

EXPANDED BASIC

KISS Gy.

A program 30 új szót definiál az eredeti BASIC szavak mellé, és három kihasználatlan funkciót is felhasználttá tesz, valamint a LIST, EDIT és DELETE utasítások "bogarait" is kijavítja. A három felhasználttá tett funkció: a MERGE utasítás (leírást ld. a parancsoknál), a TIME\$ és az & függvények (leírást ld. a függvényeknél). 1987.
02.
23.

A LIST, EDIT és DELETE utasítások megszüntetett "bogarai":

- 1./ Ha egy sorba macskakörmök közé (a lenyil használatával) 128 feletti kódú karaktereket vittünk be, listázáskor, és editáláskor ezek helyett az ilyen kódú utasítások jelentek meg.
- 2./ Ha \emptyset (nulla) sorszámú sort javítottunk, a sorszám nem jelent meg.
- 3./ Ha editáláskor BRK-et nyomt az editálásból kiléptünk ugyan, de soremelés nem történt a képernyőn.
- 4./ Ha a DELETE utasításban nemlétező sorra hivatkoztunk, vagy DELETE utasítást próbáltunk meg használni, hibaüzenetet kaptunk. Ezentúl a DELETE utasítás szintaxisa megegyezik a LIST utasításéval, egyetlen különbség, hogy ha csak DELETE-et írunk nem törlődik az egész program, hanem hibaüzenetet kapunk.
- 5./ Ha 210 karakternél hosszabb sort javítottunk, a gép nem korlátozta a sorhosszt, és megőrült, jobb esetben elszállt. Most, ha 210 karakternél hosszabb sort javítunk, a gép addig nem enged beszúrást végezni, míg a sorhosszt nem csökkentjük 210 karakter alá.

A program további szolgáltatása, hogy teljes szövegű hibaüzeneteket ad. Ide tartozik még, hogy a tokenizálás módosítása miatt szükség volt egy új hibaüzenet bevezetésére:

poze 16 454, 1 gyors forrás
1

?Line too long error-t kapunk, ha a tokenizálás során a sor hosszabb lett 255 byte-nál (pl.: ha shift + lenyil + csillag karakterrel - LOAD tokennel végig írjuk a sort, majd elküldjük).

A beépített óra kikapcsolása szükségessé válik bármely file-kezelő utasítás esetén. Ez automatikusan megtörténik, ha ezeket az utasításokat a programunkba akkor gépeltük, amikor az Expanded BASIC rendszerrel dolgoztunk. De ha egy olyan programot folytatunk, amibe normál BASIC-kel gépeltünk, file-kezelő utasításokat, és a programunkkal az órát is működtetni kívánjuk, jobb ha az összes olyan sort, melyben file-kezelő utasítás van, le-hívjuk edit-be, és shift + előrenyillal elküldjük minden különösebb javítás nélkül. (Kivételt képeznek az olyan sorok, melyekben csak PRINT# vagy INPUT# utasítás szerepel.) Ha file kezelő utasítást használunk, és az óra be van kapcsolva az utasítás végrehajtása alatt a kijelzésre, a file visszaolvashatatlan lesz.

Az Expanded BASIC-kel írt (átírt) programokat ne javítsuk és futtassuk a normál BASIC-kel. Ezt, ha speciális utasítás szerepel a programban nem is tudjuk megtenni.

Az Expanded BASIC új szavainak leírása:

Az Expanded BASIC szavai három csoportba oszthatók: parancsok, utasítások, és függvények.

A Parancsok csak parancs módban használhatóak, program módban ? Syntax error-t eredményeznek. Az utasítások és a függvények mind parancs, mind program módban használhatóak.

I. Parancsok:

REN a,b (RENUMBER) sorszámozza a programot a-tól kezdve b lépésközzel (az elhagyott paraméter értéke automatikusan 10 lesz). Természetesen a programban szereplő GOTO, GOSUB, THEN, ELSE, RUN, RESTORE, RESUME, ERL=, ERL>, ERL<, ERL<>, ERL=>, és ERL<- utasítások mögötti sorszámokat is átsorszámozza.

FIND szöveg

Keresés; a BASIC program olyan sorait, melyben előfordul a szöveg, kilistázza a képernyőre. Folyamatos listázáshoz egy billentyűt (kivétel: BRK) folyamatosan nyomunk kell. A BRK megnyomására a gép visszakerül parancs üzemmódba. Ha egy parancsot akarunk megkeresni, nem elég a parancs egy részét beírni a FIND mögé, hanem a teljes parancsot be kell gépelni. (pl.: ha NEXT utasításokat akarunk keresni, nem elég azt beírni, hogy FIND NEXT.)

CMCT

(COMPACT) tömöríti a BASIC programot. A programban található felesleges kettőspontokat és szóközőket és REM utasításokat kiszedi a programból, majd 1-től kezdve egyesével átsorszámozza azt. (Természetesen a macskakörmök közötti szóközőket és kettőspontokat nem bántja.) Ezzel a funkcióval programunk hosszát jelentősen csökkenthetjük.

VARs

(VARIABLES) Változók. A parancs kilistázza a program változóit, és ezek értékeit a képernyőre. Folyamatos listázáshoz bármely billentyűt (kivétel: BRK) folyamatosan kell érinteni. A BRK megérintésére a gép újra parancs üzemmódba kerül.

ARRS

(ARRAYS) Tömbök. Lásd VARS, csak a tömböket listázza.

TYPs

(TYPES) Tipusok. A változók alapértelmezésű típusát kilistázza a képernyőre. (Ezeket a típusokat tudjuk megváltoztatni a DEFIT, DEFSNG, DEFDBL, DEFSTR utasításokkal.)

OLD

Régi. A NEW utasítással törölt programot visszaállítja, ha időközben még nem vettünk fel változókat.

APPEND "filenév"

Ráakaszt. A memóriában lévő program után tölti a szalagon lévő filenév nevű file-t. (ha nem adunk meg nevet, az első útjába kerülő file-t tölti be.). A nembetöltött blokkokat, melyekben képernyők, vagy assembler programok találhatóak, inverz blokkszám kijelzéssel jelzi a gép. Az auto startok éppen ezért nem hatnak.

Ha az eredeti programban szerepel egy lØ-es sor, és a kazettán is van egy lØ-es sor, a programunk két lØ-es sort fog tartalmazni, ami a BASIC megzavarodását okozhatja. (de lásd a MERGE parancsot!)

MERGE "filenév"

EGYBEOLVASZT. A memóriában lévő programhoz hozzáfűzi a szalagon lévő programot, majd az eredetileg a memóriában lévő programot 1-től kezdve egyesével átsorszámozza, ezután a keletkezett utolsó sorszám + 1-től kezdve egyesével átsorszámozza a kazettáról betöltött programot, végül az egészből egy programot csinál. A file betöltésre igaz az APPEND parancs file betöltése.

HELP

SEGIT. Az utolsó hibás sort kilistázza, a hibás utasítást inverzen. Ha ez nem történik meg valahogyan töröltük a hiba állapotot.

II. UTASÍTÁSOK:

WRITE "filenév"...

CLS: PRINT\$Ø,2,CHR(z)CHR(y)"_KISS Gy._1987.Ø7.",S,178,191,
B,1,VRUN

IR. Tetszőleges file generálható vele a megadott névvel. A file tartalmazhat képernyő darabokat, assembler v. BASIC programot. A file lehet autostartos BASIC v. assembler file is. Az egyes blokkokat külön-külön kell megadni a név után egymástól vesszővel elválasztva őket. Képernyő blokk megadása: S, az elsőnek kimentendő képernyősor száma, az utolsónak kimentendő képernyősor száma. Assembler blokk megadása: A, az elsőnek kimentendő byte címe, ! ahova az elsőnek kimentett byte majd betöltődik, az utolsónak kimentendő byte címe AUTO indítási cím. A második cím elhagyása esetén a file oda töltődik, ahonnan kimentettük. Az AUTO és az utána lévő cím elhagyása esetén a file nem lesz autostartos, és további blokkok adhatók meg. BASIC blokk megadása: B, az elsőnek kimentendő programsor sorszáma, az utolsónak kimentendő programsor sorszáma RUN:

Ha elhagyjuk a RUN-t a program nem lesz auto-startos, és további blokkok adhatók meg. Ha BASIC és Assembler programot is kimentünk, és Assembler autostartot rendelünk el, a gépi kódú programunknak gondoskodnia kell a BASIC mutatók állításáról! (Legegyszerűbb ha így kezdődik a program: CALL 1AF8H; INC HL; LD (40F9H), HL; CALL 1B61H. Ezek után a stack üres!)

CLOCK ON

Óra bekapcsolása. Ezentúl az óra minden másodpercben 1-gyel nő, és kijelzésre kerül a képernyő jobb felső sarkában (ÓÓ:PP:MM formában.)

CLOCK OFF

Órakijelzés letiltása. Az óra eltűnik a jobb felső sarokból (ha ott volt, de a gépben jár tovább).

CLOCK STOP

Az óra megállítása. Az óra eltűnik a jobb felső sarokból (ha ott volt) és megáll.

CLOCK START

Az óra indítása. Az óra elindul (ha állt, egyébként jár tovább), de nem kerül kijelzésre.

CLOCK SET "ÓÓ:PP:NN" MM

Az óra beállítása. Az óra a string kifejezésben megadott értéket veszi fel. (A string kifejezés formátuma: 8 karakter hosszú, az első két karakter az óra, a harmadik karakter kettőspont, a negyedik és az ötödik a perc, a hatodik szintén kettőspont, a hetedik és a nyolcadik a másodperc. Az óra 00-tól 23-ig terjedhet, a perc és a másodperc 00-tól 59-ig.

DPOKE cím, ..., A memória átírása kétbyte-os számokkal (Double POKE). A megadott címre helyezi az utána álló szám(ka)t. A felsorolt számot tetszőleges egészek - 32768-tól 32767-ig. A számok alacsonyabb byte-ja kerül a címre, majd a cím nő egyel és ide a felső byte kerül majd megint nő egyel a cím és ide a következő szám alacsonyabb byte-ja kerül ... egészen addig, míg a listának nincs vége.

SPOKE cím,string,string

A memória átirása a stringekben lévő karakterek értékeivel. (String POKE). A címtől kezdődően beírja a memóriába a stringekben lévő karakterek kódjait, minden egyes karakter után egy növelve a címet, míg a stringek listájának végére nem ér.

III. FÜGGVÉNYEK:

- XOR (A,B) (eXclusive OR) Kizáró vagy függvénykapcsolat (bitenkénti). Az A és B egész kifejezést egymással kizáró vagy logiaki függvénykapcsolatba hozza, és ezt vissza adja eredményül.
- DPEEK (cím) (Double PEEK) A címen lévő byte értékéhez hozzáadja a cím + 1-en lévő byte értékének 256 szorosát.
- BIN\$ (egész kif.) A függvény eredménye egy 16 karakter hosszú bináris karaktorsorozat, mely csak 0 és 1 karaktereket tartalmaz, és melynek bináris értéke a zárójelek közötti kifejezés értéke. A kifejezés értéke - 32768-tól 32767-ig terjedhet.
- BIN (string kif.) A függvény eredménye egy egész szám - 32768 és 32767 között, melynek értéke a zárójelek közötti stringkifejezés bináris értékével egyezik meg.
- CFP (egész kif.) (Convert to Floating Point = lebegőpontossá alakítás.) A zárójelek közötti egész kifejezést lebegőpontossá konvertálja, és ha az kisebb volt nullánál, akkor hozzáad 65536-ot, így a függvény eredménye egy 0 és 65535 közötti egyszeres pontosságú szám.

CTI (kifejezés) (Convert To Integer = egészé alakítás) Ha a kifejezés string, akkor azt mint hexadecimális karaktersorozatot értékeli ki. Ha normális egyszeres vagy kétszeres pontosságú szám, akkor egészé alakítja, de ha nagyobb volt a kifejező értéke, mint 32767, akkor levon belőle 65536-ot, így a függvény értéke - 32768-tól 32767-ig terjedhet. Ha egész számot adunk meg a kifejezésben, a függvényérték ez a szám lesz.

ADDR\$ (egész kif.)

(ADDRESS = cím.) A függvényérték egy olyan 4 karakter hosszú string, mely hexadecimálissá alakítva tartalmazza az egész kifejezés értékét. (ami - 32768-tól 32767-ig terjedhet.)

BYTE\$ (egybyte-os kif.)

A függvényérték egy olyan 2 karakter hosszú string, mely hexadecimálissá alakítva tartalmazza az egybyte-os kifejezést (melynek értéke 0-tól 255-ig terjedhet).

HEXA\$ (egész kif.) A függvényérték hexadecimálissá alakítva tartalmazza (string formájában) az egész kifejezést (melynek értéke - 32768-tól 32767-ig terjedhet), de úgy, hogy a felesleges nullák nem jelennek meg.

PEEK\$ (cím, hossz, maszk)

A függvény egy olyan stringet ad eredményül, melyben a megadott címtől megadott darabszámú byte van felhozva, egyesével bitenkénti logikai és kapcsolatban hozva a maszk-kal. Ha maszkot nem adunk meg, értéke automatikusan 255 lesz. A hossz értéke 1-től 255-ig terjedhet. Nulla hossz esetén üres stringet kapunk, de a többi stringváltozó értéke megzavarodik.

VALUE (string kif.)

A függvényérték típusa függ a megadott string-től. A függvény ugyanis a megadott stringet mint aritmetikai kifejezést tekinti, és kiszámítja a helyettesítési értékét, attól függetlenül, hogy az stringet, vagy számváltozót eredményez-e. Pl.: ha A\$ értéke az, hogy "5 * 6" a VALUE (A\$) függvény eredménye az, hogy 30, és típusa számváltozó. Ha A\$ értéke az, hogy "CHR\$(42)" az eredmény egy karakter hosszú és az, hogy "*". (természetesen ez csak egy stringben tárolható.) Ha a VALUE függvényre ? Syntax error-t kapunk, és a függvény formátuma tökéletes, lehet hogy a független változóban (a megadott stringben) keletkezett a hiba. Pl.: ? VALUE ("5 * (6+z") utasításra ? Synt^aex error-t kapunk, mert a stringben egy zárójel nincs bezárva.

TOKEN\$(string kif.)

A megadott stringet tokenizálja, és ezt visszaadja eredményül. A visszaadott stringben a műveleti jelek és függvények, valamint az utasítások helyett tokenjük lesz található. A függvényre azért van szükség, mert ha a VALUE-t sorozatosan ugyanazzal a stringgel használjuk, az minden alkalommal tokenizálja, és ez elég hosszú idő. Tehát: ha egy stringet sokszor kell függvényként végigszámolni, akkor gyorsabb, ha az elején a TOKEN\$ függvénnyel tokenizáljuk és utána ezt a stringet számoltatjuk végig.

FUNC (string kif.) A TOKEN\$-^lal tokenizált stringek helyettesítési értékét számítja ki. (mint a VALUE, csak csak ennek már tokenizált stringet kell átadni.)

TIME\$

A függvény értéke egy 8 karakter hosszú string, melyben az óra aktuális értékét kapjuk vissza OO:PP:MM formában.

& hexadecimális számsor

Hexadecimális számmegadást tesz lehetővé. Pl. a PRINT & 56CE utasítás hatására megjelenik az