32, N2–64, Link (2 bájt), I_1, I_2 és 3 * (I_1+1) * I_2+1) bájt a tömb elemeinek.

Erre a típusra is ugyanaz vonatkozik, mint az előbbire, azzal a különbséggel, hogy itt egy tömbelemhez 3 bájt tartozik. A 3 bájt jelentése ugyanaz mint az egyszerű string típusnál.

A változótáblát egy HEX 60 karakter zárja le. Erre mutat a HEX 401A tartalma is.

A változótábla után köevetkezik a stringterület. Ezen a memóriaterületen tárolódik a stringváltozók egy részének tartalma. Azok a stringek kerülnek ide, amelyek valamilyen művelet során keletkeztek. Ezeknek a stringeknek a változótáblabeli címük erre a területre mutat. Azoknak a stringeknek a címei viszont, melyek a programban stringkonstanst kaptak értékül a programszövegbe mutatnak és a tartalmuk is ott tárolódik. A stringerület végét a HEX 401C-n levő kétbájtos rendszerváltozó tartalmazza.

Ezután a BASIC szabad memóriaterülete következik egészen a SP által meghatározott címig. A BASIC FRE függvénye ennek a területnek az aktuális hosszát adja meg. A SP-től a BASIC terület végéig a BASIC stack-je van. Ide kerülnek a szubrutinok visszatérési címei, a FOR ciklusok adatai stb.

73.3. A BASIC inicializálása

Az alap BASIC kezdőcíme HEX 1D4D. Az itt levő program ellenőrzi, hogy az előbb leírt memóriaterületek „rendben" vannak-e. Ha igen, akkor a program elugrik a BASIC parancsmódjába, és várja a felhasználó parancsait.

Ha viszont a gép most lett bekapcsolva, vagy valami katasztrófális programhiba keletkezett és a rendszerváltozók ellentmondásosak, akkor a BASIC inicializálja magát:

a programterület elejét HEX 40A0-ra teszi, üres programterületet, változótáblát és string-területet állít be, a SP felkerül a BASIC terület tetejére. A NEW parancs hatására is ez történik, de ott be lehet állítani a programterület elejét is.

7.3.4. Aritmetika

Mint már említettük, a HOMELAB 4 a valós számokat 4 bájtban, ún. lebegőpontos formában ábrázolja. Az ábrázolásban szereplő 4 bájt értelmezése a következő:

EX az exponens
M1 a mantissza felső 8 bitje
M2 a mantissza középső 8 bitje
M3 a mantissza alsó 8 bitje

Az ebben a formában megadott szám előjelét az M1 legfelső (7-es) bitje határozza meg. Ha ez 0, akkor pozitív a szám, különben negatív. A szám abszolútértékét a

$$M * 2^{EX-152}$$

kifejezéssel lehet kiszámolni, ahol M az (M1 OR HEX 80), az M2 és az M3 bájtokból képzett 24 bites egészszám értéke. Az M1 7-es bitjét az előjel miatt kell 1-be állítani. A 0 számhoz az EX = M1 = M2 = M3 = 0 ábrázolás tartozik.

Két példa:
A PI ábrázolása HEX 82 49 0F DB, 0.1 pedig HEX 7D 4C CC CD.

Az aritmetikai szubrutinok az ún. akkumulátorokban tárolt számok között végzik el a műveleteket. Két ilyen akkumulátor van, az S és a P, melyek a következő regiztereket jelentik:

	S	P
EX	D reg.	HEX 401F
M1	E reg.	HEX 401E
M2	H reg.	HEX 4021
M3	L reg.	HEX 4020

Az alább felsorolt műveleti szubrutinok az eredményt a P akkumulátorba rakják és a műveleteket sorrendben az S és a P között végzik (P ← S műv. P).

071B +	0713 –	07CB *	06A9 /
0C49 hatvány	0A23 AND	0A18 OR	0C63 >
0C6D <	0C72 <>	0C68 =	0C7C =>
0C77 <=			

A következő függvények szubrutinjai is a P-be rakják az eredményt, az argumentumot pedig mind a két akkumulátorban feltételezik!

0974 INT	0994 ABS	0998 SGN	09AA RND
0B68 COS	0B2C SIN	0B78 TAN	0AA8 ATN
0C3C SQR	0BA4 EXP	0BEB LOG	

További praktikus szubrutinok:

063A	P ← HL értéke
09E6	Az S értékét 16 bites számmá alakítja és a HL-be teszi.
0A09	Az S ill. P értékét 16 bites számmá alakítja és a HL ill. a DE-be teszi.

ØE86	P ← –P
ØB8C	P ← NOT P
Ø671	P ← S
Ø69F	P, S ← Ø
Ø726	S ← P
Ø7BD	P ← P * 1Ø
Ø7C5	P ← P/1Ø

Ha számolás közben valamilyen hiba keletkezik (osztás Ø-val, túlcsordulás stb.), akkor a BASIC hibaüzenetét kapjuk, és parancsmódba jtunk.

7.3.5. Gépikódu programok illesztése

Miután megismertük a BASIC hasznos szubrutinjait, már csak azt kell tudnunk, hogyan lehet BASIC-ből gépi programokat hívni. Erre az alábbi négy lehetőség van:

– A USR (X) függvény kiszámolásakor a BASIC meghívja a HEX 4ØØØ-en levő változó által meghatározott szubrutint. Meghívásakor a HL regiszterpárban van az X argumentum 16 bites egész értéke. A BASIC-be visszatérni egy RET utasítással lehet. A RET után a USR függvény értéke a HL-ben levő szám lesz. A HEX 4ØØØ kezdeti értéke hibaüzenetre mutat!

– A BASIC CALL X utasításának hatására az interpreter meghívja az X címen levő szubrutint. Ebben az esetben direkt paraméterátvitel nem lehetséges.

– A PRINT # X utasítás hatására, ha az X nagyobb mint 1 a program elugrik a HEX 4Ø1Ø által mutatott szubrutinra. Az X értéke az A-ban lesz. A DE regiszter a BASIC program X utáni bájtjára mutat. Visszatérni RET-el lehet. A visszatérés után az interpreter a DE által mutatott helytől folytatja a BASIC porgramot, ezért ügyelni kell arra, hogy a DE visszatéréskor a PRINT # utasítás végére mutasson (ott egy : vagy HEX 60 karakter van)!

– Az INPUT # X utasítás hatására, ha az X nagyobb mint Ø a program elugrik a HEX 400E által mutatott szubrutinra. Itt is érvényesek a PRINT-nél leírtak.

A PRINT # és az INPUT # utasítások elsősorban a periféria-bővítések megoldására valók, de nagyon jól használhatók más gépikódu programok kultúrált beépítésére is.
A PRINT # INPUT # szubrutinjaiból a következő programok hívhatók:

– A CALL 1ØCD a P-be rakja a soron köevetkező kifejezés értékét. A B értéke HEX FF, ha a kifejezés eredménye string volt, különben Ø.

– A CALL 1FC9 a HL-be rakja a soron következő aritmetikai kifejezés 16 bites egészértékét. Az alsó 8 bit az A-ban is benne van.

– A JP 157Ø hatására egy Szintaktikus hiba c. hibaüzenet keletkezik és a BASIC parancsmódba kerül. Az interpreterben a DE regiszterpár olyan szerepet tölt be, mint a gépi programoknál a Z–80 PC-je: a DE mutatja, hogy mi a BASIC program következő bájtja. Ezért lényeges, hogy a PRINT # és INPUT # szubrutinjaiban csak akkor változzon az értéke, ha a BASIC program végrehajtásával továbbhaladtunk. Ha ez elkerülhetetlen, akkor gondoskodni kell a DE eltárolásáról és visszatöltéséről.

Bemutatunk egy mintaprogramot, amit pl. a PRINT # 2, X, Y utasítással hívva a POKE X, Y utasítással lesz egyenértékű.

```
CALL    1FC9        ; HL<—  X értéke
RST4                ; lásd 7.9-ben
CP      2C
```

```
JPNZ    1570        ; Ha nincs vessző, akkor hibát üzen
PUSH    HL
CALL    1FC9
POP     HL          A<— Y értéke
LD      (HL), A
RET                 ; Visszatérés
```

A BASIC CALL utasításánál a szubrutin meghívása után a stack tetején a visszatérési cím van, alatta pedig az a cím, ahol a BASIC program folytatódik. Ezt a címet a DE-be töltve a CALL-nál is használhatók a fenti szubrutinok, de a visszatérés előtt vissza kell rakni a stack-be a DE új értékét és a visszatérési címet!

Enélkül a BASIC elszállna!

7.4. A képernyő

A gép képernyőjének két változata van: 32 vagy 64 karakter/sor. Hogy egységesen tudjuk tárgyalni a két módot, tekintsük a 32 karakteres változatot olyannak, mintha az is 64-es lenne, de a páros és páratlan helyeken ugyanazok a karakterek vannak. Ha az egyiket módosítjuk, akkor a másik is változik.

A képernyőn összesen 32 * 64 = 2048, azaz 2k-nyi karakter hely van, és minden karakterhelyen a függelékben leírt 256 karakter közül bármelyik szerepelhet. A képernyő aktuális tartalma a VIDEO RAM tartalmától függ a következőképpen:

A képernyő legfelső sorának első karakterhelye megfelel a VIDEO RAM HEX F800 címének, a második karakterhely a HEX F801-nek . . . stb. A második sor első helye a HEX F840-nek (64-gyel több), a második helye a HEX F841-nek . . . stb. A képernyő jobb alsó karakterhelye a HEX FFFF-nek. Tehát a képernyő karakterei a bal felső karakterektől kezdve sorfolytonosan megfelelnek a HEX F800-n kezdődő memóriaterület egymást követő címeinek. (A HEX F000—F7FF memóriaterület pontosan ugyanazt tartalmazza, mint a HEX F800—FFFF).

Pl.: Ha a képernyő legfelső sorának elején az áll, hogy „SZIA!", akkor az 1. lap memória tartalma a következő:

```
: F800   53   5A   49   41   21  .....
```

Ennek alapján már tetszőleges módon hozzányúlhatunk a képernyőhöz, de nem árt a következőket szem előtt tartani.

A VIDEO RAM meg van osztva a processzor és a TV-kép frissítőáramköre között. Ha egyszerre próbálják meg címezni a VIDEO RAM-ot, akkor a processzoré az elsőbbség, ami egy zavaró felvillanást okoz a képernyőn. Ez úgy kerülhető ki, hogy a processzor csak képszinkron alatt (amikor a frissítő nem szól a VIDEO RAM-hoz) címezi ezt a területet. Hogy mikor van képszinkron az a következő rövid programmal tesztelhető:

A programnak az 1. lapon levő HEX E802-t kell megszólítania!!

```
MV      A,($E802)
RRA
```

Ha a processzor Carry flag-ja 1, akkor képszinkron van, ha 0, akkor nincs.

A képszinkronhoz való szinkronizációt végzi el a HEX 00F6-on levő szubrutin:

```
;00F6       PUSH        AF          ; Elmenti az A-t
;00f7       MV          A,(E802)
```

```
;00FA    RRA
;00FB    JRC     00F7      ; Levárja az előző k.sz.-t
;00FD    MV      A,(E802)
;0100    RRA
;0101    JRNC    00FD      ; Vár a következő k.sz.-ig
;0103    POP     AF        ; Visszatölti az A-t
;0104    RET
```

Ezt a rutint is az 1. lapon kell meghívni !! Visszatérése után közvetlenül 4 ms-ig írhatunk a képernyőre.

– Mint már említettük a 16k-s gépeken felesleges a lapozás, de a komolyabb programokat célszerű úgy megírni, hogy alkalmazzák a lapozást, hiszen így a 64k-s gépeken is helyesen tudnak futni.

A képernyőkezeléshez tartozik még két fontos szubrutin:

– A CALL 0284 utasításon keresztül hívható szubrutin a cursor által kijelölt helyre kiteszi az A-ban levő karaktert (a speciális karaktereket is kezeli). Ezt a szubrutint csak a 0. lapon lehet meghívni.

A már említett képszinkron miatt ez a szubrutin nem közvetlenül a képernyőre dolgozik, hiszen így mindig szinkronoznia kellene, ez pedig nagyon lelassítaná a működését. Ezért először egy bufferba gyűjti a karaktereket és azok csak akkor kerülnek ki a képernyőre, ha a buffer megtelt (a buffer 64 bájt hosszú), vagy ha egy kontroll karakter (HEX 10 alatti karakter) következik. Ha lényeges a program szempontjából, hogy a karakter azonnal kikerüljön, akkor az alábbiakat kell alkalmazni.

```
CALL    0284    ; A bufferba rakja az A-t
SUB     A       ; A<– 0
CALL    0284    ; A buffert kirakja a képe.-re
```

A cursort a következő programmal lehet tetszőleges helyre állítani:

```
SUB     A
CALL    0284
MV      HL,     a kívánt hely
CALL    0277
```

A kívánt hely a VIDEO RAM megfelelő bájtjának címét jelenti.

A 0284-es szubrutin a RST5-tel is meghívható. Erre vonatkozóan lásd a 7.9. részt.

7.5. A billentyűzet

A billentyűzet állapotának lekérdezésére szolgál a keyboard-terület. Jóllehet ez a terület is 4k (HEX E000–EFFF az 1. lapon), de 32 bájtonként ugyanaz ismétlődik. A billentyűzetkezelés megértéséhez tekintsük a következő táblázatot:

Címek (HEX)	billentyűk			
E800:	cur.le	cur.fel	cur.jobb	cur.bal
E801:	SPACE	CR		
E802:	k.sz.	SH1	SH2	ALT
E803:	magnó	F2	F1	
E804:	Ø	1	2	3
E805:	4	5	6	7
E806:	8	9	:	;
E807:	,	=	.	?
E808:	Λ	A	A	B
E809:	C	D	E	E
E80A:	F	G	H	I
E80B:	J	K	L	M
E80C:	N	O	O	Ö
E80D:	P	Q	R	S
E80E:	T	U	Ü	V
E80F:	W	X	Y	Z
bitek:	D0	D1	D2	D3

A k.sz. a képszinkron bitet jelenti, a magnó a magnó input bitjét. A cím + HEX 8Ø ugyanazt olvassa, csak a magnó/hangszóró kimenetet 1-be billenti. Lásd a hangkeltésnél.

Egy billentyű állapotát úgy lehet meghatározni, hogy a billentyűhöz tartozó címet be kell olvasni és a táblázat alján bejelölt bitet le kell tesztelni: ha a bit Ø, akkor a billentyű le van nyomva.

Pl.: egy mintaprogram, ami addig vár, amíg a CR gombot le nem nyomják.

```
VAR:  CALL    ØØF6          ; A k.sz.-t levárja
      MV      A, (E8Ø1)
      BIT 1   A             ; A CR gombot teszteli
      JRNZ    VAR
```

A billentyűzetre is érvényes, hogy csak a képszinkron alatt lehet olvasni, különben helytelen eredményt ad. A következő szubrutinok a billentyűkezelést segítik:

– A CALL Ø35B rutin az A-ba hozza az éppen lenyomott billenytű kódját. Ha nincs lenyomva egy gomb sem, akkor ØØ-al tér vissza. Ez a rutin kezeli a SH és az ALT funkciókat is. Ha az F1, vagy F2 le van nyomva, akkor a HEX 4ØØ6 címen levő vektor által meghtározott címen folytatódik a billenytűdekódoló program. Alapesetben a HEX 4ØØ6-on levő vektor HEX Ø6ØØ-ra mutat. Ez a rutin a grafikus karaktereket adja az F1, F2 lenyomására. Az ugráskor az A-ban van a karakter kódja, és a D regiszter második bitje (BIT1 D-vel tesztelhető) határozza meg, hogy az F1 vagy F2 van lenyomva. Ezt a szubrutint csak a 0. lapon lehet meghívni.

Ez a szubrutin meghívható a RST3-al is. Erre vonatkozóan lásd a 7.9. részt.

– Szükség lehet olyan szubrutinra is, amely bevesz egy komplett sort a billentyűzetről, és a bevett karaktereket ki is írja a képernyőre (mint a BASIC-ban az INPUT utasítás). Ilyen szubrutint szinte minden komolyabb program használ. A CALL Ø546 szubrutin „képernyőeditoros" sorbevételt tesz lehetővé. Akkor tér vissza, ha a felhasználó lenyomta a CR gombot. Ezután a RST2 utasítással hívhatók le az input sor karakterei sorban az A-ba. Ha már az input sor összes karakterét lehívtuk, akkor a RST2 ØØ-t hoz vissza.

Pl.: a következő program bevesz egy input sort és utána kinyomtatja a képernyőre.

```
      CALL    Ø546
      LD      A,OD
CIK:  RST5
      RST2
      AND     A
      JRNZ    CIK
```

Az itt leírt szubrutinokat csak a 0. lapról lehet hívni!!

7.6. Hangkeltés

A magnó és a hangszóró kimenetet ugyanaz az output bit vezérli. Ezt a bitet a keyboard-területen keresztül lehet állítani: a megszólított keyboard-cím 7-es bitje lesz az output bit állapota (pl.: HEX E8ØØ Ø-át, HEX E88Ø 1-t állít be).

Hangefektusokat úgy kelthetünk, hogy az output bitre Ø-ák és 1-ek valamilyen sorozatát küldjük. Megfelelő sorozatokkal kelthetünk füttyöt, zajt, stb., sőt még gyenge minőségű beszédhangot is előállíthatunk. Itt most csak a legegyszerűbb esettel foglalkozunk: bemutatunk két olyan szubrutint, amivel tetszőleges füttyöt, illetve dallamot tudunk előállítani.

– Az első a CALL 18E1-el hívható. Az A határozza meg a hang magasságát, a C pedig a hosszát. A hangmagasság a

$$f = 57692/A \quad (Hz)$$

képlettel számolható, míg a hang hossz kb. C * 10.24 (ms)

– A CALL 1BE7 meghívásával dallamot tudunk lejátszani (ezt a szubrutint használja a BASIC Beep utasítása is). A szubrutin inputja egy memóriacím, ami a DE regiszterpárban van. Ez a cím egy a területre mutat, ahol a dallamot tároltuk a következő módon:

A HEX 21–3F közti számok a ritmus idejét állítják be

$$t = 4 * (x-32) * 10.24 \text{ ms-ra.}$$

A HEX 4Ø–FF közti számok a következő hang magasságát határozzák meg, az előbbi képlet szeint. A zenei hangok táblázata a 8.2. részben találhatók.

A HEX 2Ø a dallamsor végét jelzi. Ezt fontos kitenni, hiszen nélküle a „végtelenségig" fütyülne.

Ezeket a szubrutinokat is csak a 0. lapról lehet hívni!!

7.7. Magnókezelés

A magnókezelésnek az az elve, hogy a gépben tárolt információt „hangá" alakítjuk, amit a magnetofonon rögzíteni tudunk (save). Ezután bármikor, ha szükségünk van rá, a „hangot" visszaalakítjuk információvá és betöltjük a memóriába (load). Először vizsgáljuk meg, hogy mit alakítunk „hangá"!

Minden felvételnek a következő logikai szerkezete van (ezt úgy kell tekinteni, mint egy bájt-sorozatot):

A fejléc:

256 db	HEX		HEX	
		Név		utána
HEX 000	A5		00	

Az adatok:

Kezdő	Adatok		Ell.	Vége
		Adatok		
cím	hossza		bájt	bájt

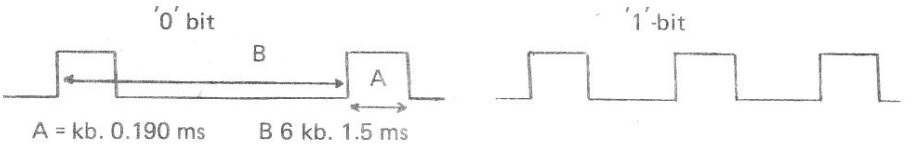
A fejléc elején a 256 db HEX ØØ és a HEX A5 egyrészt a felvétel kezdetét azonosítja, másrészt a betöltő programot szinkronizálja. A NÉV egy tetszőleges karaktersorozat, ami alapján azonosítani lehet a felvételt. Ezt egy HEX ØØ zárja le.

Ezt követi az adat rész. A Kezdő cím azt jelenti, hogy az Adatok a memória melyik részéről származnak és hogy majd hova kell őket visszatölteni. A Hossz az adatok száma bájtokban. A cím és a hossz kétbájtos mennyiségek, először az alsó bájt jön utána a felső. Az Ellenőrző bájt az Adatokban szereplő bájtok összegének alsó 8 bitje. Ezzel ellenőrizhető, hogy a tárolás során nem keletkezett-e valamilyen hiba.

Ezután következik a Vége bájt. Ha ez HEX ØØ akkor a felvételnek vége. Ha nem HEX ØØ, akkor ez azt jelenti, hogy ehhez a felvételhez tartoznak még adatok. Ekkor újból jön a fejléc stb. azzal a különbséggel, hogy a szinkronizációs rész rövidebb és nincs név. Azt, hogy egy felvétel több részből is állhat, akkor lehet kihasználni, ha nem folytonos memóriaterületet kell eltárolni.

Ahogy az előbb bájtsorozatnak tekintettük a felvételt, most tekintsük bitsorozatnak, úgy hogy minden bájt helyett sorrendben a 7., 6., ...0. bitjét vesszük.

Ezek után a hanggáalakítás nagyon egyszerűen történik: a 0 és 1 bitek helyett az ábrán jelölt jelalakokat adjuk ki a magnó output bitre.



A = kb. 0.190 ms B 6 kb. 1.5 ms

A következő szubrutinok használhatók a magnókezelésre:

– A CALL Ø7A3 a BC-ben meghatározott címtől a HL-ben levő címig kitárolja a memóriát. A felvétel nevét a memória egy szabad területére kell letenni és a kezdőcímét a DE-be kell rakni. A nevet egy HEX ØØ vagy HEX 22 vagy HEX 60 karakter zárja le. Ez a rutin egy komplett felvételt készít a magnóra.

– CALL Ø74E ez a rutin beolvassa a magnóról beérkező első felvételt. A beolvasott adatokat a felvételben meghatározott helyre tölti be. Ha hiba történt, akkor a Z flag 0.

– CALL Ø751 a DE-vel meghatározott nevű (ugyanúgy, mint a CALL Ø7A3-nál) felvételt betölti. Addig, vár amíg nem jön ilyen nevű felvétel a magnóról. A név egyezésére ugyanaz vonatkozik mint a BASIC LOAD-nál.

A következő programokat speciális tárolási módoknál lehet használni:

– CALL Ø6AC kitárol egy fejlécet a magnóra. Itt is a DE-vel lehet a nevet meghatározni. Meghívása után a RST1-el lehet az A-ból 1 bájtot a magnóra tárolni.

– CALL Ø61F beveszi a magnóról jövő első fejlécet és összehasonlítja a DE-vel meghatározott névvel. Ha egyezés van akkor az A értéke HEX ØØ. A CALL Ø61F meghívása után a RST1-el lehet az A-ba bevenni a következő magnóról jövő bájtot.

Ezeknél a szubrutinoknál vigyázni kell a programok futási idejével, mert ha két RST1 között túl sok idő telik el, akkor kieshet a szinkronból a beolvasás. Ezeket a programokat csak a 0. lapról lehet hívni.

A magnó input bitjét a képszinkron bithez hasonlóan olvashatjuk:

```
MV      A, ($E883)
RRA
```

Ha a Carry = 1, akkor a bemenet 0, különben 1. Fontos, hogy az input bit olvasása közben az output bit 1 legyen!!

Ezeket a biteket célfeladatokban fel lehet használni bármilyen egyéb I/O-ra (pl. más gépek kazettáit kezelő programokhoz, vagy események számlására), csak arra kell vigyázni, hogy ezek nem standard TTL jelek. (A magnó miatt meg vannak szűrve stb., de ez egyszerűen kiköthető.)

7.8. Printer, PIO

A HOMELAB 4 alapnyákján van hely egy Z–80 PIO-nak (MK 3881) is. Ez kezelheti a printert, vagy ha az nincs, akkor tetszőlegesen felhasználható mint programozható I/O. A továbbiakban a PIO tulajdonságaival nem foglalkozunk, ezek megtalálhatók pl. az LSI ATSZ: Z–80 család c. könyvében.

A PIO a következő 4 I/O címen érhető el (a 0. lapon!):

```
HEX   3C   A port data reg.
HEX   3D   B port data reg.
HEX   3E   A port control /status reg.
HEX   3F   B port control /status reg.
```

Megjegyezzük, hogy a PIO engedélyezését a címbusz A6 bitje végzi, így a PIO több helyen is elérhető, de ez a későbbi bővítésekkel egyszerre szólítanák meg a PIO-t.

A CENTRONIX interface-es printerek kezelésére a következő szubrutinok szolgálnak:

– A printer használata előtt a PIO-t inicializálni kell. Ezt a következő rutinok végzik:

```
CALL   0404    (ezeket a rutinokat lásd a 7.9. részben)
CALL   0423
```

Ez a PIO A port-ját outputmódba, a B port 7-es bitjét inpuba, a többi bitjét pedig outputba állítja. A printer az A port-on keresztül kapja a 8 adatvonalat, a B port 6-os bitjén a STROBE-ot, a 7-es bitjén pedig a BUSY jelet. A maradék 6 bit szabad, de a printerre való íráskor mindig felülíródnak.

– A CALL 0429 kirak az A-ból egy karaktert a printerre. Ez a program a speciális karaktereket (TAB, ékezetes betűk stb.) is lekezeli. A program a Telefongyár TMT 120 típusú printerére készült abban az értelemben, hogy az ékezetes betűket is helyesen kinyomtatja. Természetesen más CENTRONIX interface-es nyomtatók is használhatók, csak azoknál az ékezetes betűk helyett esetleg más jelek fognak megjelenni.

– Néha szükség lehet olyan printer output rutinra, amelyik nem csinál kódkonverziót (pl. printeres grafikánál). Ezt a

```
PUSH    AF
JP      0478
```

szubrutin meghívásával érhetjük el.

7.9. Szubrutinok, konvenciók

Ebben a részben olyan szubrutinokról lesz szó, amelyek általában a gépikódú programok írását könnyítik, gyorsítják meg. Ezenkívül megemlítünk még néhány — a rendszerprogramok által is követett — konvenciót, amit érdemes megtartani.

— Az inicializáláskor a processzor IY regisztere a HEX 4000 értéket kapja. Ez a rendszerváltozók területetének kezdőcíme. Ezért ezt a regisztert nem célszerű használni a gépi programokban, vagy ha mégis elengedhetetlen, akkor a használat után a HEX 4000-et vissza kell tölteni bele. A rendszerprogramok ezt az értéket feltételezik, így ha más érték lenne benne a gép hamarosan „elszállna".

— A BASIC és még sok más program a DE regiszterpárt mint egy általános pointert használja (szövegekre, táblázatokra, programokra stb.). A RST4 utasításra az A-ba bekerül a DE által mutatott bájt és a DE regiszterpár értéke eggyel nő. (Ezt az igen fontos műveletet más processzoroknál pontincrement címzésnek hívják).

— A Z—80 processzor igen hasznos utasításai a RST utasítások. A HOMELAB 4 a következőképpen használja őket:

RST0 Elugrik az inicializálásra.
RST1 A HL-t lerakja a stack-be és elugrik a HEX 400C által mutatott címre.
RST2 Lásd a CALL 0546-nál. (7.4. rész).
RST3 A HL-t lerakja a stack-be és elugrik a HEX 4002 által mutatott címre.
RST4 Az A-ba teszi a DE által mutatott rekesz tartalmát és eggyel növeli a DE-t.
RST5 A HL-t lerakja a stack-be és elugrik a HEX 4004 által mutatott címre.
RST6 A HL-t lerakja a stack-be és elugrik a HEX 4043 által mutatott címre.
RST7 A HL-t lerakja a stack-be és elugrik a HEX 403F által mutatott címre. Ide ugrik az interrupt is az 1-es interrupt módban!

Ezek szerint a RST0, RST2 és RST4 fixen le van kötve, míg a többi vektoros, bárhova lehet őket irányítani.

Inicializáláskor az RST3 vektor a beállítódik a billentyű rutinra (HEX 035C), a RST5 vektora pedig a képernyő output rutinra (HEX 0283). A rendszerprogramok a RST3-at és az RST5-öt hívják, ha egy karaktert be akarnak venni, illetve ki akarnak írni (kivétel a HEX 0546-on levő sorbevevő). Ennek két előnye van. A RST egybájtos utasítás szemben a CALL-al. Másrészt így mód van az adatforgalom átirányítására a vektorokon kerezttül.

Pl.: a CALL 0404 azon kívül, hogy inicializálja a PIO-t, átállítja a RST5 vektorát a printerkezelő programra is. Igy minden olyan output, amit a rendszerprogramok generálnak a képernyő helyett a printerre kerül ki. A CALL 0428 visszaállítja a RST5-öt a képernyőre.

Igy célszerűbb a korábban leírt címek (035B és 0284 helyett inkább az RST3 és RST5-öt használni.

A maradék RST-okat (RST1, RST6 és RST7) a felhasználó szabadon használhatja, azzal a megszorítással, hogy a RST1-et használják a magnókezelő programok is és a RST7 az interruptot is lekezeli (1-es módban).

A továbbiakban leírunk még néhány szubrutint (ezek mind a RST5-öt használják outputnak):

— A CALL 01A0 kiírja a DE regiszterpár tartalmát HEX-ben.

— A CALL 01A5 kiírja az A tartalmát HEX-ben.

— A CALL 1538 kiír egy tetszőleges szöveget. A HL mutatja a szöveg kezdőcímét, az E pedig a szöveg hosszát (max. 255).

— Gyakran előfordul, hogy egy gépi programból üzeneteket kell kiírni. Ezt megkönnyítendő használjuk a következő eljárást:

Irjuk le az összes üzenetet egy szabad memóriaterületre egymás után, úgy hogy minden üzenet utolsó karakteréhez hozzáadunk HEX 80-at (ez fogja elválasztani az üzeneteket egymástól). Az első üzenet elé tegyünk egy HEX FF-et. Ennek a címe legyen XXX. Ezután, ha ki akarjuk nyomtatni az N-edik üzenetet, akkor töltsük be a B-be az N-et, majd hívjuk meg a következő szubrutint:

```
LD      C,Ø
MV      HL, XXX
JP      Ø188
```

A CALL Ø546 (sorbevétel) után bármikor hívható rutinok:

— CALL Ø3E9 megkeresi a következő nem szóköz karaktert, amit a következő RST2 hoz majd be az A-ba.

— CALL Ø3FØ az A-ba bevesz egy ASCII karakterekkel leírt HEX számot. Ha a szövegben nem HEX szám jött, akkor a Z flag Ø (NZ feltétel). Ez a rutin legfeljebb az első két számjegyet veszi be.

— CALL Ø1DC a HL-be bevesz egy HEX számot. Ha nem HEX szám következik a szövegben, akkor HL = Ø-val tér vissza. A B regiszterben meg lehet adni, hogy legfeljebb hány input karaktert vegyen figyelembe. Pl.: ha B = 8-cal hívjuk meg, és az input szövegben több mint 8 HEX számjegy van, akkor az első nyolcat beveszi és ezek közül az utolsó 4 határozza meg a HL értékét.

— CALL ØE6E az input szövegből bevesz egy decimális számot a HL-be. Ha nincs szám, akkor Z = Ø-val tér vissza. Ha a bevett szám nagyobb, mint 32760, akkor elugrik a BASIC-ba és ott hibaüzenetet ad. Előjelet nem figyel!

Ez utóbbi három program a számok előtt levő szóközöket automatikusan kihagyja.

7.10. A rendszerváltozók

Cím (HEX)	TARTALOM	Kezdőérték (HEX)
4000 — 4001	A USR függvény címe.	157C
4002 — 4003	A RST3 vektora.	Ø35C
4004 — 4005	A RST5 vektora.	Ø283
4006 — 4007	F1/F2 vektor.	Ø6ØØ
4008	ALT flag.	ØØ
4009	Scroll stop flag. Ha ez Ø, akkor a képernyő alján megáll a kiírás. Folytatni az SH/SPACE lenyomásával lehet.	nem ØØ
400A	A kép mérete. Ha ez 1, akkor 64, ha 2 akkor 32 karakter fér ki egy sorba. A fizikális méret megváltoztatásához hardver átkapcsolás is kell.	
400B	A képernyőkezelő bufferének számlálója.	
400C — 400D	A RST1 vektora.	
400E — 400F	Az INPUT # vektora.	157C
4010 — 4011	A PRINT # vektora.	157C
4012 — 4013	A gépikódu monitor használja.	
4014 — 4015	A cursor címe a VIDEO RAM-ban.	

Cím (HEX)	TARTALOM	Kezdőérték (HEX)
4016 – 4017	A memória vége (Hm).	7FFF/FFFF
4018 – 4019	A BASIC program vége.	40A0
401A – 401B	A változótábla vége.	40A1
401C – 401D	A stringterület vége.	40A2
401E – 4021	A P akkumlátor.	
4022 – 402B	Stack az 1. lapon.	
402C – 402D	A stringkezelés használja.	
402E – 402F	Az aktuális BASIC sor sorszáma.	
4030 – 4031	A BASIC program eleje.	40A0
4032	Flag-ek a BASIC számára.	
4033 – 4034	Az SP kerül ide lapváltáskor.	
4035 – 4036	A vétrehajtandó utasítás címe a CONT parancs számára.	
4037 – 4038	A következő adat címe a READ utasítás számára.	40A0
4039 – 403A	A végrehajtandó sor sorszáma a CONT parancs számára.	
403B – 403E	Az utolsó véletlenszám értéke.	
403F – 4040	A RST7 vektora.	
4041 – 4042	A bővítések használják.	
4043 – 4044	A RST6 vektora.	
4045 – 4046	A bővítések használják.	
4047 – 4048	A sorbevevő és a RST2 használja.	
4049 – 404A	A BASIC hibakezelő program címe	1581
404B	Interrupt flag.	
404C	Védelmi flag. Ha értéke nem 0, akkor a BASIC programot nem lehet leállítani, betöltéskor automatikusan elindul.	00
404F	A BASIC Cr változójának értéke.	
4050 – 4052	A bővítések használják.	
4052 – 4053	A BASIC használja.	16B2
4054 – 4055	A BASIC használja.	0167
4056 – 4057	A BASIC használja.	1581
4058 – 4059	A BASIC használja.	1C3C
405A – 405D	A printer fejének poziciója.	
405F	A pronter rutin használja.	

8. Függelék

*8.1. Kulcsszavak elhelyezése a billentyűzeten**

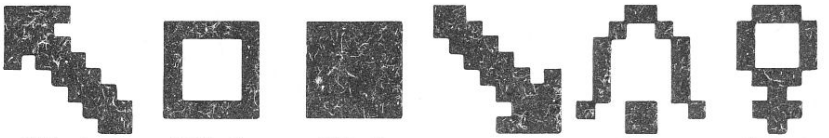
Gomb	+F1	+F2
Q	LIST	SIN
W	LOAD	COS
E	MERGE	SQR
R	REPEAT	INT
T	UNTIL	MIN
Y	RETURN	MOD
U	READ	VAL
I	INPUT	INKEY
O	BEEP	CHR$
P	POKE	PEEK
Ö	REM	MID$
Ó	END	STRING$
A	RUN	TAN
S	SAVE	ATN
D	VERIFY	RND
F	FOR	ABS
G	NEXT	MAX
H	GOSUB	ROUND
J	RESTORE	LEN
K	PLOT	POINT
L	CUR	STR$
Ü	CALL	LFT$
Z	DELETE	EXP
X	EDIT	LOG
C	CONT	USR
V	STEP	SGN
B	POP	FSW
N	GOTO	FRA
M	DATA	ASC
Á	DIM	FORM$
É	EXT	RGH$

*Csak 10 k-s BASIC esetén !

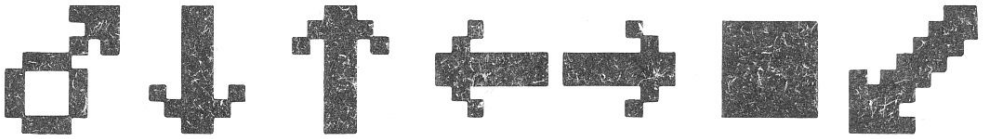
8.2. Hangmagasságok

Zenei hang:	ASCII kód:	Karakter bevitele:
C	253	F2/Shift-bal/Ü
CISZ	240	F2/Shift-bal/P
D	230	F2/Shift-bal/F
DISZ	219	F2/É
E	209	F2/Q
F	197	F2/E
FISZ	188	F1/Shift-bal/Ö
G	180	F1/Shift-bal/T
GISZ	172	F1/shift-bal/L
A	164	F1/Shift-bal/D
AISZ	156	F1/Ö
H	150	F1/V
C	143	F1/O
CISZ	136	F1/H
D	130	F1/B
DISZ	125	ü
E	120	x
F	114	r
FISZ	110	n
G	105	i
GISZ	101	e
A	98	b
AISZ	94	^
H	91	É
C	87	W

Az I-es karakterkészlet



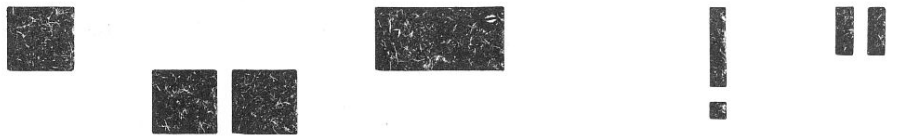
Belső kód: $00–0	$01–1	$02–2	$03–3	$04–4	$05–5	$06–6
PRINT-el: $10–16	$11–17	$12–18	$13–19	$14–20	$15–21	$16–22
Bill/Norm/:					F/SH/	F/SH/
Bill/Alt/:					F/SH/	F/SH/



Belső kód:	$07–7	$08–8	$09–9	$0A–10	$0B–11	$0C–12	$0D–13
PRINT-el:	$17–23	$18–24	$19–25	$1A–26	$1B–27	$1C–28	$1D–29
Bill/Norm/:	F/SH ←	F/ ↓	F/ ↑	F/ ←	F/ →	F/S	F/CR
Bill/Alt/:	F/SH ←	F/ ↓	F/ ↑	F/ ←	F/ →	F/SH/CR	F/CR

Belső kód:	$0E–14	$0F–15	$10–16	$11–17	$12–18	$13–19	$14–20
PRINT-el:	$1E–30	$1F–31					
Bill/Norm/:	F/SH/↑	F/SH/Space					
Bill/Alt/:	F/SH/↑	F/SH/Space					

Belső kód:	$–15–21	$16–22	$17–23	$18–24	$19–25	$1A–26	$1B–27
PRINT-el:							
Bill/Norm/:							
Bill/Alt/:							



Belső kód:	$1C–28	$1D–29	$1E–30	$1F–31	$20–32	$21–33	$22–24
PRINT-el:					$20–32	$21–33	$22–34
Bill/Norm/:					Space	SH/1	SH/2
Bill/Alt/:					Space	SH/1	SH/2

Belső kód:	$23–35	$24–36	$25–37	$26–38	$27–39	$28–40	$29–41
PRINT-el:	$23–35	$24–36	$25–37	$26–38	$27–39	$28–40	$29–41
Bill/Norm/:	SH/3	SH/4	SH/5	SH/6	SH/7	SH/8	SH/9
Bill/Alt/:	SH/3	SH/4	SH/5	SH/6	SH/7	SH/8	SH/9

Belső kód:	$2A–42	$2B–43	$2C–44	$2D–45	$2E–46	$2F–47	$30–48
PRINT-el:	$2A–42	$2B–43	$2C–44	$2D–45	$2E–46	$2F–47	$30–48
Bill/Norm/:	SH/:	SH/;		SH/=		SH/?	0
Bill/Alt/:	SH/:	SH/;		SH/=		SH/?	0

Belső kód:	$31–49	$32–50	$33–51	$34–52	$35–53	$36–54	$37–55
PRINT-el:	$31–49	$32–50	$33–51	$34–52	$35–53	$36–54	$37–55
Bill/Norm/:	1	2	3	4	5	6	7
Bill/Alt/:	1	2	3	4	5	6	7

Belső kód:	$38–56	$39–57	$3A–58	$3B–59	$3C–60	$3D–61	$3E–62
PRINT-el:	$38–56	$39–57	$3A–58	$3B–59	$3C–60	$3D–61	$3E–62
Bill/Norm/:	8	9	:	;	SH/,	=	SH/"
Bill/Alt/:	8	9	:	;	SH/,	=	SH/"

Belső kód:	$3F–63	$40–64	$41–65	$42–66	$43–67	$44–68	$45–69
PRINT-el:	$3F–63	$40–64	$41–65	$42–66	$43–67	$44–68	$45–69
Bill/Norm/:	?	á	a	b	c	d	e
Bill/Alt/:	?	SH/á	SH/a	SH/b	SH/c	SH/d	SH/e

Belső kód:	$46–70	$47–71	$48–72	$49–73	$4A–74	$4B–75	$4C–76
PRINT-el:	$46–70	$47–71	$48–72	$49–73	$4A–74	$4B–75	$4C–76
Bill/Norm/:	f	g	h	i	j	k	l
Bill/Alt/:	SH/f	SH/g	SH/h	SH/i	SH/j	SH/k	SH/l

Belső kód:	$4D–77	$4E–78	$4F–79	$50–80	$51–81	$52–82	$53–83
PRINT-el:	$4D–77	$4E–78	$4F–79	$50–80	$51–81	$52–82	$53–83
Bill/Norm/:	m	n	o	p	q	t	s
Bill/Alt/:	SH/m	SH/n	SH/o	SH/p	SH/q	SH/r	SH/s

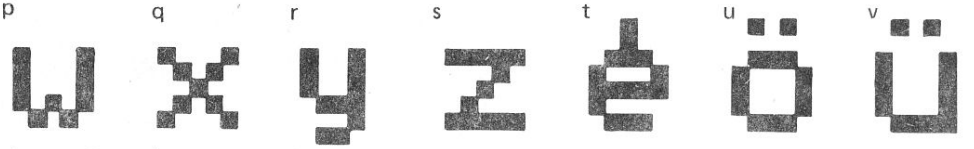
Belső kód:	$54–84	$55–85	$56–86	$57–87	$58–88	$59–89	$5A–90
PRINT-el:	$54–84	$55–85	$56–86	$57–87	$58–88	$59–89	$5A–90
Bill/Norm/:	t	u	v	w	x	y	z
Bill/Alt/:	SH/t	SH/u	SH/v	SH/w	SH/x	SH/y	SH/z

Belső kód:	$5B–91	$5C–92	$5D–93	$5E–94	$5F–95	$60–96	$61–97
PRINT-el:	$5B–91	$5C–92	$5D–93	$5E–94	$5F–95	$60–96	$61–97
Bill/Norm/:	é	ö	ü	^	ó		a
bill/Alt/:	SH/é	SH/ö	SH/ü	^	SH/ó		SH/a

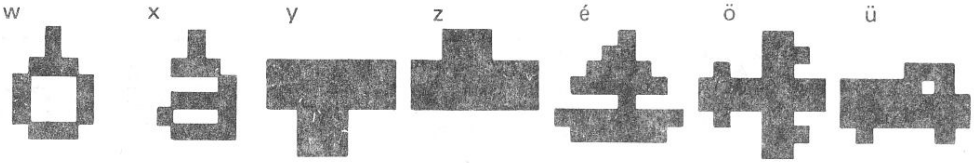
Belső kód:	$62–98	$63–99	$64–100	$65–101	$66–102	$67–103	$68–104
PRINT-el:	$62–98	$63–99	$64–100	$65–101	$66–102	$67–103	$68–104
Bill/Norm/:	SH/b	SH/c	SH/d	SH/e	SH/f	SH/g	SH/h
Bill/Alt/:	b	c	d	e	f	g	h

Belső kód:	$69–105	$6A–106	$6B–107	$6C–108	$6D–109	$6E–110	$6F–111
PRINT-el:	$69–105	$6A–106	$6B–107	$6C–108	$6D–109	$6E–110	$6F–111
Bill/Norm/:	SH/i	SH/j	SH/k	SH/l	SH/m	SH/n	SH/o
Bill/Alt/:	i	j	k	l	m	n	o

Belső kód:	$70–112	$71–113	$72–114	$73–115	$74–116	$75–117	$76–118
PRINT-el:	$70–112	$71–113	$72–114	$73–115	$74–116	$75–117	$76–118
Bill/Norm/:	SH/p	SH/q	SH/r	SH/s	SH/t	SH/u	SH/v
Bill/Alt/:	p	q	r	s	t	u	v



Belső kód:	$77–119	$78–120	$79–121	$7A–122	$7B–123	$7C–124	$7D–125
PRINT-el:	$77–119	$78–120	$79–121	$7A–122	$7B–123	$7C–124	$7D–125
Bill/Norm/:	SH/w	SH/x	SH/y	SH/z	SH/é	SH/ö	SH/ü
Bill/Alt/:	w	x	y	z	é	ö	ü

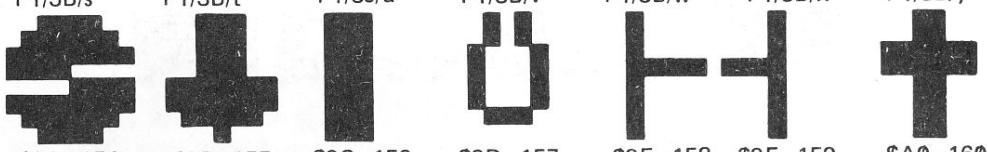


Belső kód:	$7E–126	$7F–127	$80–128	$81–129	$82–130	$83–131	$84–132
PRINT-el:	$7E–126	$7F–127	$80–128	$81–129	$82–130	$83–131	$84–132
Bill/Norm/:	SH/ó	SH/é	F1/á á	F1/a a	F1/ b	F1/ c	F1/ d
Bill/Alt/:	ó	é	F1/SB/á	F1/SJ/a	F1/SB/b	F1/SB/c	F1/SB/d

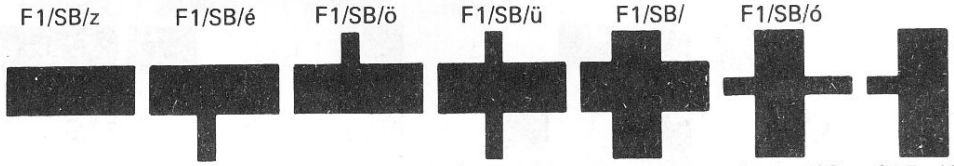
Belső kód:	$85–133	$86–134	$87–135	$88–136	$89–137	$8A–138	$8B–139
PRINT-el:	$85–133	$86–134	$87–135	$88–136	$89–137	$8A–138	$8B–139
Bill/Norm/:	F1/ e	F1/ f	F1/ g	F1/ h	F1/ i	F1/ j	F1/ k
Bill/Alt/:	F1/SB/e	F1/SB/f	F1/SJ/g	F1/SB/h	F1/SB/i	F1/SB/j	F1/SB/k

Belső kód:	$8C–140	$8D–141	$8E–142	$8F–143	$90–144	$91–145	$92–146
PRINT-el:	$8C–140	$8D–141	$8E–142	$8F–143	$90–144	$91–145	$92–146
Bill/Norm/:	F1/ l	F1/ m	F1/ n	F1/ o	F1/ p	F1/ q	F1/ r
Bill/Alt/:	F1/SB/l	F1/SB/m	F1/SB/n	F1/SJ/o	F1/SB/p	F1/SB/q	F1/SB/r

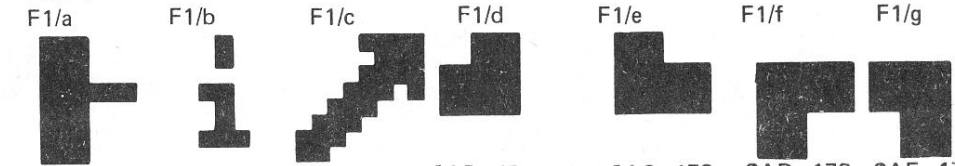
Belső kód:	$93–147	$94–148	$95–149	$96–150	$97–151	$98–152	$99–153
PRINT-el:	$93–147	$94–148	$95–149	$96–150	$97–151	$98–152	$99–153
Bill/Norm/:	F1/ s	F1/ t	F1/ u	F1/ v	F1/ w	F1/ x	F1/ y
Bill/Alt/:	F1/SB/s	F1/SB/t	F1/SJ/u	F1/SB/v	F1/SB/w	F1/SB/x	F1/SB/y



Belső kód:	$9A–154	$9B–155	$9C–156	$9D–157	$9E–158	$9F–159	$A0–160
PRINT-el:	$9A–154	$9B–155	$9C–156	$9D–157	$9E–158	$9F–159	$A0–160
Bill/Norm/:	F1/z	F1/é	F1/ö	F1/ü	F1/	F1/ó	
Bill/Alt/:	F1/SB/z	F1/SB/é	F1/SB/ö	F1/SB/ü	F1/SB/	F1/SB/ó	

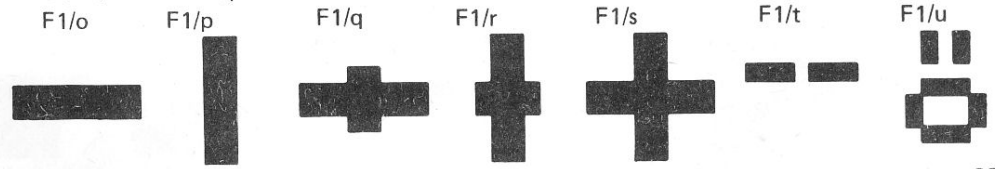


Belső kód:	$A1–161	$A2–162	$A3–163	$A4–164	$A5–165	$A6–166	$A7–167
PRINT-el:	$A1–161	$A2–162	$A3–163	$A4–164	$A5–165	$A6–166	$A7–167
Bill/Norm/:	F1/SJ/a	F1/SB/b	F1/SB/c	F1/SB/d	F1/SB/e	F1/SB/f	F1/SJ/g
Bill/Alt/:	F1/a	F1/b	F1/c	F1/d	F1/e	F1/f	F1/g

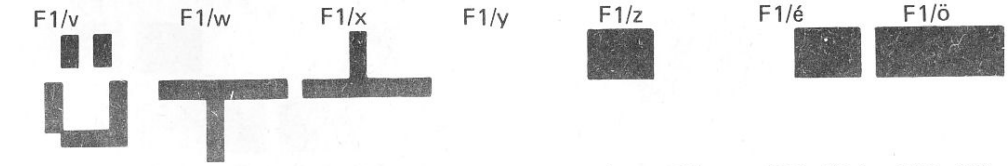


Belső kód:	$A8–168	$A9–169	$AA–170	$AB–171	$AC–172	$AD–173	$AE–174
PRINT-el:	$A8–168	$A9–169	$AA–170	$AB–171	$AC–172	$AD–173	$AE–174
Bill/Norm/:	F1/SB/h	F1/SB/i	F1/SB/j	F1/SB/k	F1/SB/l	F1/SB/m	F1/SB/n
Bill/Alt/:	F1/h	F1/i	F1/j	F1/k	F1/l	F1/m	F1/n

Belső kód:	$AF–175	$B0–176	$B1–177	$B2–178	$B3–179	$B4–180	$B5–181
PRINT-el:	$AF–175	$B0–176	$B1–177	$B2–178	$B3–179	$B4–180	$B5–181
Bill/Norm/:	F1/SJ/o	F1SB/p	F1/SB/q	F1/SB/r	F1/SB/s	F1/SB/t	F1/SJ/u
Bill/Alt/:	F1/o	F1/p	F1/q	F1/r	F1/s	F1/t	F1/u



Belső kód:	$B6–182	$B7–183	$B8–184	$B9–185	$BA–186	$BB–187	$BC–188
PRINT-el:	$B6–182	$B7–183	$B8–184	$B9–185	$BA–186	$BB–187	$BC–188
Bill/Norm/:	F1/SB/v	F1/SB/w	F1/SB/x	F1/SB/y	F1/SB/z	F1/SB/é	F1/SB/ö
Bill/Alt/:	F1/v	F1/w	F1/x	F1/y	F1/z	F1/é	F1/ö



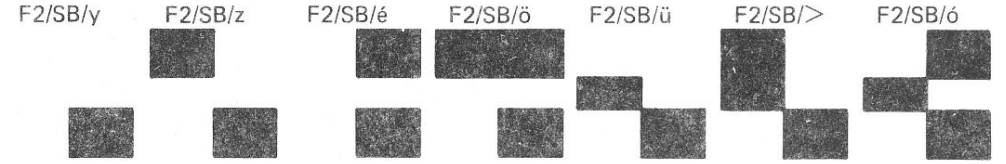
Belső kód:	$BD–189	$BE–190	$BF–191	$C0–192	$C1–193	$C2–194	$C3–195
PRINT-el:	$BD–189	$BE–190	SBF–191	$C0–192	$C1–193	$C2–194	$C3–195
Bill/Norm/:	F1/SB/ü	F1/SB/ó	F1/SB/á	F2/á	F2/a	F2/b	F2/c
Bill/Alt/:	F1/ü	F1/ó	F1/á	F2/SB/á	F2/SJ/a	F2/SB/b	F2/SB/c

Belső kód:	$C4–196	$C5–197	$C6–198	$C7–199	$C8–200	$C9–201	$CA–202
PRINT-el:	$C4–196	$C5–197	$C6–198	$C7–199	$C8–200	$C9–201	$CA–202
Bill/Norm/:	F2/d	F2/e	F2/f	F2/g	F2/h	F2/i	F2/j
Bill/Alt/:	F2/SB/d	F2/SB/e	F2/SB/f	F2/SJ/g	F2/SB/h	F2/SB/i	F2/SB/j

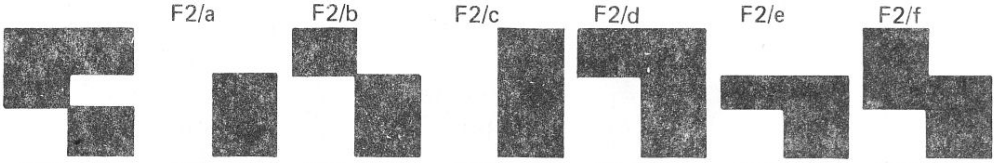
Belső kód:	$CB–203	$CC–204	$CD–205	$CE–206	$CF–207	$D0–208	$D1–209
PRINT-el:	$CB–203	$CC–204	$CD–205	$CE–206	$CF–207	$D0–208	$D1–209
Bill/Norm/:	F2/k	F2/l	F2/m	F2/n	F2/o	F2/p	F2/q
Bill/Alt/:	F2/SB/k	F2/SB/l	F2/SB/m	F2/SB/n	F2/SJ/o	F2/SB/p	F2/SB/q

Belső kód:	$D2–210	$D3–211	$D4–212	$D5–213	$D6–214	$D7–215	$D8–216
PRINT-el:	$D2–210	$D3–211	$D4–212	$D5–213	$D6–214	$D7–215	$D8–216
Bill/Norm/:	F2/r	F2/s	F2/t	F2/u	F2/v	F2/w	F2/x
Bill/Alt/:	F2/SB/r	F2/SB/s	F2/SB/t	F2/SJ/u	F2/SB/v	F2/SB/w	F2/SB/x

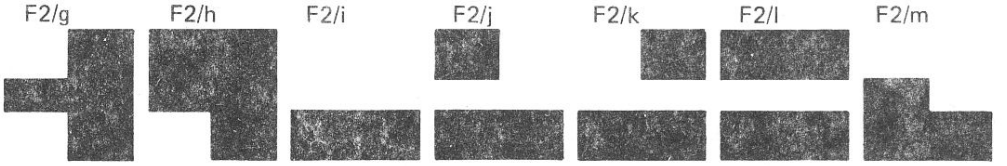
Belső kód:	$D9–217	$DA–218	$DB–219	$DC–220	$DD–221	$DE–222	$DF–223
PRINT-el:	$D9–217	$DA–218	$DB–219	$DC–220	$DD–221	$DE–222	$DF–223
Bill/Norm/:	F2/y	F2/z	F2/é	F2/ö	F2/ü	F2/>	R2/ó
Bill/Alt/:	F2/SB/y	F2/SB/z	F2/SB/é	F2/SB/ö	F2/SB/ü	F2/SB/>	F2/SB/ó



Belső kód:	$E0–224	$E1–225	$E2–226	$E3–227	$E4–228	$E5–229	$E6–230
PRINT-el:	$E0–224	$E1–225	$E2–226	$E3–227	$E4–228	$E5–229	$E6–230
Bill/Norm/:		F2/SJ/a	F2/SB/b	F2/SB/c	F2/SB/d	F2/SB/e	F2/SB/f
Bill/Alt/:		F2/a	F2/b	F2/c	F2/d	F2/e	F2/f



Belső kód:	$E7–231	$E8–232	$E9–233	$EA–234	$EB–235	$EC–236	$ED–237
PRINT-el:	$E7–231	$E8–232	$E9–233	$EA–234	$EB–235	$EC–236	$ED–237
Bill/Norm/:	F2/SJ/g	F2/SB/h	F2/SB/i	F2/SB/j	F2/SB/k	F2/SB/l	F2/SB/m
Bill/Alt/:	F2/g	F2/h	F2/i	F2/j	F2/k	F2/l	F2/m

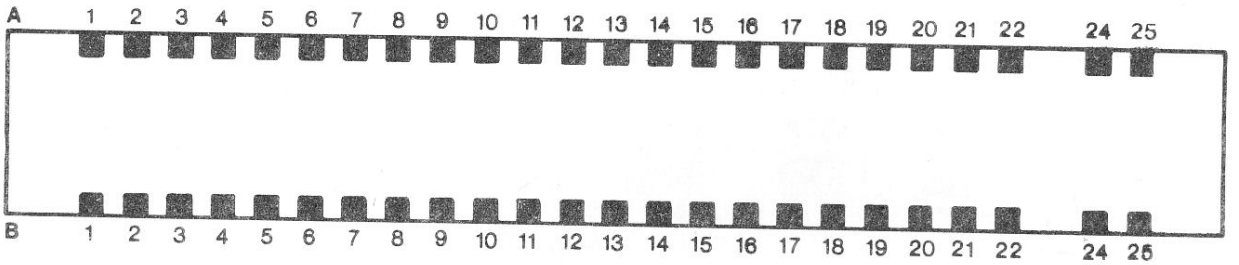


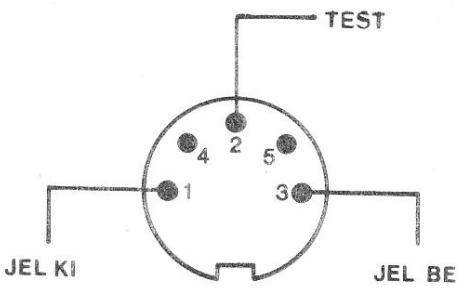
Belső kód:	$EE–238	$EF–239	$F0–240	$F1–241	$F2–242	$F3–243	$F4–244
PRINT-el:	$EE–238	$EF–239	$F0–240	$F1–241	$F2–242	$F3–243	$F4–244
Bill/Norm/:	F2/SB/n	F2/SJ/o	F2/SB/p	F2/SB/q	F2/SB/r	F2/SB/s	F2/SB/t
Bill/Alt/:	F2/n	F2/o	F2/p	F2/q	F2/r	F2/s	F2/t

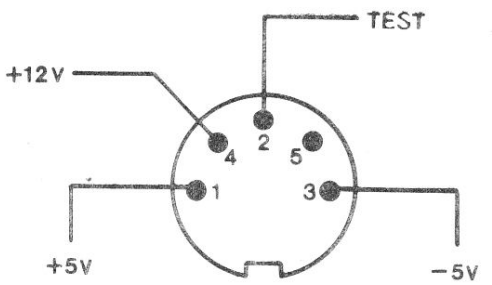
Centronics szabványú Printerek bekötése

PORT 1	PRINTER
PA0	DATA 0
PA1	DATA 1
PA2	DATA 2
PA3	DATA 3
PA4	DATA 4
PA5	DATA 5
PA6	DATA 6
PA7	DATA 7

PORT 1	PRINTER
PB6	STROBE
PB7	BUSY

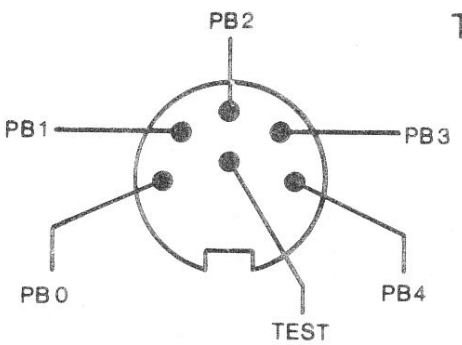






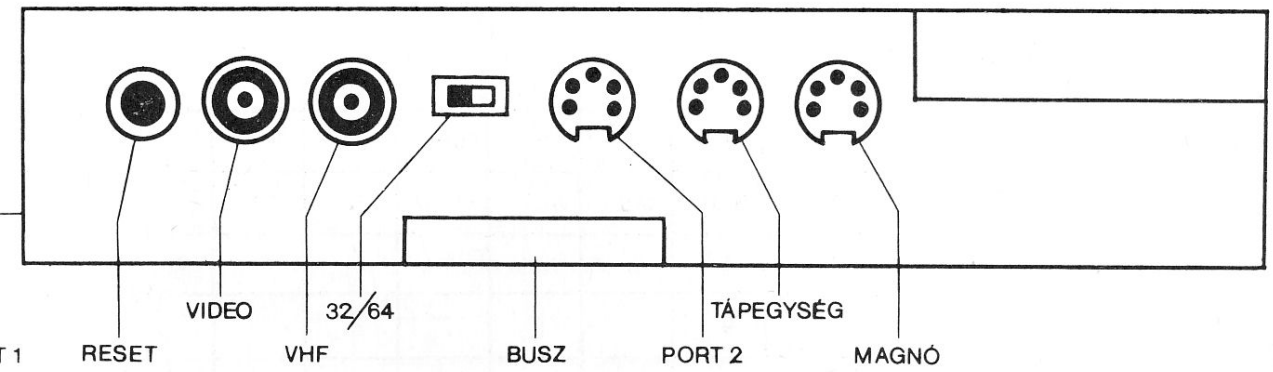
MAGNÓ

TÁPEGYSÉG



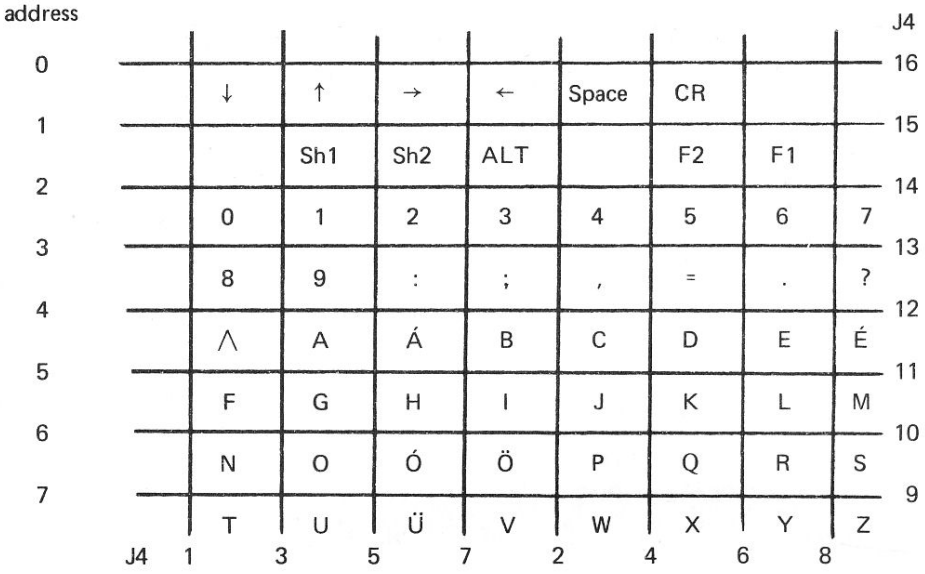
PORT

8.4. A csatlakozók bekötése



BUS CSATLAKOZÓ			PORTL. CSATLAKOZÓ		
Alk. old.		Forr. old.	Alk. old.		Forr. old.
A		B	A		B
50 Hz	1	GND	GND	1	GND
12 MHz	2	+5V	+5V	2	VIDEO
$\overline{MNI}$	3	$\overline{BUSACK}$	PB7	3	KÉP SZ.
$\overline{M1}$	4	A15	PB6	4	GND
A14	5	A13	PB5	5	A6
A12	6	11	PB4	6	A1
A10	7	A9	PB3	7	A0
A8	8	A7	PB2	8	NC
A6	9	A5	PB1	9	NC
A4	10	A3	PB0	10	NC
A2	11	A1	PA7	11	NC
A0	12	$\overline{RESET}$	PA6	12	NC
$\overline{MR}$	13	D0	PA5	13	$\overline{IORQ}$
$\overline{IORQ}$	14	D1	PA4	14	$\overline{WAIT}$
$\overline{WAIT}$	15	D2	PA3	15	GND
$\overline{BRQ}$	16	D3	PA2	16	GND
$\overline{RFSH}$	17	D4	PA1	17	GND
$\overline{RD}$	18	D5	PA0	18	NGD
$\overline{WR}$	19	D6	STBA	19	GND
$\overline{MERQ}$	20	D7	STBB	20	GND
$\overline{INT}$	21	ϕ (3 MHz)	RDYA	21	GND
$\overline{VS}$	22	$\overline{CS5}$	RDYB	22	I
–	23	–	–	23	–
$\overline{CS6}$	24	KS	IEI	24	IEO
GND	25	GND	GND	25	GND

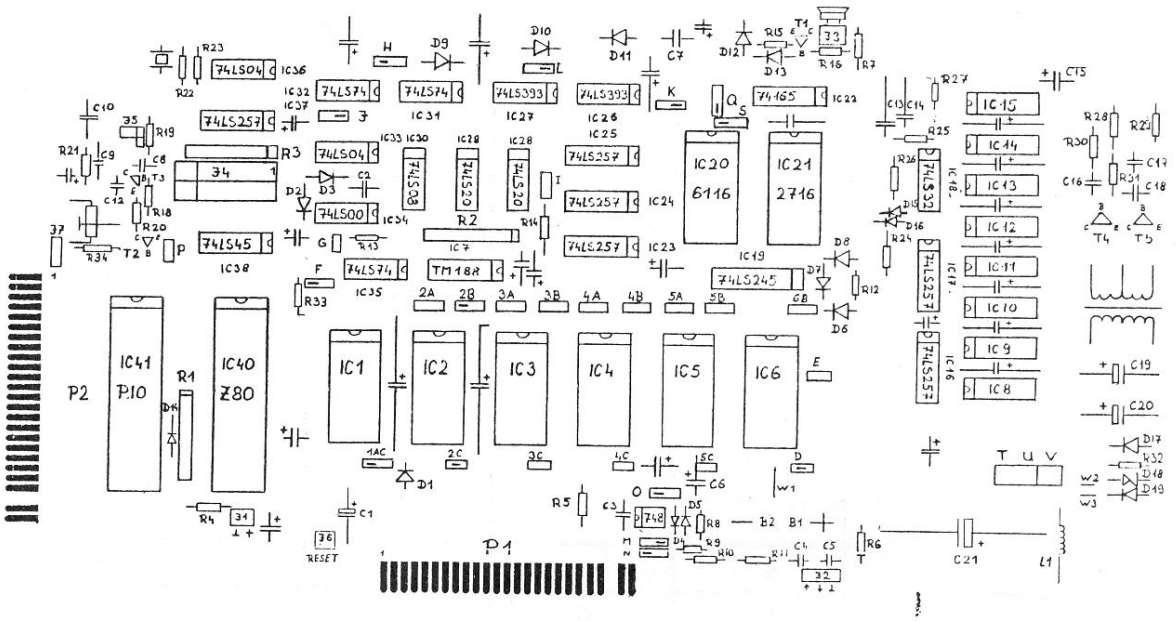
A KEYBOARD MÁTRIXA

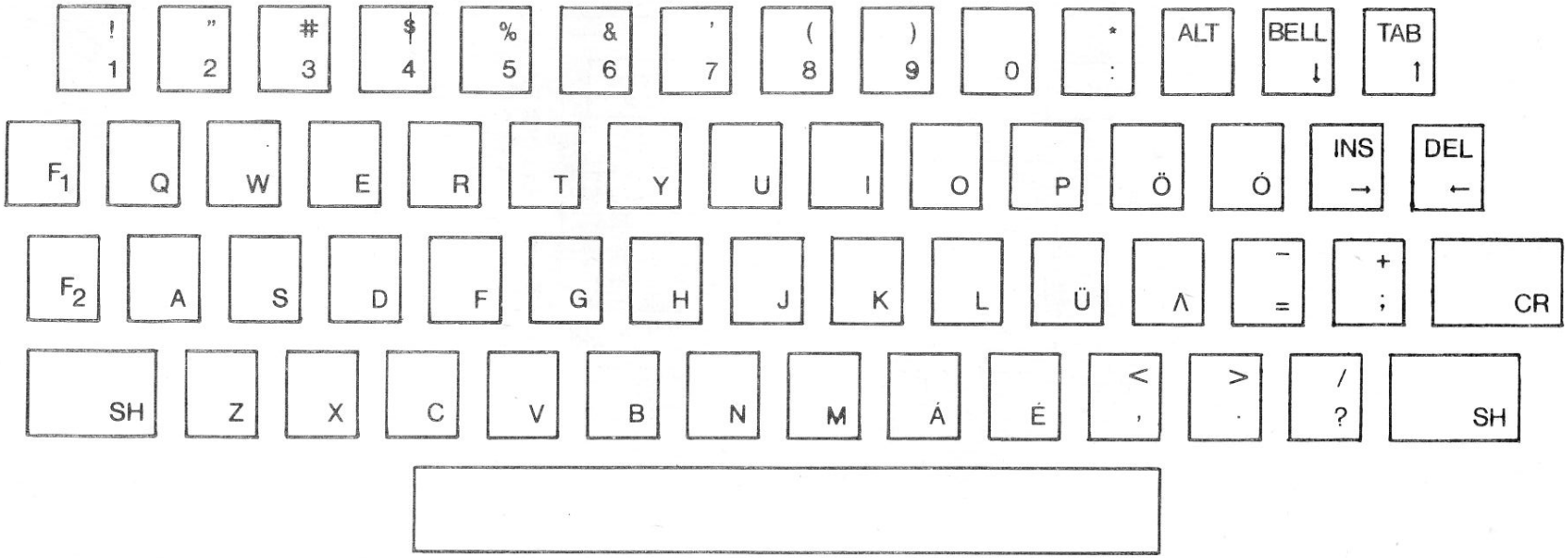


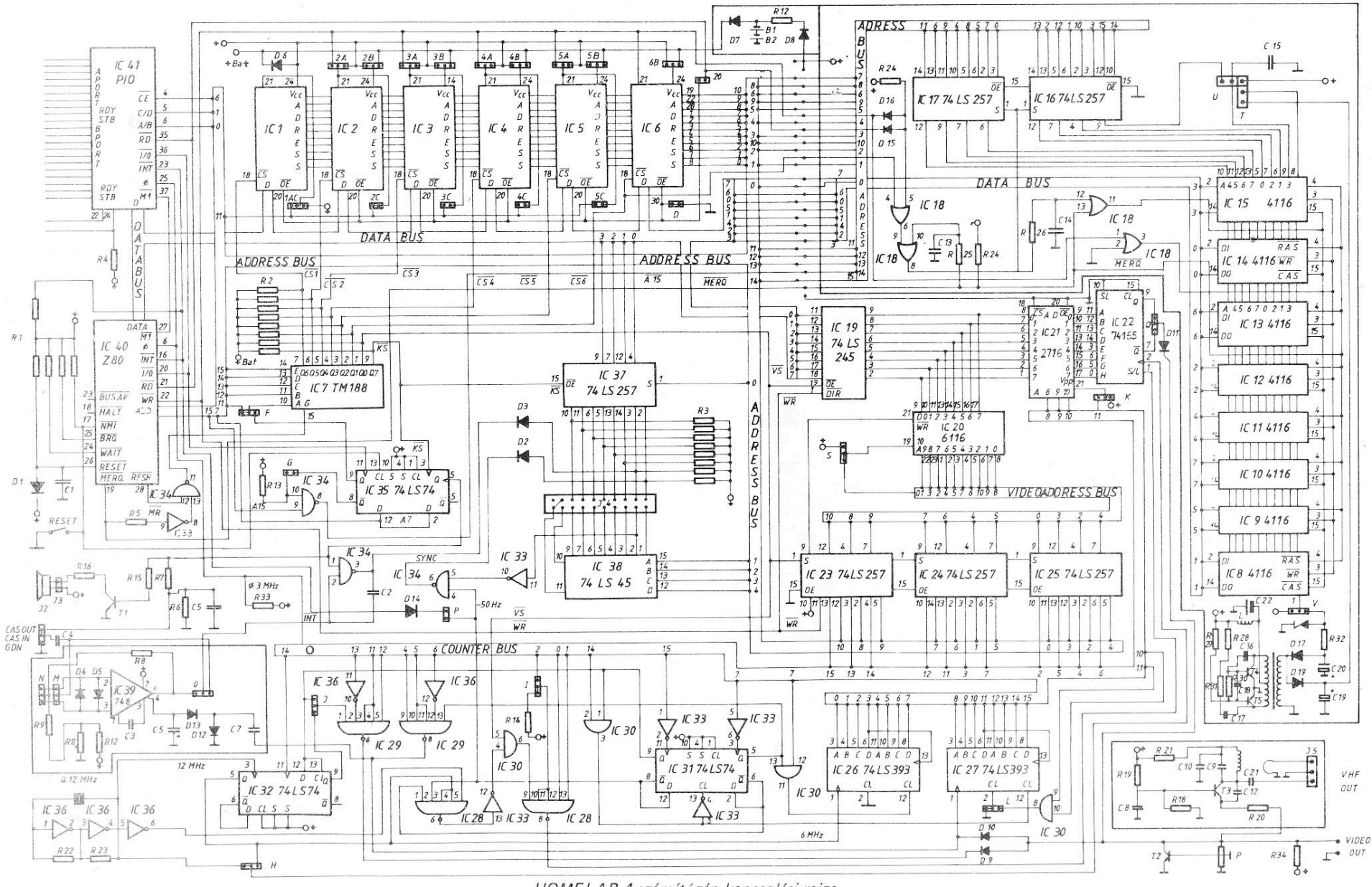
A KEYBOARD MÁTRIXA A GÉP-FELŐL

address				
E000				
E001	↓	↑	→	←
E002	Space	CR		
E003	input sync	Sh1	Sh2	ALT
E004	input cas	F2	F1	
E005	0	1	2	3
E006	4	5	6	7
E007	8	9	:	;
E008	,	=	.	?
E009	∧	A	Á	B
E00A	C	D	E	É
E00B	F	G	H	I
E00C	J	K	L	M
E00D	N	O	Ó	Ö
E00E	P	Q	R	S
E00F	T	U	Ü	V
	W	X	Y	Z
data	D0	D1	D2	D3

8.6. A HOMELAB 4 számítógép beültetési rajza:




! 1
" 2
3
$ 4
% 5
& 6
' 7
(8
) 9
0
* :
ALT
BELL ↓
TAB ↑
F1
Q
W
E
R
T
Y
U
I
O
P
Ö
Ó
INS →
DEL ←
F2
A
S
D
F
G
H
J
K
L
Ü
Ä
_ =
+ ;
CR
SH
Z
X
C
V
B
N
M
Á
É
< ,
> .
/ ?
SH



HOMELAB 4 számítógép kapcsolási rajza

8.8. A HOMELAB 4 számítógép alkatrészjegyzéke

Poz. szám	*Típus*	*Megjegyzés*
IC 1 –IC 6**	2716, 2732, 5516	igény szerint, de legalább 8k EPROM
IC 7 **	TM188, 74S188	megfelelően beégetve
IC 8 –IC 15*	4116, 4164	16 vagy 64k-s gépben
IC 16 –IC 17	74LS157, 257, 258	csak dinamikus RAM-hoz
IC 18	74LS32	csak dinamikus RAM-hoz
IC 19	74LS245	
IC 20 *	5516, 6116	32 betű/sor esetén 4118
IC 21 *	2716	karaktergenerátornak bégetve
IC 22	74LS165, 74165	
IC 23	74LS157, 74LS257	
IC 24 –IC 25	74LS157, 257, 258	
IC 26 –IC 27	74LS393	
IC 28 –IC 29	74LS20	
IC 30	74LS08	
IC 31 –32	74LS74	
IC 33	74LS04	
IC 34	74LS00	
IC 35	74LS74	
IC 36	74LS04	
IC 37	74LS257	
IC 38	74LS42, 7445	további 8 diódával 74LS42
IC 39	748	vagy hasonló
IC 40 *	Z80, MK3880, U880	
IC 41 *	Z80–PIO, U855	
** foglalatba kell tenni	*ajánlott foglalatba tenni	
T1	BC182,549,2N2219	
T2	BC182,184,549	esetleges 2N22 9
T3	BFY90,(BC182)	csak modulátorhoz
T4 –T5	2N2219	csak 4116-hoz
Q	12MHz kvarc	
hangszóró	8 Ohm-tól fölfelé	kisméretű legyen
mikrokapcsoló	bármilyen típus	
C1	1µF–10µF	
C2	1nF–2.2nF	
C3	10PF–15pF	csak IC 39-hez
C4	0.1µF–1µF	
C5	100nF–0.1µF	
C6	100nF–0.47µF	csak IC 39-hez
C8	1nF–22nF	csak modulátorhoz
C9	5pF–15pF	csak modulátorhoz
C10–C11	1nF–22nF	csak modulátorhoz
C12	10pF–15pF	csak modulátorhoz
C13–C14	200pF–390pF	csak dinamikus RAM-hoz

Poz. szám	*Típus*	*Megjegyzés*
C15	100nF–0.47µF	csak 4116-hoz
C16–17	2.2nF–3.3nF	csak 4116-hoz
C18	6.8nF	csak 4116-hoz
C19–C20	100µF–330µF	csak 4116-hoz
C21	1000µF	csak 4116-hoz

A jelöletlen kondenzátorok 47nF–100nF kerámia, vagy 0.1µF–0.47µF csepptantál típusúak.
Ezekből elszórva 15–20 db. szükséges.

D1 –D17, D19	1N914, BAY49, 4819	bármilyen Si típus
D4 –D5		csak IC 39-hez
D7 –D8		csak akkumulátorhoz
D14		csak P Jumper zárásakor
D15–D16		csak Dinamikus RAM-hoz
D17		csak 4116-hoz
D18	5.IV zener	csak 4116-hoz
D19		csak 4116-hoz
R1	1K–33K 5R létra	vagy darabokból összerakva
R2	1K–4.7K 8R létra	vagy darabokból összerakva
R3	1K–4.7K 8R létra	vagy darabokból összerakva
R4	1K–10K	
R5	200–510	
R6	1K–3.3K	R7/R6 = 50–100
R7	56K–150K	
R8	10K–22K	csak IC 39-hez
R9	10K–22K	csak IC 39-hez
R10	1K–4.7K	csak IC 39-hez
R11	4.7K–22K	csak IC 39-hez
R12	1K–3.3K	csak akkumulátorhoz
R13–R14	1K–10K	csak I ill. G nyitásakor
R15	470–2.2K	
R16	33–68	
R17	1K–10K	
R18–R19	1K–10K	csak modulátorhoz
R20	470–680	csak modulátorhoz
R21	10–33	csak modulátorhoz
R22–R23	470–1.5K	
R24	1K–4.7K	csak Dinamikus RAM-hoz
R25–R26	180–330	csak Dinamikus RAM-hoz
R28–R29	1K–4.7K	csak 4116-hoz
R30–R31	33–68	csak 4116-hoz
R32	1.5K–3.3K	csak 4116-hoz
R33	270–330	
R34	1K–4.7K	
P	22K–100K	trimmer

8.9 Mintaprogram

B e m u t a t k o z á s

```
10  PRINT CHR$(12) :PRINT: PRINT
20  INPUT "HOGY HIVNAK?"; A$
30  PRINT : PRINT : PRINT "NAGYON ÖRÜLÖK"
40  PRINT : PRINT, "KEDVES __ "; A$
50  PRINT : PRINT : PRINT "ÉN A HOMELAB 4 SZÁMITÓGÉP VAGYOK"
60  PRINT : PRINT : PRINT
70  FOR I = Ø TO 2Ø : B = RND (19Ø) + 64
80  BEEP CHR$ (35, INT (B), 32) : NEXT
```

F ü g g v é n y á b r á z o l á s

```
  1 Print CHR$ (12) : Cr = 1
  5 Input cur Ø,Ø; ,,KÉREM A FÜGGVÉNYT;" F$
 10 A = –1Ø : B = 1Ø
 15 If A * B < = Ø then E = 127 * A/ (A–B):For I=Ø to 95:Plot E , I : Next
 2Ø C = –5 : D = 5
 25 If C * D < = Ø then F = 95 * D/ (D–C) : For I = Ø to 127 : Plot 1 , F : Next
 3Ø M = 1 : P = Ø : N = 1 : Q = Ø : FF = Ø
 32 For WA = 1 to (len (F$)–2)
 34 WB$ = mind$ (F$,WA,3) : If WB$ = ,,SQR" or WB$ = ,,LOG" ,FF = 8Ø
 36 Next
 5Ø For T = A to B step (B–A) /127
 6Ø X = M * T + P
 65 If X = Ø Goto '12Ø
 67 If FF = 8Ø and sgn(X) = –1 Goto '12Ø
 7Ø Y = val (F$)
 75 Y = –Y
 8Ø Z = N * Y + Q
 85 If B–A = Ø Goto '12Ø
 9Ø E = 127 * (T–A) / (B–A)
 95 If C–D = Ø Goto '12Ø
1ØØ F = 95 * (Z–D) / (C–D)
1Ø2 If E < Ø Goto '12Ø
1Ø3 If E > 127 Goto '12Ø
1Ø5 If F < Ø Goto '12Ø
1Ø6 If F > 95 Goto '12Ø
11Ø Plot E, F
12Ø Next
13Ø Print CHR$ (15) : Goto 5
```

Készült a Somogy Megyei Nyomdaipari Vállalat
kaposvári üzemében — 85-6723
1500 példányban
Felelős vezető: Mike Ferenc igazgató