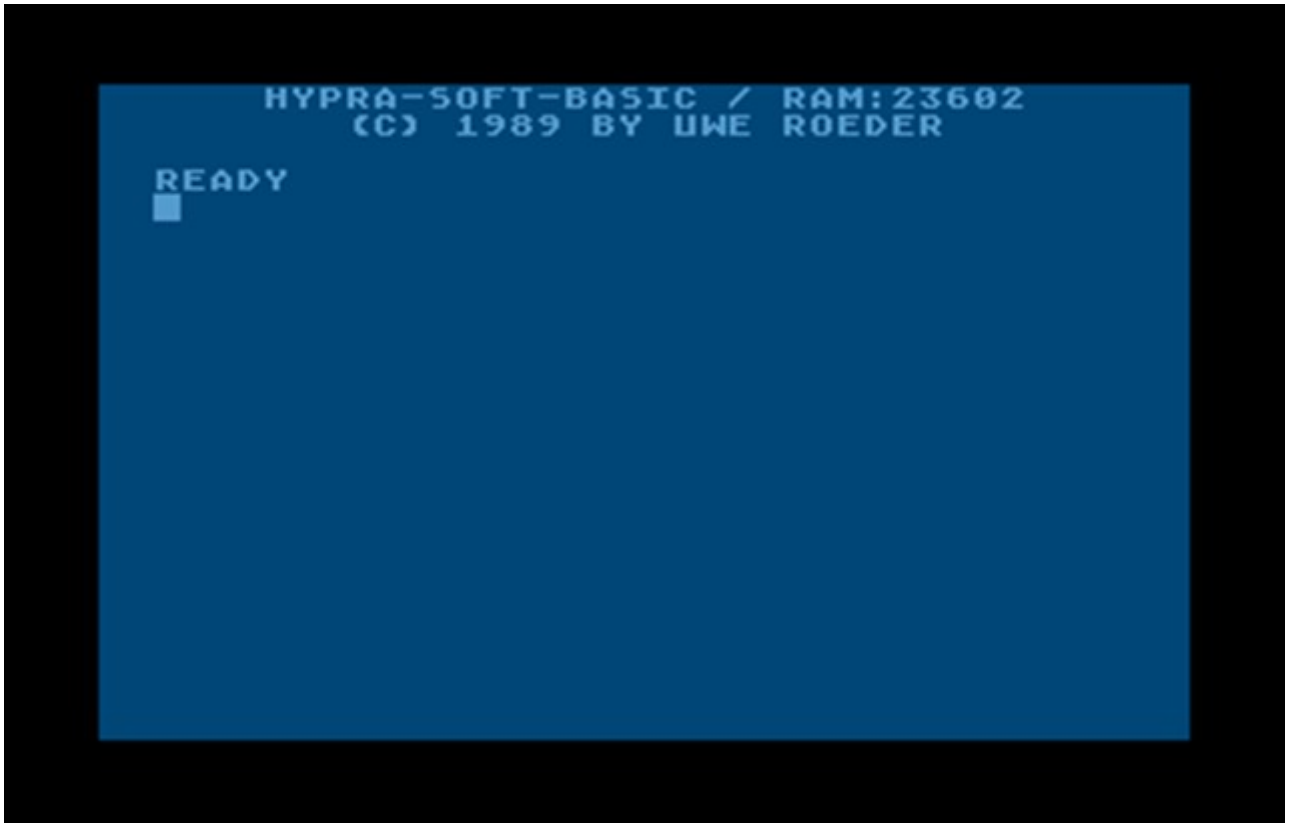

Hypra Soft BASIC wersja 1.0

Opis dodatkowych poleceń

Bluki, 2015

Hypra Soft BASIC v. 1.0

Opis dodatkowych poleceń

Ten język jest rozszerzeniem Atari BASIC-a. Oznacza to, że wszystkie dotychczasowe polecenia Atari BASIC-a są tu dostępne. Ponadto posiada 48 nowych poleceń zapewniających bardziej oszczędne korzystanie z pamięci, a w niektórych przypadkach większą szybkość wykonywania programu. W szczególności Hypra Soft BASIC został wyposażony w polecenia do obsługi grafiki player-missile i zaawansowane operacje na pamięci. Chociaż zajmuje od Atari BASIC-a o 8 kB więcej pamięci, to dzięki tym dodatkowym poleceniom możliwe jest uzyskanie bardziej zwięzłego kodu, co z nawiązką może zrekompensować mniejszą dostępną pamięć RAM.

Pomimo pełnej zgodności programowej z Atari BASIC-em ze względu na wspomniany nieco mniejszy dostępny RAM i inną lokalizację RAMTOP niektóre programy napisane w Atari BASIC-u nie będą się uruchamiać lub mogą działać niepoprawnie pod Hypra Soft BASIC-em.

Poniżej zostały przedstawione dodatkowe polecenia różniące Hypra Soft BASIC od Atari BASIC-a. Przyjęta została następująca konwencja:

- Nazwa polecenia.
- Format.
- Przykład (przykłady) – jeżeli zachodzi taka potrzeba.
- Opis polecenia.

Spis treści

System, edycja programu

BASIC

```
BASIC n
BASIC 1
```

Może się tak zdarzyć, odczytując program w Atari BASIC-u, że znajdą się w nim zmienne o nazwach będących słowami kluczowymi w Hypra Soft BASIC-u. Dlatego przed zmienną, do której następuje podstawienie wartości, zostaje automatycznie wstawione specjalne polecenie „Let” (nie mylić z „LET” - pisane dużymi literami). Jeżeli teraz program zostanie zapisany, to będzie on po ponownym odczycie czytelny dla Hypra Soft BASIC-a, ale nieczytelny dla Atari BASIC-a, ponieważ nie występuje w nim polecenie „Let”. Dlatego przed zapisem należy ustawić poleceniem BASIC odpowiedni tryb zapisu:
– BASIC `0` – zapisany program można będzie uruchamiać bez problemów pod Hypra Soft BASIC-em (stan po starcie systemu);
– BASIC 1 – zapisany program można będzie prawidłowo uruchamiać pod Atari BASIC-em.

BOOT

```
BOOT
```

Wywołuje zimny start. Jeżeli Hypra Soft BASIC znajduje się na kartridżu, wówczas pozostaje w pamięci.

ERL

```
ERL
? ERL
```

Podaje numer linii, w której wystąpił błąd.

ERR

```
ERR
? ERR
```

Informuje o numerze błędu, jaki wystąpił w czasie wykonywania programu.

HANDLER

```
HANDLER
```

Wyświetla tabelę wszystkich aktywnych sterowników (edytor, klawiatura, ekran i in.) wraz z ich identyfikatorami (E:, K:, S: ...).

HELP

```
HELP
```

Wyświetla listę wszystkich poleceń, które nie występują w Atari BASIC-u.

INFO

```
INFO
```

Wyświetla, które obszary pamięci są zajęte. Może mieć to znaczenie dla użytkownika przy rozmieszczaniu własnych zbiorów danych.

IOCB

```
IOCB
```

Wyświetla, które kanały są w użyciu i dla jakiego kierunku zostały otwarte.

KLICK
```
KLICK n
KLICK 1
```
Wyłącza (n=0) lub włącza (n=1) klik z klawiatury.

REN
```
REN nowy_numer_linii,skok
REN 100,10
```
Przenumerowuje program, nadając pierwszej linii programu numer określony przez *nowy_numer_linii*. Następna linia programu otrzymuje numer równy *nowy_numer_linii* + *skok* itd.

SYSTEM
```
SYSTEM
```
Wyświetla ważniejsze adresy (wektory) systemu operacyjnego. Użytkownik może np. sprawdzić, czy i w jaki sposób są używane przerwania albo pod jakim adresem znajduje się display list.

VAR
```
VAR
```
Drukuje listę zmiennych. W przypadku tabel i ciągów podaje ich maksymalny rozmiar.

Sterowanie przebiegiem programu

BREAK
```
BREAK n=m
BREAK 8=8
BREAK 3=1
```
Jeżeli wyrażenie logiczne n=m jest prawdą (czyli liczba n jest taka sama jak m) to klawisz Break działa normalnie, jeżeli wyrażenie logiczne jest fałszywe (n<>m) to klawisz Break jest zablokowany.

DO … OD
```
DO [dowolne polecenia lub linie programu] OD
DO :?"TEST PETLI":OD
```
Pętla bezwarunkowa. Wszystkie polecenie znajdujące między słowem kluczowym DO a OD będą powtarzane bez końca. Jedynym sposobem jej przerwania jest polecenie EXIT.

EXIT
```
EXIT numer_linii
EXIT 1500
```
Przerywa pętlę DO … OD i wykonuje skok do linii o wskazanym numerze.

REPEAT ... UNTIL
```
REPEAT [dowolne polecenia lub linie programu] UNTIL warunek
REPEAT :L=L+0.5:? L:UNTIL L=12.5
```

Pętla warunkowa. Ciąg instrukcji zawarty pomiędzy REPEAT a UNTIL będzie powtarzany tak długo, jak długo warunek nie jest spełniony. Jeżeli warunek (może to być wyrażenie logiczne) zostanie spełniony, pętla zostanie zakończona (program przejdzie do następnego polecenia po UNTIL).

Operacje na pamięci

BGET

```
BGET #n,adres,ile
BGET #1,AD,X
X=DPEEK(88):Y=512:BGET #1,X,Y
```

Pobiera dane ze wskazanego kanału (#n) i przesyła je pod wskazany *adres*. Ilość przesyłanych bajtów określa trzeci parametr. Uwaga: 2. i 3. parametr musi być zmienną. Należy pamiętać o wcześniejszym otworzeniu do odczytu wybranego kanału (polecenie OPEN).

BPUT

```
BPUT #n,adres,ile
BPUT #1,AD,X
X=DPEEK(88):Y=960:BPUT #1,X,Y
```

Wysyła dane przez wskazany kanał (#n) na urządzenie zewnętrzne, począwszy od *adresu* w pamięci. Ilość przesyłanych bajtów określa trzeci parametr. Uwaga: 2. i 3. parametr musi być zmienną. Należy pamiętać o wcześniejszym otworzeniu do zapisu wybranego kanału (polecenie OPEN).

DPEEK

```
DPEEK(adres)
? DPEEK(88)
```

Odczytuje słowo spod wskazanego *adresu* (patrz DPOKE).

DPOKE

```
DPOKE adres,słowo
DPOKE 560,30720
```

Dwubajtowe POKE. Wpisuje *słowo* (0 ÷ 65535) pod wskazany *adres* w kolejności młodszy – starszy bajt.

MOVE

```
MOVE+ skąd,dokąd,ile
MOVE- skąd,dokąd,ile
A=1536:B=DPEEK(88):X=640:MOVE+ A,B,X
MOVE- A,B,640
```

Kopiuje blok pamięci, począwszy od adresu *skąd*, pod adres *dokąd*. Ilość kopiowanych bajtów określa parametr *ile*. MOVE+ kopiuje dane w kolejności od pierwszego do ostatniego bajtu. MOVE- odwrotnie. Ma to znaczenie, gdy adres *dokąd* jest mniejszy niż *skąd* + *ile*. Używając wtedy MOVE- unika się nadpisania danych.
Uwaga: 1. i 2. parametr musi być zmienną.

STORE

```
STORE adres,ile,bajt
STORE 1536,256,24
STORE A,B,C
```

Wypełnia obszar pamięci, począwszy od *adresu*, wartością określoną przez parametr *bajt*. Wielkość wypełnianego obszaru określa parametr *ile*. W powyższym przykładzie zostanie wypełniona cała 6. strona pamięci wartością 24.

Polecenia ekranowe

CLS

```
CLS
```

Czyści ekran w trybie GRAPHICS 0 lub okna tekstowe w innych trybach.

LINECOL

```
LINECOL numer_linii,kolor,jaskrawość
LINECOL 22,1,6
LINECOL 23,9,X
```

Ustala kolor tła w GRAPHICS 0 oddzielnie dla każdej linii obrazu (0 ÷ 23). *Kolor* i *jaskrawość* to liczby z przedziału 0 ÷ 15. Współpracuje z poleceniem MULTICOL.

MULTICOL

```
MULTICOL n
MULTICOL 1
```

Włącza (n=1) lub wyłącza (n=0) wielokolorowe tło w trybie GRAPHICS 0 (aktywuje 24 DLI i jedno VBI). Współpracuje z poleceniem LINECOL.

SCREEN

```
SCREEN n
SCREEN 0
```

Wyłącza (n=0) lub włącza (n=1) ekran. Wyłączenie ekranu przyspiesza pracę komputera o około 30%.

Grafika player – missile

Uwaga. Poniższe polecenia ułatwiają wykorzystanie grafiki PM. Należy pamiętać, aby pierwszym poleceniem było PMGRAPHICS, aktywujące PMG. Playery (gracze) mają oznaczenia 0 ÷ 3 a missile (pociski) 4 ÷ 7.

PMGRAPHICS

```
PMGRAPHICS n
PMGRAPHICS 2
```

Włącza lub wyłącza grafikę player – missile:
n=0 – PMG wyłączona;
n=1 – rozdzielczość dwuliniowa (mniejsza);
n=2 – rozdzielczość jednoliniowa (większa).

PMCLR
```
PMCLR n
PMCLR 1
```
Czyści pamięć przeznaczoną dla graczy (oddzielnie) i pocisków (łącznie).

PMCOLOR
```
PMCOLOR numer_gracza,kolor,jaskrawość
PMCOLOR 0,11,8
```
Ustala kolor graczy i pocisków, przy czym pociskom 4 ÷ 7 zostaną nadane kolory graczy odpowiednio 0 ÷ 3.

PMWIDTH
```
PMWIDTH numer_gracza/pocisku,szerokość
PMWIDTH 7,2
```
Ustala szerokość gracza lub pocisku:
0 – pojedyncza;
1 – podwójna;
2 – poczwórna.

PMPOS
```
PMPOS numer_gracza/pocisku,n
PMPOS 0,80
```
Ustawia gracza lub pocisk na poziomej pozycji *n*.

PMDOWN
```
PMDOWN numer_gracza/pocisku,skok
PMDOWN 0,2
```
Przesuwa wybranego gracza lub pocisk w dół o określony *skok* – ilość linii (należy zwrócić uwagę na wybraną rozdzielczość pionową PMG). W przykładzie player nr 0 zostanie przesunięty o dwie linie.

PMUP
```
PMUP numer_gracza/pocisku,skok
PMUP 0,2
```
Przesuwa wybranego gracza lub pocisk w górę o określony *skok*. Patrz PMDOWN.

PMADR
```
PMADR(numer_gracza/pocisku)
A=PMADR(1)
```
Funkcja zwraca adres w pamięci graczy (oddzielnie) lub pocisków (łącznie).

PM
```
PM(numer_gracza/pocisku)
J=PM(3)
```
Funkcja pozwalająca ustalić, który z obiektów PMG jest aktywny i w jakiej rozdzielczości:
0 – nieaktywny;
1 – aktywny, podwójna rozdzielczość;
2 – aktywny, pojedyncza rozdzielczość.

Funkcje losowe

RAND
```
RAND(n)
? RAND(50)
```
Generuje liczbę pseudolosową z przedziału od 0 do *n*, a dokładniej [0, n).

WSL
```
WSL(n)
Z=WSL(0.25)
```
Wynikiem funkcji jest 0 albo 1. Parametr *n* określa prawdopodobieństwo wystąpienia zera lub jedynki. Im mniejsza wartość, tym częściej losowane jest zero i odwrotnie. Przy n=0.5 prawdopodobieństwo wylosowania zera lub jedynki zbliżone jest do 50%. Parametr *n* musi mieścić się w zakresie 0 ÷ 1.

Dżojstik

HSTICK
```
HSTICK(n)
A=HSTICK(0)
```
Funkcja podaje poziome wychylenie dżojstika (n=0 to dżojstik #1 a n=1 to dżojstik #2):
-1 – w lewo;
 0 – brak (neutralne);
+1 – w prawo.

VSTICK
```
VSTICK(n)
VERT=VSTICK(1)
```
Funkcja podaje pionowe wychylenie dżojstika (n=0 to dżojstik #1 a n=1 to dżojstik #2):
-1 – w górę (od siebie);
 0 – brak (neutralne);
+1 – w dół (do siebie).

Współpraca ze stacją dysków

Opisane tu polecenia są odpowiednikami równoważnych poleceń DOS-a.
Uwaga od autora niniejszego opracowania. Poniższe polecenia z wyjątkiem DIR i SHELTER działają nieprawidłowo (DOS 2.5, MyDOS 4.50T). Nie należy więc raczej ich używać.

DIR
```
DIR "Dn:nazwa_pliku"
DIR "D2:*.HSB"
DIR A$
DIR
```
Odczytuje katalog dysku i wyświetla żądane pozycje. DIR bez parametrów (ostatni przykład) wyświetli całą zawartość dysku umieszczonego w napędzie #1.

ERASE
```
ERASE "Dn:nazwa_pliku"
ERASE "D:TEST.COM"
ERASE A$
```
Kasuje plik o podanej nazwie.

LOCK
```
LOCK "Dn:nazwa_pliku"
LOCK "D:GRA.HSB"
LOCK "D3:*.*"
LOCK A$
```
Zabezpiecza wskazany plik przed skasowaniem lub zmianą nazwy. W drugim przykładzie zostaną zabezpieczone wszystkie pliki na dysku w stacji #3.

RENAME
```
RENAME "Dn:stara_nazwa_pliku,nowa_nazwa_pliku"
RENAME "D:GRA.BAS,GRA.HSB"
RENAME A$
```
Zmienia nazwę programu znajdującego się na dysku.

SHELTER
```
SHELTER"D:nazwa_pliku"
SHELTER"D:NOWAGRA.HSB"
```
Zabezpiecza program przed edycją i listowaniem, zapisując w taki sposób, że może być wczytany i uruchomiony wyłącznie poleceniem `RUN"D:nazwa_pliku"`. Jeśli to twój program pamiętaj o zachowaniu niezabezpieczonej kopii na wypadek konieczności wprowadzenia zmian lub poprawek.

UNLOCK
```
UNLOCK "Dn:nazwa_pliku"
UNLOCK "D:GRA.HSB"
UNLOCK "D3:*.*"
UNLOCK A$
```
Polecenie odwrotne do LOCK.

Polecenia dla rozszerzenia16k Bibomon

BIB
```
BIB
```
Powoduje przejście do monitora. Powrót do BASIC-a i kontynuacja przerwanego programu klawiszem Q.

UHR
```
UHR n
UHR 1
```
Ukazuje (n=1) lub ukrywa (n=0) linię z zegarem.