TURBO BASIC INTERPRETER

ATARI 800XL/130XE

WERSJA DYSKOWA

W ponizszym opisie wystepuja nastepujace wyrazenia:

>>aexp<< = wyrazenie arytmetyczne

>>sexp<< = wyrazenie tekstowe (alfanumeryczne np. A$, "TEXT", CHR$(), STR$(), HEX$()).

>>lineno<< = numer linii

>>...<< = ciag instrukcji. W instrukcji IF...THEN chodzi o nastepujacy po THEN tekst. W innych wypadkach nie ma ograniczen. >>...<<< moze takze obejmowac kilka linii programu lub czesc linii.
W Basicu Atari istnieja wylacznie rozkazy skoku ukierunkowanego do numeru linii i * * * FOR...NEXT do tworzenia petli
W TURBO-BASIC wystepuja elementy strukturalnego programowania zapozyczone z jezyka PASCAL.

IF aexp THEN lineno

IF aexp THEN ...

Standardowy rozkaz IF...THEN

IF aexp ... ENDIF

IF aexp ... ELSE ... ENDIF

Gdy warunek aexp (<>0) jest spelniony, wykonywana jest czesc programu miedzy IF oraz ELSE. Gdy nie jest spelniony, czesc miedzy ELSE i ENDIF. ELSE moze zostac pominiete.
Koniec wyrazenia aexp oznaczany jest przez dwukropek (:) lub koniec linii (RETURN), a nie przez THEN.
Przed i po ELSE i ENDIF musi stac taki znak, (tzn.(:)).
Po ELSE nie moze byc numeru linii (w Basicu Atari w tego rodzaju przypadkach czesto konieczne jest wielokrotne uzycie rozkazu GOTO).

REPREAT ... UNTIL aexp

Ciag instrukcji >>...<< jest powtarzany do chwili spelnienia zadanego warunku. Sprawdzenie warunku nastepuje na koncu petli.
Czesc programu pomiedzy REPEAT i UNTIL bedzie zatem wykonana conajmniej raz.

WHILE aexp ... WEND

Powtarzany jest >>...<< tak dlugo, jak warunek jest spelniony, tzn. gdy warunek przestanie byc spelniany, petla nie bedzie wykonana.

DO ... LOOP

Petla bez konca. Wyrazenie >>...<< jest wciaz powtarzane.

EXIT

Opuszczenie petli, skok do konca petli. Rozkaz ten jest stosowany przy petlach >>DO...LOOP<< oraz >> REPEAT...UNTIL<<, >>WHILE...WEND<< jak tez >>FOR...NEXT<<. Rozkaz ten stanowi rodzaj "wyjscia awaryjnego" z petli, (przy >>DO...LOOP<< jedynego) dozwolonego w programowaniu strukturalnym. Spowodowany jest skok na koniec petli. Petle mozna takze przerwac przez >>POP:GOTO lineno<<. Takie przerwanie powinno byc wykorzystywane tylko w przypadkach awaryjnych, by nie pogarszac przejrzystosci programu.

* F

* F+

Po tym rozkazie nie sa wykazywane petle >>FOR...NEXT<< Tak wiec przed pierwszym obiegiem petli bedzie wystepowalo sprawdzenie, czy licznik osiagnie wartosc koncowa.
Np. >>FOR I=2 TO 1: ?I:NEXT I<<.

1. Wartoscia poczatkowa petli jest 2.
2. Podwyzszamy licznik petli o 1.
3. Porownanie z wartoscia koncowa (1)
4. Opuszczenie petli, poniewaz warunek zostal spelniony.

Po *F+ zmienna I bedzie zaladowana wartoscia poczatkowa 2, a nastepnie porownana ze zmienna I (wartosc koncowa = 1). Ostatecznie petla przeskakuje do rozkazu >>NEXT I<<. Nie nastepuje zadne wyjscie na ekran. Rozkaz >>F+<< czesto zaoszczedza specjalnej procedury wywolawczej.

*F-

Przywraca normalny status. Petle >>FOR...NEXT<< sa wykonywane conajmniej raz. Rowniez przy RUN rozkaz *F- jest wykonywany automatycznie (odpowiada to Basicowi Atari).

PROC nazwa

Poczatek procedury (podprogramu) o podanej nazwie.

ENDPROC

Koniec procedury odpowiada rozkazowi RETURN po >>GOSUB lineno<<.

EXEC nazwa

Wywoluje procedure o podanej nazwie. Rozkaz odpowiada >>GOSUB lineno<<. Normalne rozkazy GOSUB i RETURN pozwalaja na wywolanie podprogramu jedynie jednym numerem linii. Wywolanie procedury nastepuje poprzez podanie nazwy. Ponadto EXEC-PROC-ENDPROC zajmuje najczesciej mniej miejsca w pamieci niz standardowe rozkazy GOSUB-RETURN (nie mowiac juz o szybkosci).

Nazwy procedur sa magazynowane w pamieci jako nazwy zmiennych. Kazda nazwa zajmuje 8 bajtow + 1 bajt na znak. Kazde odwolanie sie wymaga tylko jeden (2) bajt. Numer linii instrukcji GOSUB zajmuje 7 bajtow. Mimo, ze TURBO-BASIC jest kompatybilny do ATARI-BASIC. Mozna obecnie uzyc zamiast 128 az 256 (!) roznych zmiennych lub nazw procedur. Od 129-ej zmiennej odwolanie sie wymaga dwu bajtow (dotychczas jednego) pamieci.

ON aexp EXEC pnazwa, pnazwa...

Odpowiada rozkazowi BASIC >>ON GOSUB<<. pnazwa = PROCedurynazwa.

nazwa

GO # nazwa

ON aexp GO # naz, naz, naz

TRAP # nazwa

RESTORE # nazwa

Definicja etykiet, ktorej # odpowiada PROC. W TURBO-BASIC niestrukturalny skok GOTO staje sie czytelny przez to, ze "GOTO lineno" zostaje zastapiony przez "GOTO # nazwa". Przy TRAP i RESTORE mozna stosowac znaczniki >>#<< aby informowac interpreter, ze nie jest to numer linii, a nazwa. Ponadto >>GOTO nazwa<< dziala szybciej niz >>GOTO lineno<<. Znacznikiem dla GOTO#, TRAP# i RESTORE# jest znak "#", podobnie jak w przypadku EXEC z procedurami.

Ten scisly rozdzial sluzy podwyszeniu przejrzystosci programow.

POP

POP sluzy do tworzenia podprogramu z GOSUB i EXEC oraz petli >>FOR...NEXT<<, >>REPEAT...UNTIL<<, >>WHILE...ENDWHILE<< oraz >>LOOP...ENDLOOP<<.
Na stosie GOSUB, EXEC, REPEAT, WHILE i LOOP zajmuja 4 bajty (tak jak GOSUB w ATARI-BASIC). >>FOR...NEXT<< zajmuje 13 zamiast 12 bajtow, a to za przyczyna stojacych do dyspozycji 256 (!) roznych zmiennych. Jednak obecnie na stosie nie sa juz skladowane numery linii (przyczyna powolnosci ATARI-BASIC), a jedynie adresy tych linii w pamieci. Tym samym petle wykonywane sa szybciej. Ponadto predkosc powtarzania petli nie spada wraz z rosnacym oddaleniem od poczatku programu (!).
Mimo to podtrzymano jedna z zalet ATARI-BASIC: program zatrzymany np. przez blad mozna edytowac (poprawiac na biezaco) i kontynuowac wykonanie przez CONT lub GOTO.
Przy tym ani zmienne, ani stos nie sa kasowane.
W ten sposob mozna pisac programy, ktore samoistnie generuja linie data albo kasuja niepotrzebne juz czesci programu.
(przy POKE 842,13 komputer zachowuje sie tak, jakby caly czas nacisniety byl klawisz RETURN; POKE 842,12 przywraca normalny stan systemu).
Do kasowania czesci programu w TURBO-BASIC sluzy rozkaz DEL.

-
-

Jest to specjalne polecenie REM. Wszystko co w linii programu nastepuje po >>-<< nie bedzie brane przez komputer pod uwage.
Przy rozkazie LIST ukaze sie nie dwa, a 30 znakow minus.
Rozkaz ten zajmuje nawet jeden bajt pamieci mniej niz jeden rozkaz REM bez tekstu.
W ATARI-BASIC odpowiadajacy rozkaz REM zajmuje znacznie wiecej miejsca w pamieci.

LIST

Przy listowaniu programow petle zostana optycznie uwypuklone przez przesuniecie dwoch spacji. Przez to programy sa bardziej przejrzyste. Ponadto w ten sposob mozna uniknac wielu bledow.
Przy niestarannym programowaniu (zbyt wiele NEXT do jednego FOR lub zbyt wiele ENDPROC do jednej PROC) lilinie staja sie rowniez nieporzadne.
Takie konstrukcje, ktore w ATARI-BASIC i wielu innych wersjach Basicu sa trudne do opisania, w TURBO-BASIC moga byc zastapione przez IF-ELSE-ENDIF lub EXIT. Rezultatem sa latwo czytelne programy.

* L-

Wylacza tabulacje. Moze to byc przydatne przy edytowaniu dlugich linii programowanych lub dla zaoszczedzenia miejsca przy magazynowaniu (save'owaniu) na dysku (LIST "D:X"). Znak >>-<< po >>*L-<< bedzie listowany jako podwojny, a nie potrojny minus.

* L

* L+

Wlacza ponownie tabulacje (normalny stan po zaladowaniu interpretera). Ponadto uruchamia to LIST od danej linii do konca programu. >>LIST 3000,<< listuje program od linii 3000 do konca programu, >>LIST "P:",3000,<< odpowiednio na drukarke (pojedynczy przecinek nakazuje komputerowi automatyczne uzupelnienie najwyzszego mozliwego numeru linii 32767.

****

NOWE KOMUNIKATY BLEDOW

****

Istnieja nastepujace nowe komunikaty bledow:

ERROR - 22 ?NEST

Tzw. zagniezdzenie. Wystepuje gdy nie zostaje znaleziona instrukcja ENDWHILE w petli WHILE lub ENDIF z warunku IF z petli FOR (po >>*F+<<).
Po opuszczeniu podprogramu (przez RETURN lub ENDPROC) petla zostaje przerwana.
Nastepuje to w ten sam sposob jak przy >>FOR...NEXT<< czy innych petlach stojacych do dyspozycji w TURBO-BASICU.

ERROR - 16 ?GOSUB

Brak RETURN do istniejacej instrukcji GOSUB.

ERROR - 13 ?FOR

Brak FOR do istniejacej instrukcji NEXT.

ERROR - 23 ?WHILE

Brak WHILE do istniejacej instrukcji WEND.

ERROR - 24 ?REPEAT

Brak REPEAT do istniejacej instrukcji UNTIL.

ERROR - 25 ?DO

Brak DO do istniejacej instrukcji LOOP.

ERROR - 28 ?EXEC

Brak EXEC do istniejacej instrukcji ENDPROC.

ERROR - 29 ?PROC

Zostala wywolana nieznana procedura.

ERROR - 30 ?#

Zostal zastosowany nieznany znak.

ERROR - 27 XPROC

(Wykonanie PROC). Ten komunikat bledu wystepuje, gdy wykonywany jest rozkaz PROC. Procedury powinny byc wywolywane poprzez rozkaz EXEC.

ERROR - 26 ?EXIT

EXIT ustawiony poza petla.

ERROR - 15 ?DEL

Zostal skasowany rozkaz GOSUB do RETURN, NEXT, FOR, REPEAT lub UNTIL.
W TURBO i ATARI BASIC programy pozwalaja sie edytowac bez zniszczenia wartosci zmiennych i stosu. Ww. blad moze wystapic, gdy przy powrocie z podprogramu (petli) odpowiednia linia zostanie zmieniona lub skasowana. Wystepuje to takze, gdy podprogram zawiera w sobie rozkaz DEL, powodujacy autokasacje.
Po numerze bledu nastepuje komentarz (w przeciwienstwie do ATARI-BASIC, w ktorym nie ma zadnych komentarzy).

np.

```
>>138 TIMEOUT<<,
```

>> 29 ?PROC << itd.

Wyczerpujace informacje na ten temat znajduja sie w podreczniku ATARI-BASIC lub instrukcji DOS.
Dluzsze teksty (sposrod 60 stojacych do dyspozycji komunikatow bledow) znacznie zwiekszyly by obszar zajmowany przez niniejszy interpreter.

DEL od,do

Kasuje linie programu od, do (wlacznie).

RENUM stara,nowa,inkr

Przenumerowuje wszystkie linie programu. Nowe linie programu rozpoczynaja sie od 'nowa' i sa zwiekszane o 'inkr'.
Numeracja linii przed 'stara' pozostanie niezmieniona.

UWAGA! Rozkaz ten zmienia numery linii po GOTO, GOSUB, TRAP,

RESTORE, LIST, DEL, ON-GOTO oraz ON-GOSUB.

Przy niezdefioniowaniu numeru linii w odpowiednie miejsce zostanie wstawiona liczba ujemna np. GOTO-100.
Przy obliczaniu skokow (GOTO VAR, GOSUB 100+10*A, RESTORE A*10+1000) komputer odmawia wykonania RENUM.

Gdy po instrukcji wystepuje wyrazenie >>GOTO 1000 + A*10<<, jest ono traktowane jako linia GOTO 1000.
Pozostala czesc linii pozostaje niezmieniona. Gdy nie nastepuje liczba, nazwa zmiennej lub nawias - rozkaz nie zostanie zmieniony.

DUMP

DUMP nazwa

Rozkaz ten tworzy liste zastosowanych zmiennych. Tak jak przy LIST moze nastapic wydruk na drukarce (DUMP"P:")

np.

A=10	zmienna numeryczna
B(10,1	tablica, DIM B(9) lub DIM B(9,0)
C(0,0	tablica niezdefiniowana; przy tablicach oba mozliwe wymiary podwyzszane sa stale o jeden.
D(10,10	DIM D(9,9)
E$ 10,20	zmienna alfanumeryczna, LEN=10, DIM E$(20)

```
F$ 0,0        nie zdefiniowana zmienna alfanumeryczna

G$ 0,10       DIM G$(10), LEN(G$)=0

H PROC 100    PROC H w linii 100

I # 120       znacznik I w linii 120

J ?           niezdefiniowany znacznik lub procedura
```

Wydruk zmiennych/znacznikow nastepuje w kolejnosci ich save'owania.

TRACE

TRACE +

Wlacza tryb TRACE (sledzenie). Numery kolejno wykonywanych linii beda ukazywac sie w kwadratowych nawiasach u dolu ekranu.

TRACE -

Znosi tryb TRACE. Zniesienie trybu TRACE **wystepuje takze przy** wystapieniu bledu. Przy wywolaniu EXEC lub GO # nie beda podawane numery linii procedury ani GO#.

* B
---

* B +

Po tym rozkazie nacisniecie klawisza BREAK bedzie traktowane jako blad. By ochronic program przed przypadkowym przerwaniem, mozna zastosowac rozkaz TRAP.

* B -

Znosi wyzej opisany tryb. Przy rozkazie RUN, zostaje automatycznie wykonane >>* B -<<

****

ROZKAZY

****

<=> odpowiada normalnemu ATARI-BASIC

DPOKE adr,slowo

POKE-dwu-bajtowy <=> POKE,adr,slowo - 256*INT(slowo/256):POKE adr+1,INT(slowo/256)

MOVE zrodlo,cel,licznik

Transfer bloku <=> FOR I = 0 TO LICZNIK-1: POKE

cel+1,PEEK(zrodlo+I):NEXT I
Przez >>MOVE 57344, nowy zestaw znakow,1024 mozna np. kopiowac zestaw znakow.

- MOVE zrodlo,cel,licznik

Transfer bloku. Najpierw przesuwany jest ostatni bajt do drugiego o wyzszym adresie. Umozliwia to przesuniecie jednego obszaru pamieci bez nakladania sie, co moglo by powodowac przeklamania (gdy zrodlo+licznik>cel).
<=> FOR I=LICZNIK-1 TO 0 STEP -1:POKE cel +I,PEEK(zrodlo+I):NEXTI.
MOVE mozna zastosowac do zapelnienia obszaru pamieci np. >>POKE DPEEK(88),128,MOVE DPEEK(88),DPEEK(88+1,959)<<. Ww. rozkazy calkowicie wypelniaja ekran kodem spacji (w inversie).
Mimo, ze ma tu miejsce szereg niepotrzebnych przebiegow ladowania, jest to znacznie szybsze niz petle w BASIC'u.

BPUT #n,adres,dlugosc

Zapis bloku <=> FOR I=0 TO dlugosc-1:PUT #n,PEEK(adres+I):NEXT I

BGET #n,adres,dlugosc

Czytanie bloku <=> FOR I=0 TO dlugosc -1.GET #n,A:POKE adr+I,A:NEXTI
Tymi rozkazami mozna z maksymalna predkoscia ladowac i magazynowac cale obszary pamieci.
Np.

```
OPEN #1,8,0,"D:BILD.PIC":BPUT #1,DPEEK(88),7680:CLOSE #1

OPEN #1,4,0,"D:BILD.PIC":BGET #1,DPEEK(88),7680:CLOSE #1
```

Magazynowanie lub ladowanie obrazow grafiki-8(lub 9,10,11,15) na dysk.

UWAGA! Liczba 7680 musi byc modyfikowana w zaleznosci od trybu graficznego (w przeciwnym wypadku laduja lub magazynuja sie obszary nie nalezace do pamieci ekranu).

FILLTO x,y

Krocej, szybciej i przejrzyscie niz:

>>POSITION x,y:XIO 18, #16,0,0,"S:"FCOLOR n<<
Wybor koloru dla FILLTO.
W ATARI-BASIC przybiera to postac rozkazu >>POKE 765,n<<

CLS

CLS #6

Czyszczenie ekranu. CLS <=> A=PEEK(766):POKE 766,0:POSITION 0,0:? (#6;)chr$(125);:POKE 766,A

PUT n

(=) ?CHR$(n); przykladowo >>PUT 253<< za >>? CHR$(253);<<
Przy PUT, GET, INPUT ... mozliwe jest podanie >>#Ø<<
Uzywanie >>IOCB Ø<< (z >>CLOSE<< lub >>OPEN #Ø<<) przeszkadza rutynowym czynnosciom edytora ekranowego. W wymienionym przypadku nacisnac klawisz RESET. Jezeli przy PUT brakuje #, automatycznie uzywany jest >>IOCB #Ø<<

GET KEY

(=) OPEN #7,4,Ø,"K:":GET #7,KEY:CLOSE #7. Komenda ta oczekuje na nacisniecie klawisza. W przypadku nacisniecia klawisza nastepuje przypisanie zmiennej KEY wartosc kodu ATASC II (nazwa zmiennej moze byc nadana dowolnie)

DIM

Przy instrukcji DIM tablice i zmienne alfanumeryczne sa kasowane - ustawione na Ø.

DIM A(1ØØ) (=) DIM A(1ØØ):FOR I=Ø TO 1ØØ:A(I)=Ø:NEXTI

INPUT "text",var,var...

INPUT "text";var,var...

INPUT jest podobny do odpowiedniego rozkazu INPUT w MICROSOFT-BASIC.
W INPUT * * * 'TEXT' mozna zaoszczedzic rozkazow PRINT.
Gdy po tekscie wystepuje srednik zamiast przecinka uzyskujemy dodatkowo jeden ? (w programowaniu 65Ø2 i 651Ø znak zapytania /?/ oznacza PRINT przyp.aut.)
Przez >>INPUT"";A<< otrzymujemy INPUT bez przeszkadzajacego czasami >>?<<

TEXT x,y,se>p

Przy wpisywaniu tekstu w ekran graficzny >>x,y<< oznacza pozycje gornego, lewego rogu pierwszego znaku wyrazenia alfanumerycznego (liczonego w pixlach).
Np.

GRAPHICS 8:TEXT 5Ø,9Ø,"TURBO BASIC":TEXT 9Ø,95,1ØØØ<<

W przeciwienstwie do normalnego rozkazu PRINT po TEXT moze wystepowac tylko jedno wyrazenie (nie lista z przecinkiem czy srednikiem).
Ponadto text jest na koncu linii lamany, nie wystepuje zatem scrolling.

CIRCLE XØ,YØ,r

CIRCLE xØ,YØ,xr,yr

Rozkaz ten rysuje kolo o srodku xØ,xØ i promieniu r.

Przy podaniu drugiego (roznego) promienia dla kierunku x i y powstaja elipsy.

PAINT x,y

Wypelnia zamkniety obszar (np. okrag) wybranym kolorem (poprzez rozkaz COLOR).

.
Rozkaz ten moze wypelnic kolorem niemal kazda figure.
Poniewaz chodzi tu o dopasowana do 6502/6510 rekursywna funkcje istnieje pewne ograniczenie (miejscem w pamieci) kompleksowosci wypelnianj figury.
W ekstremalnym przypadku konieczne jest 90 kB RAM (!).
Rowniez prostsze i mniejsze figury zajmuja setki bajtow. W przypadku przekroczenia pojemnosci pamieci (FRE(Ø) za male), podany zostaje meldunek >>ERROR-2MEM<<

Aby umozliwic proste obliczanie wspolrzednych wielu lezacych obok siebie punktow, TEXT i PAINT uzywaja wlasnych szybko-kreslacych procedur.
Niestety system operacyjny ATARI nie moze zrobic z nich uzytku.

TIME $

Patrz nizej.

PAUSE n

Przerywa wykonanie programu na n/50 sekundy.
PAUSE zastepuje niedokladne i pochlaniajace miejsce petle opozniajace FOR-NEXT.
W ATARI-BASIC dla dokonania malych opoznien stosuje sie potegowanie (A=1^1).
Poniewaz TURBO-BASIC jest wyjatkowo szybki, przy potegowaniu nalezy uzywac np. >>PAUSE 9<<

DSOUND glos,czest,dis,poziom

Podobny do normalnego rozkazu SOUND.ATARI moze laczyc dwa normalne glosy. Rozdzielczosc czestotliwosci wynosi 16 bitow (Ø.....65535) zamiast 8 bitow (Ø...255). Uzyskiwana czestotliwosc jest obliczana wg.wzoru:

1789790/(2*czestotliwosc + 14)

zamiast 63921/(2*czestotliwosc + 2)

Te wartosci pochodza ze zrodel amerykanskich i moga nieznacznie roznic sie od wartosci przy innych wersjach ATARI, poniewaz sa dzwiek oddzielany jest od czestotliwosci nosnej video (inne wartosci w USA - system TV - NTSC, inne przy europejskiej wersji - system TV - PAL).

SOUND

DSOUND

Skrocona forma dla:FOR I=Ø TO 3:SOUND I,Ø,Ø,Ø:NEXT I

CLOSE

Skrocona forma dla:FOR I=1 TO 7:CLOSE #I:NEXT I

****

FUNKCJE

****

DPEEK (adr)

Dwu-bajtowy-PEEK <=> PEEK (adr) + 256*PEEK(adr + 1)

INKEY $

Zmienna specjalna odczytujaca znaki z bufora klawiatury. Gdy zostaje nacisniety klawisz, INKEY $ zostaje przypisany odpowiedni znak. W przeciwnym przypadku otrzymuje (""). W ten sposob mozna opracowac nacisniecie klawisza bez przerywania programu.

INSTR (A$,B$)

INSTR (A$,B$,1)

Poszukuje zmiennej alfanumerycznej B$ w (dluzszej) zmiennej alfanumerycznej A$.

Jezeli ja odszuka dostarcza do A$ pozycje zmiennej B$. W przeciwnym wypadku Ø.
>>i<< ustawia index (pozycje) od ktorego ma nastapic przeszukiwanie.

UINSTR (A$,B$)

UINSTR (A$,B$,i)

Podobnie do INSTR. Bity 7 i 5 pojedynczego znaku nie sa rozpoznawane. Przy poszukiwaniu "MODEM" mozna takze odszukiwac >>Modem<<, >>MoDeM<< lub odpowiednio ten text w inversie. UINSTR dotyczy duzych liter INSTR).
Jako efekt uboczny przy przeszukiwaniu liczb lub zestawow znakow wystepuje wyszukiwanie znakow specjalnych ("!"=CTRL-A, "Ø"=CTRL-P itd.)

ERR

Skrot od >>PEEK(186)+256*PEEK(187)<< lub >>DPEEK(186)<<. Rozkaz informujacy o linii w ktorej wystapil blad. ERR i ERL winny byc stosowane w procedurach TRAP.

TIME

Specjalna zmienna odmierzajaca czas rzeczywisty (z wewnetrznego zegara (RTCLCK) komputera ATARI o "taktowaniu" 1/5Ø sekundy) pochodzacy z komorek pamieci od 18 do 2Ø.

TIME$

Specjalna zmienna zawierajaca czas jako 6-pozycyjna zmienna alfanumeryczna o formacie hhmmss (hh = godzina ØØ do 23, mm = minuta ØØ do 59, ss = sekunda ØØ do 59)

TIME$=

Sluzy do ustawienia zegara.
>>TIME$="151520"<< ustawia zegar na godzine 15, 15 minut i 2Ø sekund. Zmienna TIME nie pozwala na bezposrednia zmiane, dlatego tez korzysta sie z >>TIME$=<< lub odpowiednie POKE w pozycje 18 i 2Ø pamieci.
Zegar nie chodzi calkiem dokladnie, jako ze czestotliwosc otrzymywanego z ATARI obrazu TV nie wynosi dokladnie 5Ø Hz. TIME$ zalezy wlasnie od tej czestotliwosci.

FRAC(exp)

Funkcja ta informuje o czesci liczby dziesietnej po przecinku.
>>FRAC(exp) nie zawsze rowna sie >>exp-INT(exp)<<, poniewaz INT podaje kolejna najmniejsza liczbe, a wiec >>?INT(-Ø.3)<< -1, >>? FRAC(-Ø.3)<< podaje Ø.

TRUNC(exp)

Funkcja ta podaje czesc calkowita liczby. Jest to funkcja komplementarna do FRAC.
>>? TRUNC(-Ø.3)<< daje Ø

RND

RND(cokolwiek) moze byc w TURBO-BASIC skrocony przez usuniecie nawiasow.
Uzywa sie zatem RND zamiast RND(Ø)

RND(n)

W skrocie jest to >>TRUNC(RND(Ø)*n) i generuje liczbe przypadkowa pomiedzy Ø (wlacznie), a n (wylacznie).

HEX$(exp)

Rozkaz zblizony do STR$. HEX$ zamienia liczbe calkowita exp(Ø<=exp<=65535) na hexadecymalna zmienna alfanumeryczna. Gdy exp jest mniejsza od 256, zamiana daje 2-znakowy, w przeciwnym wypadku - 4-znakowy ciag.

DEC(sexp)

Rozkaz podobny do VAL. Odwrotnosc HEX$.Zmienna tekstowa sexp zostaje zamieniona na calkowita liczbe dziesietna. Gdy sexp zawiera wiecej niz 4 znaczace cyfry hexadecymalne - brane sa pod uwage tylko ostatnie cztery.

$ aaaa

Hexadecymalne liczby w programie.

Np.

```
FOR I=$Ø6ØØ TO $Ø67F

READ A

POKE I,A

NEXT I
```

zamiast:

```
FOR I=1536 TO 1663
READ A
POKE I,A
NEXT I
```

>>&<< BINARNE AND (i)

>>!<< BINARNE OR (lub)

>>EXOR<< BINARNE OR WYLACZNE

Ww. 3 operatory pracuja z 16 bitowymi liczbami calkowitymi pomiedzy Ø a 65535, lecz nie wielkosciami boolowskimi (1 lub Ø) jak i operatorami logicznymi AND, OR i NOT.

DIV

Dzielenie bez reszty: a DIV b <=> TRUNC(a/b)

MOD

Okreslenie reszty z podzialu:

a MOD b <=> a-b*TRUNC(a/b)

%Ø %1 %2 %3

Liczby Ø do 3 sa zdefiniowane jako stale. Zastosowanie liczby (takze $aaaa) w programie wyznacza czesto 7 bajtow. Uzycie zmiennej i niezaleznie od dlugosci nazwy tylko 1 lub 2 bajty. Rowniez uzycie %Ø do %3 zajmuje 1 bajt, ale nie powoduje naniesienia zmiennych w tablice (ograniczonzch do 256).
W zmiennych tekstowych pomiedzy znacznikami >>"<< (odpowiada CHR$(34)) mozliwe jest podwojne ("") skladanie.

Np.

```
>>? "TEST""TEXT"<< uzyskuje sie wyrazenie
>>TEST"TEXT"<<
```

Aby otrzymac ten efekt w ATARI-BASIC nalezy napisac:

```
>>? "TEST";CHR$(34);"TEXT"<<
```

Przy podstawieniu (A$="TEST""TEXT") jest to jeszcze bardziej rozwlekle.

TURBO-BASIC automatycznie zamienia przy wprowadzaniu linii programu litery male na duze. Nie dotyczy to oczywiscie znakow w cudzyslowiu jak tez linii REM i DATA.
Mozna przez to wprowadzac programy malymi literami bez ciaglego przelaczania klawisza CAPS czy SHIFT.
W zmiennych i nazwach procedur poza literami istnieje znak podkreslenia >> << (SHIFT-).
Tak wiec obecnie nazwy typu MAX LEN czy PROC SORT ADRES sa dopuszczalne.

****

ROZKAZY DOTYCZACE DYSKU

****

DIR

DIR"D1:*.*"

Ukazuje na ekranie spis zbiorow (directory) zawartych na dysku. >>DIR""D1:*.*"<< listuje wszystkie zbiory zmagazynowane na dysku 1, >>D2:A*.*<< wszystkie zbiory na dysku 2, ktorych nazwa rozpoczyna sie od A. >>D1:*.*<< moze zostac pominiete (podobnie jak w DOS opcja >>A RETURN RETURN<<. Zmienne alfanumeryczne sa dopuszczalne przy tym rozkazie (jak i przy nastepnych).

RENAME "D:STARY,NOWY"

Zmienia nazwe zbioru ze 'STARY' na 'NOWY'. Odpowiada to >>XIO 32,#7,0,0,"D2:STARY,NOWY" lub opcje E z Menu DOS.

DELETE "D:ZBIOR"

Kasuje zbior (odpowiada to >>XIO 33<< lub opcji D z Menu DOS.

LOCK"D:ZBIOR"

Zaklada protekcje na zbior (przeciwko ponownemu zapisaniu tego obszaru). Odpowiada to >>XIO 35<< lub opcji F z Menu DOS.

UNLOCK"D:ZBIOR"

Ponownie znosi protekcje. Odpowiada to >>XIO 36<< lub opcji G z Menu DOS.

BLOAD"D:FILE.OBJ"

Laduje zbior binarny. Odpowiada to opcji L z Menu DOS z >>/N<< po nazwie zbioru.

BRUN"D:FILE.OBJ"

Laduje zbior binarny i uruchamia go, gdy adres RUN znajduje sie w tym zbiorze. Odpowiada to opcji L z Menu DOS, bez nastepujacego >>/N<<.

Oprocz opisanych wyzej rozkazow i komend stoja do dyspozycji wszystkie pozostale rozkazy z normalnego BASICu.

UWAGI KONCOWE

Ladowanie TURBO-BASIC nastepuje przez opcje L z DOS 2.0 lub 2.05.
Do dyspozycji programisty stoi 34 021 bajty RAM, a wiec 1747 bajtow wiecej niz przy programowaniu w normalnym ATARI-BASIC. Jak powiedzielismy, TURBO-BASIC jest kompatybilny z ATARI-BASIC. Nalezy jednak uwazac przy laczeniu programu napisanego w Basicu z procedurami pisanymi w jezyku maszynowym (aby nie zajac przypadkowo obszaru pamieci, gdzie zmagazynowany jest niniejszy interpreter).