ZZIP

BASIC INTEGER
COMPILER V. 1.0.

A ZZZIP FORDITÓ KÉZIKÖNYVE

TARTALOMJEGYZÉK

1. BEVEZETÉS

A ZZZIP egy Egész értékű Basic Fordító. A Basicben írt programot átalakítja gépi kódúra a gyorsabb futtatás érdekében. A ZZZIP a nagyobb sebességnövekedést azáltal éri el és az "Egész értékű" elnevezés is arra utal, hogy a Basic programban szereplő konstansok és változók értékeit egész értékeknek tekinti (azaz a tizedespont utáni számjegyeket nem veszi figyelembe). Ennek következtében van néhány megkötés a ZZZIP használatára vonatkozólag, de ha gondosan követjük a kézikönyv utasításait, a ZZZIP-et könnyen fogjuk kezelni és meglepően jó eredményeket érhetünk el.

Az első nyilvánvaló kérdés: "Hányszor lesz gyorsabb a lefordított programok futása?" A válasz attól függ, hogy milyen típusú Basic program került lefordításra. Egy olyan program, amely véletlenszerűen megadott számokat rendez sorba, körülbelül 50-szer lenne gyorsabb, mint a Basic, míg egy stringeket sorbarendező program esetleg csak (!) 12-szer. Másrészt egy olyan program, amely különösebb számolás nélkül ábrázol pontokat és vonalakat, valószínűleg csak kétszer lenne gyorsabb a Basicnél (ami még mindig bizonyos haladást jelent). Ha Basic programunkat úgy módosítjuk, hogy a gépi kódú programoknál alkalmazott módszereket használja, sokkal gyorsabb képernyőkezelést kaphatunk (a szalagon található BENCH. BAS program a gyors képernyőkezelésre ad példát).

Szalagunkon a ZZZIP (mely három részből áll- ZIP, ZIPA, ZIPB) és a BENCH. BAS programok találhatók, ez utóbbi néhány egyszerű rutint tartalmaz, s ezekkel szemlélteti a ZZZIP által biztosított sebességnövekedést. Javasoljuk, hogy először a BENCH.BAS-t fordítsa le, mert ezáltal Jobban megismerheti a ZZZIP használatához szükséges eljárást.

2. A ZZZIP FUTTATÁSA

A ZZZIP egyformán jól működtethető szalagon és lemezen. A 2. 1 fejezet a szalagon történő használatot írja le, de a lemezes használatra is alkalmazható a nyilvánvaló különbségek (és a 2. 2 fejezet pontjainak) figyelembevételével.

2. 1 Szalagos rendszerek

Az indulás a Basic képernyő-editorából történjék, miután törölt belőle minden más programot. A ZZZIP futtatásához nyomja le a START billentyűt, s az három részletben automatikusan betöltődik. Ha ez megtörtént, a ZZZIP egy szöveg kiírásával jelzi, hogy annak a kazettának a bekészítésére vár, amely a fordítani kívánt Basic programot tartalmazza. A ZZZIP feltételezi a magnó távvezérlésének lehetőségét (ha nincs ilyen, a PAUSE billentyűt kell használnia). Gépelje be a Basic program nevét (vagy egyszerűen nyomja meg az ENTER billentyűt, s ezáltal az első szóbajöhető program betöltődik). A ZZZIP a Basic programot a memória egy lefoglalt területére tölti be. Megjegyezzük, hogy az ENTERPRISE 64 gépen ennek a nagysága 16K-ra korlátozódik, s ha a Basic program túl nagy, a ZZZIP a "Memory Full" (a memória megtelt) üzenetet írja ki.

Ezután a ZZZIP egy szöveg Kiírásával Jelzi, hogy annaK a KazettánaK a bekészítésére vár, amelyre programjának lefordított változatát Ki aKarJa menteni, továbbá a magnó RECORD billentyűjének lenyomására. Gépelje be azt a nevet, amelyet a lefordított programnak kíván adni, ezután a ZZZIP működésbe lép. (A programnevekről a 2. 3 fejezet ír bővebben). A ZZZIP négy menetben dolgozza fel a Basic programot, s kiírja a Basic sorok sorszámait is, így teszi lehetővé a program fejlődésének nyomonkövetését.

Ha az első két menet alatt a ZZZIP szembekerül valamilyen problémával (például egy Basic paranccsal amelyet nem tud kezelni), egy üzenetet hagy a Kritikus sorszámnál, majd rátér a Következő sorra egészen addig, míg ennek a menetnek a végére nem ér. Ily módon rendelkezésre áll azoknak a sorszámoknak a listája, amelyekhez az adott menetben problémás sorok tartoztak, s a mellettük feltüntetett hibaüzenetek Jelentését a 4. fejezetben található útmutatások alapján értelmezhetjük.

Rendszerint ebben a részben a ZZZIP probléma nélkül fut, s a harmadik és negyedik menettel folytatódik, amikor is kiír a szalagra egy kis Basic betöltő programot, majd magát a lefordított programot. Ez utóbbi két menet alatt is van egyfajta ellenőrzés, s ha előfordul valamilyen hiba, a ZZZIP kiír egy üzenetet és azonnal leáll.

Végezetül a ZZZIP megkérdezi, hogy akarJa-e más egyéb programok fordítását. Ha igen, a ZZZIP visszatér ahhoz a ponthoz, amelynél a Basic forrásprogram betöltődik.

2.2 Lemezes rendszerek

Mindenekelőtt át Kell másolnia a ZIP, ZIPA, ZIPB és BENCH. BAS programokat szalagról lemezre.

A Basic forrásprogram vagy a lefordított program nevének bevitele során beírhatok olyan többletinformációk, mint pl. a lemezegység azonosítója vagy a directory elérési út. A forrás és a lefordított program lehet különböző lemezen is, ha szükséges, s ha csak egy lemezegység áll rendelkezésre, a szokásos lemezcserélésre szólító üzenetek Jelennek meg.

2.3 ProgramneveK

A Különböző lemezrendszerek Kompatibilitását néhány programnévre vonatkozó szabály biztosítja.

Ha a lefordított programot nem látja el névvel (csak megnyomja az ENTER billentyűt), a ZZZIP a default, azaz alapértelmezésű nevet fogja használni: "Z".

A megkívánt név formátuma a következő: egy legfeljebb 8 karakteres fősz, melyet egy esetleges kiterjesztés követ (a pont írásjel és még legfeljebb 3 karakter), mely a program típusát Jelzi. A ZZZIP a lefordított program nevét mindig 12 karakterre korlátozza. Amennyiben talál kiterjesztést, akkor ezt a BASIC betöltőhöz csatolja, a lefordított programnak a ".Z" kiterjesztést adja. Ha nincs kiterjesztés, és a név fősze nem hosszabb 8 karakternél, a ZZZIP egyszerűen hozzácsatolja a lefordított program nevéhez a ".Z" kiterjesztést.

A programok elnevezésére a következő konvenciót javasoljuk:

PROGRHEV.	(Basic változat)
BAS	(Betöltő program)
PROGRHEV.	(Lefordított változat)

. A LEFORDÍTOTT PROGRAM FUTTATASA

A lefordított programok majdnem ugyanúgy kezelhetők, mintha Basic programok volnának. A legfontosabb különbség az, hogy egyszerre csak egy lefordított program tartható a memóriában és ez a 0. program kell, hogy legyen. A programon belül a RUN parancs más programok automatikus betöltésére és futtatására használható (szabadon keverve a Basic és a lefordított programokat).

Minden lefordított programot egy kis Basic betöltő program előz meg, ez automatikusan betölti a lefordított programot és ellenőrzi a program méretét, beleértve a változók által igényelt helyet, stb. Előfordulhat, hogy egy nagyon nagy program lefordítása után megjelenik az "Insufficient memory" (nem elegendő memória) üzenet. Ilyenkor meg lehet próbálni a gép teljes újraindítását és a program újratöltését, de ha ez nem segít, az eredeti program módosításával kell csökkenteni az igényelt memória mennyiségét.

Ezek után a lefordított programnak ugyanazokat az eredményeket kell szolgáltatnia futtatáskor, mint az eredeti Basic programnak, csak gyorsabban. A STOP billentyűvel megszakítható a program futtatása, a START billentyűvel pedig újraindítható.

Megjegyezzük, hogy a CONTINUE nem működik, valamint a SAVE és a LIST sem használható lefordított programnál.

Ha valamilyen okból a lefordított program futása mégsem lenne tökéletes, nézzen utána a lehetséges okoknak a 4.11 fejezetben. Mindamellett javasoljuk, hogy futtassa le mégegyszer a lefordított programot és próbálja megkeresni azt a helyet ahol elszáll a program, ily módon leszűkítheti a Basic programban történő hibakeresés területét.

4. PROBLÉMÁK KEZELÉSE

A felmerülő problémák Kezelésének egyetlen járható útja az eredeti Basic programba való visszatérés és annak módosítása. Jóllehet, a ZZZIP-et alapos vizsgálatoknak vetették alá, nem lehetetlen, hogy esetleg hibát talál benne (egy még ismeretlen tulajdonságát!), de ezekben az esetekben is a Basic programot kell módosítania úgy, hogy a ZZZIP azt pontosan le tudja fordítani. Ne feledje el annak a lehetőségét sem, hogy hiba következhet be a Basic program betöltése során is.

A ZZZIP a következő hibaüzeneteket adhatja:

4. 1 Memory full (memória megtelt)

A Basic forrásprogram betöltése közben azt tudatja, hogy a Basic program túl hosszú (az Enterprise 64 gépen a határ 16 Kbyte). Próbálkozzon a 4. 2 fejezetben javasolt megoldásokkal. Ha az üzenet a fordítás egy későbbi szakaszában jelenik meg, akkor azt mutatja, hogy a ZZZIP kifogyott a változónevek tárolására fenntartott területből. Probálkozzon a változók számának, vagy a változónevek hosszának csökkentéséve 1.

4.2 Too many labels (túl sok címke)

Ez azt jelenti, hogy a ZZZIP kifogyott a címkék (főleg a sorszámok) tárolására fenntartott helyből. Próbálkozzon a sorok számának csökkentésével, kitörölve a REM sorokat, többszörös utasítások használatával, vagy a program két részre bontásával.

4.3 " ..... " not supported (nem elfogadható)

A ZZZIP jelzi a Kezelhetetlen elemet és annaK Basic sorszámát. Az elem lehet egy Parancs (lásd az 5. fejezetet) vagy a REF függvény, vagy lehet egy olyan változó, amely egy konstans érték helyett határozza meg egy vektor méretét vagy egy karaktersorozat maximális hosszát (a ZZZIP ezeknek rögzített memória területeket tart fenn, s ezért nem engedi meg a "dinamikus dimenzionálást").

4.4 Syntax?

Ez valószínűleg azt Jelöli, hogy egy változó ugyanolyan néven szerepel, mint egy heépített függvény, pl. a SIZE vagy a VAL. Ez a Basichen hizonyos esetekhen megengedett (habár a megfelelő beépitett függvény használatát akadályozza), de a ZZZIP minden esetben a függvény értelmezést feltételezi. Ezért egy zárójelben lévő operandust vár, pl. a SIZE(X)-et. Akkor is problémák merülhetnek fel, ha a Kulcsszavakat, úgymint a COLOUR vagy a SCREEN szavakat változónévként használjuk.

4.5 Garbage (hulladék)

HaJdnem biztos, hogy ez a program pontos betöltésének sikertelenségét Jelzi.

4.6 Non-integer / out of range (nem egész / tartományon kívül)

A ZZZIP Jelezni fogja, ha egy Konstans nem elfogadható, vagy azért, mert nem egész, vagy mert Kívül eslK a szoKásos -32767-től +32767-ig terjedő megengedett intervallumon.

4.7 Loop / block termination (ciklus / blokk lezárás)

A ZZZIP ellenőrzi a ciklusok és blokkok pontos lezárását. Ne felejtsük el, hogy a Basic programot a ZZZIP a sorszámok szerint sorban fordítja le, s ez a sorrend nem biztos, hogy megegyezik a futtatás alatti végrehajtási sorrenddel. Mindenfajta ciklus és blokk egymásbaskatulyázható, de a ZZZIP elvárja a sorrendben összeillő nyitó és záró utasításokat.

4. 8 Reference not found (nem találja a hivatkozást)

Olyan helyre történő hivatkozást jelöl, amelyet a ZZZIP nem címkézett meg. Ez a Basic egy fölösleges sorának tulajdonítható, mely GOTO-t vagy más nem létező sorra hivatkozó parancsot tartalmaz. Másrészt olyan utasítás is oka lehet, ahol pl. a 200. sor létezik, a RESTORE 200 számára viszont nem tartalmaz DATA-t.

4.9 Identifier declared twice (kétszer deklarált azonosító)

Azt jelzi, hogy a Basic sorban szereplő egy vagy több változó vagy vektor határozott deklarálása már megtörtént valamelyik előző NUMERIC, STRING vagy DIM utasításban. Ott a legvalószínűbb az előfordulása, ahol lokális változókat vagy vektorokat DEF blokkokon belül deklaráltunk (lásd az 5.2 fejezetet). Vizsgálja át azokat a változókat vagy vektorokat, amelyeket előzőleg már egyszer deklarált és válasszon nekik egyedi nevet.

4.10 Warning - global variable (figyelmeztetés - globális változó)

Azon problémák megoldása érdekében, amelyek a DEF sorokban vagy blokkokban előforduló lokális változókkal kapcsolatosak (lásd az 5.2 fejezetet), a ZZZIP ellenőrzi az átadódó paraméterként használatos változókat, úgymint az X-et és Y-t a DEF FUNC(X,Y)-ban. Abban az esetben, ha egy változónevet már használtunk valamelyik megelőző sorban, vagy ha már határozottan deklaráltuk egy tetszőleges sorban, a ZZZIP egy figyelmeztető üzenetet ad. Ez arra emlékeztet, hogy a ZZZIP a változót globálisként fogja kezelni, de mivel ez nincs hatással a lefordított program futására, az üzenet csak figyelmeztetés és a fordítás folytatódik.

11 Compiled program does not run correctly (a lefordított program hibásan fut)

A két legvalószínűbb ok a DEF blokkokban előforduló lokális/globális változók eltérő kezelése és az egész aritmetika használatának hatásai. Az 5. 2 fejezet írja le a DEF blokkokban lévő változók kezelését. Két egyszerű szabály van, egyrészt bármely lokális változónévnek egyedinek kell lennie, másrészt kerülni kell az olyan helyzeteket, amelyekben egy definiált függvény önmagát hívó CALL utasítást tartalmaz.

A használt egész aritmetikában a tizedespont után álló számjegyek nincsenek figyelembe véve, és minden értéknek a -32767 és +32767 számtartományba kell esni. Ezek a szabályok egyaránt vonatkoznak a számítások közbeeső eredményeire és a végső eredményre. Ily módon az aritmetikai függvényekben alkalmazott műveleti sorrend nagyon fontos. Például, a LET X=2/4*10 nulla eredményt fog adni, s ahhoz, hogy a pontos 5 értékű eredményt adja, a LET X=2*10/4 alakra kell módosítanunk.

Hasonlóan a LET X=5*lOOOO/2 kifejezést át kell írnunk a LET X=lOOOO/2*5 formára. Ezek a példák nyilvánvalóak, mert konstans értékeket használnak. A következő LET X=A/B*C utasítás már nem ilyen magától értetődő. Ilyen esetekben szükség lehet az A, B és C lehetséges értékeinek átgondolására, azt a szabályt alkalmazva, hogy jobb osztás előtt szorozni, hacsak az értékek nem olyan nagyok, hogy már a közbeeső eredmények is túlléphetik a 32767- et.

Bizonyos körülmények között megengedhető, hogy egy változó érték 32768 és 65535 közé essen, mégpedig akkor, ha összeadás vagy kivonás eredménye. Mindamellett óvatosan kell élnünk ezzel az engedménnyel, mivel arra épül, hogy a legnagyobb helyiértékű bit szokásos előjelbitként való használatát figyelmen kívül hagyja. Leírható például a LET X=25000+25000, s ezekután X a következő utasítástípusok bármelyikében használható:

LET X=X+Y vagy LET X=X-Y POKE X,Y vagy LET Y=PEEK(X) SPOKE S,X,Y vagy LET Y=SPEEK(S,X) CALL FUNCTION (X, Y,Z) LET A=FUNCTION (X, Y, Z) IF X=Y THEN. . . (de nem < vagy >)

Ennek a tulajdonságnak a gyakorlati haszna a memória címek kezelésében nyilvánul meg, de ügyelni kell arra, hogy csak a fent említett utasítástípusokat használjuk.

5. BASIC PARANCSOK

A ZZZIP megtart csaknem minden olyan parancsot, amely egy Basic programon belül általában végrehajtható. A Későbbiekben látni fogja, hogy néhány parancs használatakor óvatosnak kell lennie vagy némelyeket egyáltalán nem használhat, a többi viszont korlátozás nélkül rendelkezésre áll.

5.1 ALLOCATE

Az ALLOCATE parancs felmozgat a memóriában egy Basic programot, s ily módon helyet biztosít a gépi kódú beszúrásoknak. A lefordított programok nem mozgathatók el, ezért esetükben a beszúrásoknak egy 255 byte-nyi terület (az 13OOH-tol fölfelé) van fenntartva. Megjegyezzük, hogy ennek a területnek a méretét az ALLOCATE parancs nem befolyásolja. Valójában 255 byte-nál több is rendelkezésre állhat annak a kockázatával, hogy esetleg átnyúlik a 17FDH címtől lefelé terjeszkedő string-kezelő területre. Igy tulajdonképpen Körülbelül lKbyte-nyi rész tartalmazhat beszúrásokat egy programban, s ide nem értendő bele a hosszú stringek kezelése. A legokosabb, ha lefordítja a programot és megnézi, hogy működik-e!

Megjegyezzük, hogy a lefordított programoknak nincs szüksége arra a rutinra, amelyet rendszerint az EXOS 2. 0 verziójában, az ALLOCATE parancsban előforduló hiba kiküszöbölésére használunk. Ha mégis szerepel a programban, akkor a rutinnak meg kell vizsgálnia, hogy VERNUM=2 igaz-e. Igy a lefordított programban ez letiltódik (mivel ekkor a VERNUM függvény értéke 1).

2 DEF

A ZZZIP a legtöbb DEF parancsot elfogadja, mind a Különálló soroK, mind pedig a blokkok függvényként történő definiálásánál, de néhány korlátozás azért fennáll. Lényeg az, hogy a ZZZIP minden változót globálisként kezel. Ennek az az egyik oka, hogy a ZZZIP egy Basic programot a sorok számozásának sorrendjében dolgoz fel, s ez valószínűleg nem fog megegyezni azzal a sorrenddel, amelyben a sorok a futás során végrehajtódnak. Ezért visszakövetkezte- téssel nem tudja mindig meghatározni egy változó globális vagy lokális voltát. Ha egy DEF blokkon belül a változókat lokálissá akarjuk tenni, ezt egyszerűen úgy érhetjük el, hogy egyedi változóneveket használunk. Mindamellett ne felejtsük el, hogy ennek hatására sem fogja a ZZZIP a DEF blokkok futása során a változókat lokálisnak tekinteni, ezért a DEF blokkok nem használhatók rekurzívan. Kissé bővebben kifejtve, tegyük fel, hogy egy blokk, melynek fejléce a DEF FUNC(X) utasítás, tartalmazza a CALL FUNC(Y) utasítást. A Basic minden alkalommal, amikor a függvény meghívja önmagát, megőrzi az X lokális értéket és a későbbiekben, mikor a függvény "legombolyítja" önmagát, visszaadja ezeket az egymást követő X értékeket. A ZZZIP nem őrzi meg ezeket a lokális értékeket, s így egy ilyen helyzetben nem szolgáltat pontos eredményt. Ezért ne engedje meg, hogy egy függvény a CALL utasítással önmagát hívhassa.

A "csak globális" szabály alól van egy Kivétel, ha egy DEF blokkban lévő változónak a neve megegyezik magának a blokknak a nevével, ez a változó felhasználható egy LET utasításban és ezek után a ZZZIP lokális változónak fogja tekinteni. Ez képessé tesz egy DEF blokkot arra, hogy visszaadjon egy értéket a főprogramnak. Példaként nézzük a következő függvényt:

```
lOOO DEF TEN
1010 LET TEN=10
1020 END DEF
```

A főprogramban lévő PRINT TEN utasítás hatására a l0-es szám Kerül majd nyomtatásra. Az 1010-es sor egy egyszerű LET utasítást tartalmaz, s a ZZZIP az ott lévő TEN változót lokálisnak tekinti. Ellenben, ha a TEN nevet az 1010-es sorban valamilyen más típusú utasításban használnánk, ez a TEN függvény rekurzív CALL hívását okozná és a lefordított programban hibásan működne.

Hasonlóan, ha olyan függvényből akarunk értéket visszaadni, amely segédváltozókat tartalmaz paraméterátadás céljából, szintén használhatunk lokális változót. Például a LET DOUBLE = X*2 megengedett egy olyan blokkban, melynek a fejléce DEF DOUBLE(X), s a főprogramban lévő PRINT DOUBLE(3) hatására a 6 kerül majd nyomtatásra. Egyszerű függvénynevek is átadódhatnak egy másik függvényen belül felhasználandó paraméterként, így a főprogramban lévő PRINT DOUBLE(TEN) a 20 kimyomtatását eredményezi. Ezek a példák hibátlanul működnek egy lefordított programban.

Megjegyezzük, hogy a ZZZIP nem fogadja el a REF parancsot és hibaüzenetet ír ki. Valami más, a Basic programéhoz hasonló eredményeket szolgáltató módszert kell találnia.

5.3 PROGRAM

Ez a parancs nincs hatással a lefordított programra és a ZZZIP a REM-hez hasonlóan kezeli.

5.4 RESTORE

Ha adott egy sorszám (mint pl. a RESTORE 200 -ban), ez egy DATA-t tartalmazó sor kell, hogy legyen. Máskülönben a ZZZIP hibaüzenetet ad.

5.5 RUN

Ezzel a paranccsal egy lefordított program újraindítható (az elejétől vagy egy kijelölt sorszámtól). Egy másik program (Basic vagy lefordított) betöltésére és futtatására is használható, de egyik programból a másikba történő paraméterátadás már nem lehetséges.

5.6 STOP

Programban szereplő parancsként a STOP az END-hez hasonlóan kezelődik, s a Basichez való kilépést eredményezi. A STOP billentyű egy lefordított program megszakítására is használható, akárcsak a Basicben, de a lefordított program futása a megszakítás! ponttól már nem folytatható a CONTINUE segítségével. ÚJra kell indítani elölről, a START utasítással.

5.7 EXCEPTION handling (megszakítás kezelése)

Az EXCEPTION szó már önmagában is azt Jelenti, hogy valami hiba következett be, s ezért nehéz ezt lekezelni egy lefordított programban. Ennek ellenére a ZZZIP megkísérli egy jól használható EXCEPTION kezelő biztosítását. Nem valószínű, hogy a fordítás során problémák lennének, mivel a ZZZIP minden kapcsolódó parancsot elfogad, beleértve a CONTINUE és a RETRY parancsokat is. A lehetséges problémák futás során jelentkeznek. A CAUSE EXCEPTION parancs nyomán előállt helyzetek nem okozhatnak zavart, de ha egy másik forrásból, mint például egy EXOS-beli hibaellenőrzésből bekövetkező EXCEPTION után próbáljuk használni a CONTINUE parancsot, az eredményt illetően semmi biztosíték nincs!

A ZZZIP részéről két különleges korlátozás áll fenn. CsaK egyszintű EXCEPTION kezelés biztosított, más szóval a WHEN EXCEPTION parancsok egymásbaskatulyázása nem megengedett és bármely második megszakítás egy hibaüzenetet, valamint a Basicbe történő visszatérést eredményez. Kis mértékben a RETRY parancs is KülönböziK - a lefordított program ahelyett, hogy abba a sorba menne vissza, amelyikben az EXCEPTION előfordult, a legutóbbi WHEN EXCEPTION parancshoz fog visszatérni. Természetesen megfelelő elrendezéssel elérhetjüK a kívánt visszatérést, mégpedig úgy, hogy a WHEN EXCEPTION sort Közvetlenül az elé a sor elé tesszüK, amelybe vissza akarunk térni (például, egy INPUT sor elé).

4.β CHAIN, IMAGE, PRINT USING, TRACE, TYPE

EzeKet a parancsokat a ZZZIP nem fogadja el és használatuk a fordítás alatt hibaüzenetet eredményez.

5. BEÉPÍTETT FÜGGVÉNYEK

A ZZZIP megtart minden beépített függvényt, habár néhány esetben Korlátozza használatukat, ezekről külön említést teszünk. Talán meglepő lehet, hogy a ZZZIP megtartja a SIN és COS trigonometrikus függvényeket, hisz ezek valójában nem kompatibilisek az egész értékű matematikával. Ezeket használni is nehezebb, addig ne is foglalkozzunk velük, amíg nincs elegendő tapasztalatunk a ZZZIP működtetésében.

6.1 BIN(X)

Azokra az X értékekre, amelyek nem haladják meg a 32767 értéket (más szóval 0 és 11111 Között vannak), a függvény a szokásos módon használható a zárójelben lévő változó vagy konstans értékű operandussal. Nagyobb értékek esetén az operandus csak konstans lehet pl BIN(llOlOlOl), s ezt az esetet a ZZZIP megkülönböztetetten kezeli.

6.2CEIL(X), INT(X), IP(X), ROUND(X,N), TRUNCATE(X, N)

Mivel az egész értékű matematika nem veszi figyelembe a törteket, ezek a függvények gyakorlatilag nem csinálnak semmit, egyszerűen az X értékét adják vissza.

6.3 EPS(X)

Ez a függvény mindig az 1 értéket adJa vissza. Az 1 az egész értékű matematikában az értékváltás legkisebb egysége.

6.4 EXLINE

Egy lefordított programban a végrehajtás alatt álló sor számának a rekordja nem őrződik meg, s ezért ez a függvény mindig a 0 értéket adja vissza.

6.5 FP(X)

Mivel az egész értékű matematika nem veszi figyelembe a törteket, ez a függvény mindig a 0 értéket adja vissza.

6.6 FREE

A függvény a lefordított program által felhasználható byte-ok számát adja vissza, ahogy ezt el is várnánk. Ellenben, ha ez az érték meghaladja a 32767-et, negatív számként fog megjelenni.

6.7 INF

A függvény mindig a 32767 értéket adja vissza. Ez az előjeles, 16 bites bináris jelölésben a legmagasabb érték.

6.8 PI

A függvény mindig a 3 értéket adja vissza.

6. 9 RGB

Ez a függvény szokásos értelmezésében egy 0 és 1 közé eső értéksorozatot vár, mellyel az elsődleges színeket definiálhatjuk. Kivételes eset, hogy a ZZZIP elfogad ilyen értékeket, feltéve, hogy azok decimális formában lévő konstansok. Tehát az RGB(0,.4,l) elfogadható, ellenben az RGB(0,3/7,7/7) vagy az RGB(X,Y,Z) már nem.

Megjegyezzük, hogy a ZZZIP csak az első számjegyet használja a tizedespont után, s ezért az RGB(0,.42,.99)-et úgy tekinti, mint az RGB(0,.4,.9)-et.

6. 10 RND és RHD(X)

Az RND(X) pontosan úgy működik, mint a Basicben. Az RND önmagában mindig a 0 értéket adja vissza. Az RND*X különleges eset, amikor is a visszaadott érték megegyezik az RND(X) által szolgáltatott értékke 1.

6. 11 VERNUM és VER$

A VERNUM függvény mindig az 1 értéket adja vissza azért, hogy az EXOS 2.0 és 2.1 változatait ne lehessen összetéveszteni (az egész értékű matematikában a 2.0 és 2.1 mindegyike 2 lenne). A VER$ függvény azonban egy olyan karaktersorozatot ad vissza, amely az éppen használatban lévő EXOS változatát Jelzi.

6.12 Trigonometrikus és Logaritmikus függvények

Az olyan függvényeK, mint a COS és a SIN, valójában nem Kompatibilisek az egész értékű matematikával, mert értékeik mindig a O és 1 között helyezkednek el. Azért, hogy működjenek ezek a függvények, arányosító tényezőket fogunk használni. Ezeket a COS esetében részletezzük, bemutatva az elvet, majd megadunk egy olyan részletes táblázatot, mely minden érintett függvényt tartalmaz.

Hogy használható eredményeket Kaphassunk, a COS(X) által visszaadott érték automatikusan arányosítódik (beszorzódik) az l000-es tényezővel azért, hogy egy 0 és 1000 Közé eső egész értékké váljon (a 0 és 1 közé eső érték helyett). A program további részében ez az eredmény már használható, de ne felejtsük el, hogy egy arra alkalmas helyen osztanunk is Kell 1000-rel, hogy pontos eredményt kaphassunk. Természetesen az 1000-rel való osztás helyét gondosan Kell megválasztanunk, hogy elkerüljük az eredmény nullára való csökkentését, ez a hely a COS(X) értékének egy más értékkel való beszorzása után legyen. Ezt a külön lépést (az 1000-rel osztást) a Basic program szokásos vizsgálata után helyezzük el a programban, de még a fordítás előtt. Ha a Basicben is így vizsgálnánk, téves eredményt adna.

Ha a szögek meghatározásakor a DEGREES-t használjuk, a COS(X)-ben lévő X operandus megegyezik az eredeti Basic programban szerepelő értékével. Ellenben, ha a RADIANS-t használjuk, az X-et arányosítani (szorozni) kell 1000-rel a COS(X) használata előtt. Ismét ügyelnünk kell arra, hogy ezt a helyet gondosan válasszuk meg a programban, annak érdekében, hogy értelmes értéket kapjunk. Nem jó, ha hagyjuk, hogy az X értéke 1 alá essen és azután szorozzuk be 1000-rel, hisz ebben az esetben az X értékére eredményként mindig 0 adódik!

Függvény	Bejövő arányosító	Kimenő tényező
ACOS)		1 (fokok)
ASIN)	1000	vagy
ATN)		1000
		(radiánok)
COS)		
COT)	1	
CSC)	vagy	1000
SEC)	1000 (radiánoK)	
SIN)		
TAN)		
DEG	1000	1
RAD	1	lOOO
EXP	1000	1
LOG)		
LOG2)	1	lOOO
LOGlO)		
COSH)		
SINH)	1000	1000
TANH)		
		1 (fokok)
ANGLE(X, Y)	1	vagy
		1000

Megjegyezzük, hogy egyes trigonometrikus és logaritmikus függvények 32767-nél nagyobb értéket adhatnak (különösen, ha az arányosításnál megszorzódnak 1000-rel). Ezekben az esetekben a kijövő érték automatikusan 32767-re korlátozódik. Ugyanez vonatkozik a hatvány függvényre (^).

7. A LEFORDÍTOTT PROGRAMOK MÁSOLÁSA

A lefordított programok csupán a SAVE parancs használatával nem másolhatók szalagra. További másolatok készítésének egy módja az lenne, ha a Basic programot ismételten lefordítanánk. Egy másik módszer, amely csak egy szalagos készülék használatát igényli, azt a lehetőséget használja ki, amelyet az egyes lefordított programokat megelőző Basic betöltő program tartalmaz. A Basic képernyőeditorból indulva, a bent lévő programok törlése után, folytassa a következőkkel:

1) Helyezze be a lefordított programot tartalmazó szalagot és készüljön fel a betöltésre.

2) Gépelje be a LOAD "XXX"-et (ahol XXX a program neve).

3) Helyezze be a célszalagot (azt, amelyikre a másolatot szeretné kapni) és készüljön fel a kimentésre.

4) Gépelje be a SAVE ,,XXX"-et.

5) Helyezze be a lefordított programot tartalmazó szalagot és készüljön fel a betöltésre.

6) Gépelje be a RUN 100-at.

7) A program jelezni fogja, hogy mikor kell betennie a célszalagot és előkészülnie a kimentésre.

8) Ekkor nyomja le az ENTER billentyűt és a lefordított program kiíródik a célszalagra,


STUDIO
Boxsoft