Stołeczny Ośrodek
Elektronicznej
Techniki Obliczeniowej

INFORMATYKA mikrokomputerowa

INSTRUKCJA OBSŁUGI
KOMPUTERA AMSTRAD CPC 6128
tom II

AMSTRAD

Warszawa 1986 r.

Książka zawiera tłumaczenie na język polski pierwszej części oryginalnej instrukcji obsługi komputera AMSTRAD CPC 6128. Podano w niej podstawowe zasady posługiwania się komputerem, zasady programowania w języku BASIC i podstawowe informacje o posługiwaniu się dyskami elastycznymi przy użyciu systemów operacyjnych AMSDOS i CP/M. Książka zawiera także pełny opis słów kluczowych stosowanej w CPC 6128 wersji języka BASIC. Z książki mogą korzystać osoby nie mające żadnego przygotowania w tej dziedzinie.

Tłumaczenie przygotowane przez firmę wysyłkową

POLANGLIA LTD

jedyne autoryzowane przedstwicielstwo firmy AMSTRAD na Polskę.

58 St. Mary's Road
London W5 5EX
Wielka Brytania

tel.0-0441-840 1715
tlx 946581

Książka zawiera tłumaczenie na język polski drugiej części oryginalnej instrukcji obsługi komputera AMSTRAD CPC 6128. Książka obejmuje następujące tematy: wprowadzenie do języka LOGO, zestawienie informacji dotyczących sprzętu CPC 6128, opis programu BANK MANAGER oraz omówienie podstaw techniki mikrokomputerowej i programowania w języku BASIC. Książkę zamyka dodatek zawierający słowniczek najczęściej używanych terminów z dziedziny techniki komputerowej.

Tłumaczenie przygotowane przez firmę wysyłkową

POLANGLIA LTD

jedyne autoryzowane przedstawicielstwo firmy AMSTRAD na Polskę.

58 St. Mary's Road
London W5 5EX
Wielka Brytania

tel. 0-0441-840 1715
tlx 946581

INSTRUKCJA OBSŁUGI
KOMPUTERA AMSTRAD CPC 6128
tom II

Spis treści

Druk ZRiWOB Warszawa ul. Królewska 27 Format A-5 Nakład 2000 egz
Zam. Nr 121 z 1987 III 27 Ark. 13,625 P-48

ROZDZIAŁ 6

WPROWADZENIE DO LOGO

Celem tego rozdziału jest zapoznanie z istotą języka LOGO oraz przedstawienie listy komend języka z przykładami. Nie jest to wyczerpujący podręcznik czy też pełna specyfikacja LOGO. W celu uzyskania pełniejszych informacji prosimy sięgnąć do innych pozycji książkowych, wydawanych przez AMSOFT oraz inne wydawnictwa.

Rozdział zawiera:

1) Ideę języka LOGO
2) Instrukcję uruchomienia LOGO
3) Wprowadzenie do grafiki żółwia
4) Wskazówki, jak pisać własne procedury
5) Wskazówki, jak je poprawiać.

Czym jest LOGO?

Język LOGO pomoże Ci zostać programistą, niezależnie od tego, czy zdarzyło Ci się już kiedykolwiek napisać własny program.

LOGO jest językiem programowania o ogromnych możliwościach a łatwość nauczenia się i stosowania go powoduje, że jego popularność stale rośnie.

Program w języku LOGO tworzy się z procedur. Sam język LOGO jest też zbiorem procedur, zwanych czynnościami elementarnymi, prymitywami (ang. primitives), które można używać w celu pisania własnych programów.

LOGO stworzone zostało w latach siedemdziesiątych przez grupę informatyków i pedagogów pod kierunkiem Seymoura Paperta. Wraz z LOGO powstała koncepcja grafiki żółwia, pozwalająca małym dzieciom programować i używać komputery.

Wprowadzenie żółwia powodowane było, jak powiada Papert, koniecznością dania młodemu uczniowi "obiektu, który pomaga w myśleniu; narzędzia pomocnego w nauce prowadzonej nowymi sposobami".

Żółw, przedstawiany na ekranie w postaci strzałki, może być [illegible] po ekranie przy użyciu kilku prostych komend

DR. LOGO

Dr. Logo jest pełną implementacją LOGO stworzoną specjalnie dla komputera AMSTRAD 6128 tak, aby maksymalnie ułatwić proces programowania. Bogate możliwości dźwiękowe 6128 są w pełni udostępnione dla programisty, a edycja programu ułatwiona jest dzięki wykorzystaniu klawiszy sterujących kursorem.

URUCHOMIENIE LOGO

Aby rozpocząć pracę z językiem LOGO, nalezy włożyć do stacji dysków roboczą kopię dyskietki pierwszej strony pakietu dyskietek systemowych a następnie wprowadzić z klawiatury polecenie

```
|CPM
```

Po zgłoszeniu się systemu znakiem gotowości

```
> A
```

należy wyjąć dyskietkę, a w jej miejsce włożyć dyskietkę, będącą kopią 3 strony pakietu systemowego. W celu rozpoczęcia pracy z LOGO należy wprowadzić polecenie

```
SUBMIT LOGO 3
```

Po kilku sekundach na ekranie pojawi się tekst "Welcome ..." oraz znak zapytania, sygnalizujący gotowość LOGO.

DR. LOGO i system CP/M 2.2

Na czwartej stronie pakietu dyskietek systemowych znajduje się wersja języka LOGO dostosowana do pracy z systemem operacyjnym CP/M 2.2, działającym na komputerach CPC 664 oraz CPC 464 z dodatkową stacją dysków. Używanie LOGO 2 na CPC 6128, wobec istnienia wersji specjalnie dostosowanej do tego modelu, nie jest wskazane.

Jednakże, jeżeli z jakiś przyczyn pragniemy użyć LOGO 2, należy wykorzystać roboczą kopię 4 strony pakietu dyskietek systemowych i wprowadzić poleeenie

```
|CPM
```

Po pojawieniu się na ekranie znaków

```
>A
```

należy wprowadzić polecenie

```
SUBMIT LOGO 2
```

a po kilku sekundach pojawi się tekst "Welcome ..." i symbol gotowości LOGO 2 - znak zapytania.

Pierwsze kroki ...

Znak ? wskazuje, że Dr. LOGO czeka na wprowadzenie dowolnych znaków z klawiatury. Wprowadź (używając małych liter):

```
fd 6Ø
```

... a ukaże się żółw (w postaci strzałki), który przemieści się następnie do przodu o 60 jednostek. Na swojej drodze żółw pozostawi ślad - linię. Ekran zostanie oczyszczony tak, że powstanie duży obszar przeznaczony na grafikę oraz małe pole tekstowe ze znakiem ? w pobliżu dołu ekranu.

Dr. LOGO często podejmie decyzję o zmianie proporcji tych obszarów tak, aby dialog z komputerem był możliwie najwygodniejszy.

Wprowadź:

```
rt 9Ø
```

- a żółw obróci się w prawo o 90 stopni.

Następnie wprowadź:

```
fd 6Ø
```

a powstanie nowa linia o tej samej co uprzednio długości, lecz prostopadła do poprzedniej.

Radzimy przeprowadzić kilka eksperymentów z prostymi instrukcjami fd, bk ("back" - do tyłu), rt i lt ("left" - w lewo) i obserwować, co dzieje się na ekranie.

Procedury Dr. LOGO

Procedura jest listę instrukcji mówiących Dr. LOGO jak wykonać postawione zadanie.

Swoje pierwsze procedury napiszesz najprawdopodobniej przy pomocy takich procedur, które są na stałe wpisane do interpretera Dr. LOGO, a nazywane są operacjami elementarnymi bądź prymitywnymi.

fd, bk, rt i lt są takimi właśnie prymitywami, gotowymi do użycia w każdym momencie. Te, i inne prymitywy, są cegiełkami z których budować można własne procedury.

Następną przydatną operacją elementarną jest komenda cs, która powoduje wyczyszczenie ekranu i umiejscowienie żółwia w pozycji wyjściowej.

Pisanie prostej procedury

Łatwo można pokazać, że jeżeli operacje

```
fd 6Ø rt 9Ø
```

zostaną powtórzone cztery razy, to powstanie kwadrat o boku równym 60 jednostkom.

Ten sam efekt można osiągnąć przez napisanie prostej formuły

```
repeat 4 [fd 6Ø rt 9Ø]
```

Warto w tym miejscu samemu sprawdzić działanie powyższego ciągu operacji.

Aby z powyższego ciągu utworzyć procedurę o nazwie "square" (ang. "kwadrat"), wprowadź:

```
to square
repeat 4 [fd 6Ø rt 9Ø]
end
```

Od tej chwili Dr. LOGO rozumie pojęcie "square" i za każdym napotkaniem tego słowa narysuje na ekranie kwadrat. Napisanej przez nas procedurze można było oczywiście nadać inną nazwę, ale słowo "kwadrat" dobrze oddaje jej treść.

Dr. LOGO dopuszcza pisanie ciągu komend, a zatem polecenie

```
square rt 45 square
```

spowoduje narysowanie figury, na którą złożą się dwa kwadraty obrócone względem siebie o 45 stopni.

Procedury z parametrami

Możliwe jest zbudowanie procedury, której "ile razy" czy też "ile stopni" podawać będziemy w ten sam sposób, jak czynimy to używając operacji elementarnych. Aby napisać procedurę rysującą kwadraty o różnych bokach, definicja "square" może być zmieniona do postaci:

```
to squareanysize :side
repeat 4[ fd :side rt 45]
end
```

Zauważmy, że słowo "side" poprzedzone jest dwukropkiem. Wskazuje to, że :side nie jest komendą lecz zmienną.

Aby użyć zdefiniowaną procedurę, zmiennej :side musimy nadać wartość. Użycie komendy squareanysize 150 spowoduje narysowanie kwadratu o boku równym 150 jednostkom.

Spróbujmy połączyć procedury i zaobserwować efekty. Dla przykładu na podstawie ciągu

```
cs squareanysize 100 rt 45 squareanysize 150
```

żółw narysuje dwa kwadraty o różnych bokach, obrócone o 45 stopni.

Zwróćmy uwagę, że Dr. LOGO przy pomocy wykrzyknika (!) przypomni, że wprowadzona komenda nie mieści się w jednej linii ekranu i zostaje przeniesiona do linii następnej.

Używanie zmiennych do pamiętania wartości

Dr. LOGO dopuszcza również używanie zmiennych do pamiętanie różnych wartości i przekazywanie wartości do procedury.

Wpierw zdefiniujmy nową procedurę o nazwie triangle (ang. "trójkąt")

```
to triangle
repeat 3[ fd :edge rt 12Ø]
end
```

Możemy ją przetestować przez wprowadzenie

```
make "edge 1ØØ
triangle
```

Jeżeli chcemy dowiedzieć się, jaka wartość pamiętana jest przez zmienną :edge wystarczy wprowadzić :edge po znaku gotowości ? a Dr. LOGO poda nam odpowiedź.

W końcu, możemy użyć naszej zmiennej :edge w nowej procedurze rysującej pewien wzór. Zwróćmy uwagę, jak wartość zmiennej :edge jest zwiększana w czasie wykonywania procedury. Efektem tego zwiększania jest wzrost wielkości rysowanej figury

```
to pattern
triangle lt 6Ø triangle rt 6Ø
make "edge :edge + 4
pattern
end
make "edge 1Ø
cs pattern
```

Kiedy uznasz już, że należy przerwać program, uczyń to naciskając klawisz [ESC].

Edycja programów i procedur

Dr. LOGO pozwala na poprawianie błędów wprowadzania i zmienianie zdefiniowanych procedur. Klawiszami służącymi do tego celu są:

- klawisze sterowania kursorem: ↑ ↓ ← →, przesuwające kursor o literę bądź linię
- klawisze sterowania kursorem (↑ ↓ ← →) naciskane jednocześnie z klawiszem [CONTROL], co przesuwa kursor na końce linijki tekstu i o stronę w górę i w dół
- klawisz [DEL] zamazujący znak na lewo od kursora i klawisz [CLR] - zamazujący znak pod kursorem

- klawisz [RETURN], który sygnalizuje koniec linijki wprowadzanego tekstu
- klawisz [ESC] - oznaczający tyle, co "zaniechaj" i klawisz [COPY], którego wciśnięcie sygnalizuje koniec edycji procedury.

Jeżeli wprowadzamy komendy lub piszemy nową procedurę, edycja sprowadza się do właściwego ustawienia kursora na ekranie. Nowe znaki zostaną wpisane w miejsce wskazywane przez kursor.

Aby dokonać poprawek w tekście uprzednio zdefiniowanej procedury należy użyć komendy ed. Na ekranie ukaże się stara wersja tekstu - od tego momentu możemy już poprawiać ją tak jak zwykły tekst.

Spróbujmy dokonać edycji procedury "pattern".

Wprowadź

```
ed "pattern
```

Użyj klawiszy edycyjnych. Kiedy skończysz, wciśnij [ESC]. Dr. LOGO zignoruje to, co jest aktualnie na ekranie i wróci do starej wersji procedury.

Wprowadź

```
ed "pattern
```

powtórnie, zmieniając 4 na 8, a następnie naciśnij [COPY]. Następnie wykonaj procedurę obserwując, jak zmienił się wzór na ekranie. Nie zapomnij ustawić wartości początkowej zmiennej :edge.

Niektóre szczegóły działania LOGO

Bufor programu Dr. LOGO podzielony jest na obszary zwane węzłami. Aby dowiedzieć się, ile węzłów jest jeszcze wolnych wystarczy wprowadzić komendę

```
nodes
```

Od czasu do czasu, kiedy niemal wszystkie pola są już zajęte, Dr. LOGO przystępuje do porządkowania bufora. Objawia

się to chwilowym wstrzymaniem działań przez żółwia. Porządkowanie bufora można wymusić komendą

recycle

Pozwala to często na kontynuowanie pracy z LOGO pomimo sygnalizacji braku wolnych węzłów.

Jeżeli używasz wersji LOGO dostosowanej do systemu operacyjnego CP/M 2.2 (znajdującego się na 4 stronie pakietu dyskietek systemowych) - upewnij się, że na dyskietce jest wystarczająco dużo wolnego miejsca. Jest to ważne, o ile pragniesz zapisać swoje procedury na dyskietce. O ilości wolnego miejsca można przekonać się, używając komendy CAT systemu operacyjnego AMSDOS (patrz 7 część rozdziału Kurs podstawowy).

A teraz pozostaje tylko przebrnąć przez następne sekcje opisu LOGO - i nie zrażać się! Nie od razu wszystko będzie jasne. Z czasem używać będziesz coraz większej ilości komend.

Gdy już skończysz pracę z LOGO wprowadź

bye

Zestawienie komend języka LOGO

Następująca część rozdziału zawiera opis elementarnych operacji języka LOGO uszeregowanych tematycznie w porządku alfabetycznym wraz z opisem argumentów i licznymi przykładami.

Uwaga

Komendy wyróżnione gwiazdką nie są dostępne w wersji LOGO 2, pracującej pod kontrolą CP/M 2.2. Co za tym idzie, komendy te nie są wykonywane przez komputery CPC 664 i CPC 464 + DDI 1.

Przetwarzanie słów i struktur listowych

(zwróćmy uwagę na występujące w przykładach znaki gotowości ? oraz >)

ascii

Wyprowadza kod ASCII pierwszej litery wprowadzonego słowa

```
? ascii "G
71
? ascii "g
103
```

bf

(but first). Wyprowadza wszystkie, za wyjątkiem pierwszego, elementy wprowadzonego obiektu

```
? bf "smiles
miles
? bf [1 2 3]
[2 3]
```

bl

(but last). Wyprowadza wszystkie, za wyjątkiem ostatniego, elementy wprowadzonego obiektu

```
? bl "smiles
smile
? bl [1 2 3 4]
[1 2 3]
```

char

(znak). Wyprowadza znak, którego kod ASCII został wprowadzony

```
? char 83
S
```

count

Wyprowadza liczbę elementów wprowadzonego obiektu

```
? count "six
3
? count [0 1 2 3]
4
```

emptyp

Wyprowadza wartość logiczną TRUE (prawda), jeżeli wprowadzony obiekt był obiektem pustym; w przeciwnym razie wyprowadzona zostaje wartość logiczna FALSE (fałsz)

```
? emptyp "
TRUE
? emptyp []
TRUE
? emptyp [x]
FALSE
? make "x[]
? emptyp :x
TRUE
```

first

Wyprowadza pierwszy element wprowadzonego obiektu, jeżeli obiekt ten jest listą, to usunięte zostają nawiasy

```
? first "zebra
Z
? first [1 2 3]
1
```

fput

(first put). Dołącza pierwszy wprowadzony obiekt do drugiego obiektu w ten sposób, że pierwszy wprowadzony obiekt staje się pierwszym elementem drugiego obiektu

```
? fput "s "miles
smiles
? fput 1 [2 3]
[1 2 3]
```

item

Wyprowadza wskazany element wprowadzonego obiektu

```
? item 4 "dwarf
r
```

* last

Wyprowadza ostatni element wprowadzonego obiektu (porównaj z first)

```
? last "skyline
e
```

* lc

Wyprowadza całe podane słowo małymi literami

```
? lc "SOUTH
south
```

list

Tworzy listę z wprowadzonych obiektów, utrzymując nawiasy zamykające listy

```
? (list 1 2 3 4)
[1 2 3 4]
? list "big [feet]
[big [feet]]
? (list)
[]
```

* listp

Wyprowadza TRUE, jeżeli podany obiekt jest listą; w przeciwnym wypadku FALSE

```
? listp "mother
FALSE
? listp [father mother sister]
TRUE
```

* lput

(last put). Dołącza pierwszy wprowadzony obiekt do drugiego obiektu tak, że ten pierwszy staje się ostatnim elementem drugiego

```
? lput "s "plural
plurals
? lput "s [plural]
[plural s]
```

*** memberp**

Wyprowadza TRUE, jeżeli pierwszy obiekt jest elementem drugiego obiektu

```
? memberp "y "only
TRUE
? memberp "chocolate [[vanilla] [chocolate] [strawberry]]
FALSE
? memberp [chocolate] [[vanilla] [chocolate] [strawberry]]
TRUE
```

*** numberp**

Wyprowadza TRUE, jeżeli podany obiekt jest liczbą

```
? numberp 374.926
TRUE
? numberp "six
FALSE
? numberp first [2 4 6 8]
TRUE
```

*** piece**

Wyprowadza obiekt, który zawiera wskazane elementy wprowadzonego obiektu

```
? piece 4 7 "Kensington
sing
? piece 2 4 [Nana John Michael Wendy Tinkerbell]
[John Michael Wendy]
```

se

(sentence). Wyprowadza listę utworzoną z podanych obiektów, eliminując nawiasy zamykające (porównaj: list)

```
? make "instr_list rl
repeat 4[ fd 50 rt 90]
? run (se "cs  instr_list "ht)
```

Uwaga. Znak podkreślenia pomiędzy instr oraz list) uzyskuje się naciskając [SHIFT] Ø

* shuffle

Wyprowadza listę, na którą składają się losowo ustawione elementy wprowadzonej listy

```
? shuffle [a b c d]
[c b d a]
```

* uo

Wyprowadza podane słowo używając dużych liter

```
? uo "jones
JONES
```

(porównaj z lc)

* where

Wyprowadza liczbę obliczoną przy okazji ostatniej komendy memberp, która dała wartość TRUE

```
? memberp "v "river
TRUE
? show where
3
```

word

Wyprowadza słowo utworzone z wprowadzonych słów

```
? word "sun "shine
sunshine
```

wordp

Wyprowadza TRUE, jeżeli wprowadzony obiekt jest słowem lub liczą

```
? wordp "hello
TRUE
? wordp []
FALSE
```

Operacje arytmetyczne

* arctan

Wynikiem operacji jest wartość funkcji arcus tangens dla podanego argumentu (w stopniach)

```
? arctan Ø
Ø
? arctan 1
45
```

cos

Wynikiem operacji jest wartość funkcji cosinus dla podanego argumentu (w stopniach)

```
? cos 60
Ø.5
```

int

Wynikiem operacji jest część całkowita argumentu

```
? int 4/3
1
```

* quotient

Wynikiem operacji jest część całkowita z dzielenia argumentów

```
? quotient 14 4
3
? 14/4
3.5
```

random

Wynikiem jest nieujemna, losowa liczba całkowita nie większa niż podany argument

```
? random 20
7
```

* remainder

Wynikiem jest reszta z dzielenia pierwszego argumentu przez drugi

```
? remainder 7 3
1
? remainder 8 4
Ø
```

* rerandom

Wykonanie komendy powoduje powtórzenie poprzedniej sekwencji liczb losowych

```
? repeat 10[(type random 10 char 9)]
     1 3 7 5 3 2 0 4 2 6
? repeat 10[(type random 10 char 9)]
     4 9 9 1 0 6 1 3 5 1
? rerandom
? repeat 10[(type random 10 char 9)]
     5 2 9 0 3 1 6 2 3 7
? repeat 10[(type random 10 char 9)]
     5 2 9 0 3 1 6 2 3 7
```

* round

Zaokrągla podaną liczbę do najbliższej liczby całkowitej

```
? round 3.3333
3
? round 3 5
4
```

sin

Wynikiem operacji jest wartość funkcji sinus dla podanego w stopniach argumentu

```
? sin 30
Ø.5
```

+

Suma argumentów

```
? + 22
4
? 2 + 2
4
```

-

Różnica argumentów

```
? - 10 5
5
? 10 - 5
5
```

* Iloczyn argumentów

```
?  4 6
24
? 4 6
24
```

/

Iloraz argumentów

```
? / 25 5
5
? 25/5
5
```

Operacje logiczne

and

Wyprowadza wartość logiczną TRUE (prawda) jeżeli wszystkie wyrażenia logiczne, będące argumentami są prawdziwe

```
? and (3<4) (7>4)
TRUE
```

not

Wyprowadza TRUE, jeżeli wyrażenie logiczne, będące argumentem jest fałszywe. Wyprowadza FALSE (fałsz) jeżeli wyrażenie logiczne, będące argumentem jest prawdziwe.

```
? not (3 = 4)
TRUE
? not (3 = 3)
FALSE
```

or

Wyprowadza FALSE, jeżeli wszystkie wyrażenia logiczne, będące argumentami są fałszywe

```
? or "TRUE "FALSE
TRUE
? or (3 = 4) (1 = 2)
FALSE
```

=

Wyprowadza TRUE, jeżeli oba podane obiekty są identyczne (równe); w przeciwnym razie FALSE

```
? = "LOGO "LOGO
TRUE
? 1 = 2
FALSE
```

>

Wyprowadza TRUE, jeżeli pierwsze wprowadzone słowo jest większe niż drugie; w przeciwnym razie FALSE

```
? > 19 20
FALSE
? 20 > 19
TRUE
```

<

Wyprowadza TRUE, jeżeli pierwsze wprowadzone słowo jest mniejsze niż drugie; w przeciwnym razie FALSE

```
? < 27 13
FALSE
? 13<27
TRUE
```

Zmienne

local

Nadaje podanym zmiennym charakter lokalny; są one dostępne tylko wewnątrz procedury oraz w procedurach wywoływanych

```
>(local "x "y "z)
```

make

Nadaje wartość zmiennej określonej przez nazwę

```
? make "side 50
? :side
50
```

* namep

Wyprowadzona zostaje wartość logiczna TRUE, jeżeli podane słowo jest indentyfikatorem zmiennej

```
? make "flavour "chocolate
?  :flavour
chocolate
? namep "flavour
TRUE
? namep "chocolate
FALSE
```

* thing

Wyprowadzona zostaje wartość przyporządkowana podanej zmiennej

```
? make "computer "amstrad
? thing "computer
amstrad
```

Procedury

* define

Nadaje nazwie procedury treść listy będącej argumentem

```
? define "say.hello [[] [ pr "hello]]
? po "say hello
to say hello
pr "hello
? text "say hello
[[] [pr "hello]]
end
```

end

Wskazuje na koniec definicji procedury; słowo "end" winno występować samodzielnie na początku ostatniej linii definicji

```
? to square
> repeat 4 [fd 50 rt 90]
> end
square defined
? square
```

po

("print out"). Wyświetla definicję podanej procedury lub zmiennej

```
? po "square
to square
repeat 4 [ fd 50 rt 90]
end
? make "x 3
? po "x
x is 3
```

pots

("print out titles"). Wyświetla nazwy i nagłówki wszystkich procedur w obszarze roboczym pamięci

```
? pots
```

***text**

Wyprowadza ciąg definiujący podaną procedurę

```
? to star; gwiazda pięcioramienna
> repeat 5[ fd 30 rt 144 fd 30 lt 72]
> end
star defined
? text "star
[[][repeat 5[ fd 30 rt 144 fd 30 lt 72]]]
```

to

Wskazuje na początek definicji procedury

```
? to square
> repeat 4[ fd 50 rt 90]
> end
square defined
```

Edycja programu

ed

("edit"). Powoduje załadowanie podanych procedur i (lub) zmiennych do bufora edytora ekranowego

```
? ed "square
```

*** edall**

Powoduje załadowanie wszystkich zmiennych i procedur z obszaru roboczego do bufora edytora ekranowego i wywołuje edytor

```
? edall
```

*** edf**

Powoduje załadowanie podanego zbioru dyskowego do bufora edytora ekranowego bezpośrednio z dysku; jeżeli zbiór nie istnieje - jest on tworzony, a wywołanie edytora następuje z pustym buforem

```
? edf "star
```

Sterowanie drukarką

* copyon

Powoduje rozpoczęcie powielania tekstu na drukarce

```
? copyon
```

* copyoff

Powoduje zaprzestanie powielania tekstu na drukarce

```
? copyoff
```

Sterowanie tekstem na ekranie

ct

("clear text"). Powoduje oczyszczenie okienka tekstowego, które aktualnie zawiera kursor, a następnie umieszcza kursor w górnym lewym rogu okienka

```
? ct
```

* cursor

Wyprowadza listę współrzędnych kursora (kolumna, linia) w obszarze okienka tekstowego

```
? ct
? cursor
[ø 1]
? (type [The current cursor position is\] show cursor
The current cursor position is [32 32]
```

pr

("print"). Wyprowadza podane obiekty w okienku tekstowym, usuwa nawiasy ograniczające, uzupełnia ostatni wprowadzony obiekt znakiem CR (powrót karetki (porównać komendy show i type)

```
? pr [a b c]
a b c
```

*** setcursor**

Umieszcza kursor w miejscu podanym w liście współrzędnych w okienku tekstowym

```
? ct
? to picture
> make "x random 20
> make "y random 12
> setcursor list :x :y pr "*
> end
? picture
```

setsplit

Określa ilość linii w okienku

```
? setsplit 10
```

show

Wyprowadza podane obiekty w okienku tekstowym, utrzymuje nawiasy ograniczające, dodaje znak CR do ostatniego wprowadzonego obiektu (porównać komendy pr i type)

```
? show [a b c]
[a b c]
```

ts

("text screen"). Definiuje cały ekran jako okienko tekstowe

```
? ts
```

type

Wyprowadza podane obiekty w okienku tekstowym, usuwa nawiasy ograniczające, nie dodaje znaku CR do ostatniego obiektu (porównać komendy pr i show)

```
? type [a b c]
a b c
```

Sterowanie grafiką

W czasie pracy z systemem LOGO ekran jest w trybie 1, co pozwala na uzyskanie czterech kolorów. Współrzędne ekranu określane są tak samo, jak w czasie pracy z językiem BASIC. Innymi słowy współrzędne ekranu będą zaokrąglane do najbliższej liczby parzystej. Kolory czerwony, zielony i niebieski mogą mieć intensywności 0, 1 i 2.

clean

Kasuje zawartość okienka graficznego nie wpływając na symbol żółwia

```
? fd 5Ø
? clean
```

cs

("clear screen"). Kasuje zawartość okienka graficznego; ustawia żółwia w pozycji Ø,Ø i skierowuje go na północ (ku górze ekranu). Pióro żółwia zostaje opuszczone.

```
? rt 9Ø fd 5Ø
? cs
```

dot

Rozjaśnia punkt o podanych współrzędnych w aktualnie używanym kolorze

```
? dot [5Ø 1Ø]
```

* dotc

Wyprowadza numer koloru punktu o podanych współrzędnych. Jeżeli punkt jest poza obszarem okienka graficznego, wynikiem jest liczba -1

```
? cs
? setpc 1
? dot [-5Ø 5Ø]
? setpc 2
? dot [5Ø 5Ø]
? setpc 3
```

```
? dot [50 -50]
? dotc [50 50]
2
? dotc [-50 -50]
0
? dotc [1000 3000]
-1
```

fence

Tworzy granice limitujące ruchy żółwia do obszaru okienka graficznego. Komenda window likwiduje granice

```
? fence
? fd 300
Turtle out of bounds
```

co oznacza: "żółw poza obramowaniem"

fill

Pokrywa obszar na ekranie aktualnie używanym kolorem, zmieniając punkt pod żółwiem (oraz wszystkie pionowe i poziome ciągłe linie tego samego koloru) odpowiednio do aktualnego stanu pióra

```
? make "x 5
? cs
? st
? pd
? repeat 30[fd :x rt 90 make "x :"x + 5]
? fd 20 rt 90
? fd 10
? pu
? home
? bk 2
? pd
? setpc 2
? fill
```

fs

("full screen"). Definiuje cały ekran jako okienko graficzne

```
? fs
```

pal

("palette"). Wyprowadza liczby reprezentujące nasycenie przyporządkowanych do pióra kolorów: czerwonego, zielonego i niebieskiego

```
? pal 2
[Ø 2 2]
```

* setbg

Nadaje tłu ekranu kolor przyporządkowany wprowadzonej liczbie

```
? sf
[Ø SS 5 FENCE 1]
```

Powyższe pokazuje, że tłu przyporządkowany jest kolor Ø

```
? pal Ø
[Ø Ø 1]
? setbg 2
? sf
[2 SS 5 FENCE 1]
```

setpal

("set palette"). Nadaje podanemu pióru odpowiedni kolor, przyporządkowując mu nasycenie kolorami podstawowymi (czerwony, zielony, niebieski)

```
? setpal  3 [1 1 2]
? pal 3
[1 1 2]
```

* setscrunch

Ustala skalę ekranu w kierunku pionowym

```
? sf
[Ø SS 5 FENCE 1]
? to circle
> repeat 360[fd 1 rt 1]
> end
circle defined
? setscrunch 2
```

```
? sf
[Ø SS 5 FENCE 2]
? circle
? setscrunch 2
? circle
```

sf

("screen facts"). Komenda powoduje wyprowadzenie informacji dotyczących aktualnych parametrów okienka graficznego. Format odpowiedzi jest następujący: [<kolor tła> <stan ekranu> <rozmiar okienka tekstu> <ograniczenie ekranu> <skala>] ; <kolor tła> jest numerem pióra, z którego pochodzi tło ekranu; <stan ekranu> wskazuje SS (ekran dzielony na tekst i grafikę), FS (cały ekran graficzny) lub TS (cały ekran tekstowy); <rozmiar okienka tekstu> jest liczbą wskazującą, ile linijek tekstu mieści okienko tekstowe a <ograniczenie ekranu> pokazuje, który z trybów: WINDOW, WRAP czy FENCE jest używany. <skala> jest skalą ekranu - domyślnie równą 1. Skala może być zmieniana komendą setscrunch (tylko CP/M 3.0)

```
? sf
[Ø SS 5 FENCE 2.5]
```

ss

("split screen"). Tworzy okienko tekstowe na ekranie

```
? ss
```

window

Zezwala żółwiowi na kreślenie poza granicami ekranu graficznego po komendzie wrap lub fence

```
? fence fd 3ØØ
Turtle out of bounds
? window
? fd 3ØØ
```

wrap

Powoduje, że po przekroczeniu granicy ekranu żółw pojawia się po jego przeciwległej stronie

```
? cs wrap
? rt 5 fd 1ØØØ
? cs window
? rt 5 fd 1ØØØ
```

Grafika żółwia

bk

("back"). Przesuwa żółwia o podaną ilość kroków do tyłu, tj. przeciwnie do kierunku jego ustawienia

```
? cs fd 15Ø
? bk 5Ø
```

fd

("forward"). Przesuwa żółwia do przodu o podaną ilość kroków

```
? fd 8Ø
```

* home

Powoduje ustawienie żółwia w pozycji Ø Ø (na środku ekranu) i ustawia go strzałką do góry (na północ)

```
? fd 1ØØ
? rd 45
? fd 1ØØ
? home
```

ht

("hide turtle"). Czyni symbol żółwia niewidocznym - przyspiesza to rysowanie oraz powoduje, że obraz staje się bardziej przejrzysty

```
? ht
? cs fd 5Ø
? st
```

lt

("left"). Obraca żółwia w lewo (w stosunku do jego aktualnego położenia) o kąt podany w stopniach

```
? lt 9Ø
```

pd

("pen down"). Opuszcza pióro żółwia - który od chwili wykonania tej komendy kreśli linie na ekranie

```
? fd 2Ø pu fd 2Ø
? pd
? fd 2Ø
```

pe

("pen erase"). Zmienia kolor pióra żółwia na kolor tła ekranu; żółw ma możliwość wymazywania linii

```
? fd 5Ø
? pe
? bk 25
? fd 5Ø
? pd fd 25
```

pu

("pen up"). Podnosi pióro żółwia; żółw przestaje kreślić po ekranie

```
? fd 3Ø
? pu
? fd 3Ø
? pd fd 3Ø
```

px

("pen exchange"). Powoduje, że kolor punktów na przebytej przez żółwia drodze zostaje zamieniony na kolor będący logicznym dopełnieniem poprzedniego

```
? fd 2Ø pu fd 2Ø
? pd setpc 3 fd 2Ø
? px
? bk 8Ø
? fd 8Ø
? pd bk 1ØØ
```

rt

Obraca żółwia w prawo o kąt podany w stopniach

? rt 90

seth

(ang. "set heading"). Ustawia żółwia pod zadanym kątem podanym w stopniach. Jeżeli kąt jest dodatni, obrót następuje zgodnie z kierunkiem ruchu wskazówek zegara, w przeciwnym razie kąt odliczany jest przeciwnie do kierunku ruchu wskazówek

? seth 90

setpc

(ang. "set pen colour"). Wprowadzona liczba określa wybór pióra żółwia

? setpc 1

setpos

(ang. "set position"). Przesuwa żółwia na pozycję podaną jako argument komendy

? setpos [30 20]

*setx

Przesuwa żółwia na pozycję o współrzędnej x podanej jako argument komendy. (Zobacz również sety)

? setx 80
? fd 100
? setx -50
? fd 50

* sety

Przesuwa żółwia na pozycję o współrzędnej y podanej jako argument komendy. (Zobacz również setx)

? sety 90
? fd 20
? sety -50
? fd 50

st

(ang. "show turtle0). Przywraca na ekranie symbol żółwia ukryty komendą ht

```
? ht
? fd 5Ø
? st
```

tf

(ang. "turtle facts"). Powoduje wyprowadzenie na ekran informacji o żółwiu. Format odpowiedzi jest następujący: [<xcor> <ycor> <heading> <penstate> <pencolour n> <shownp>] - gdzie <xcor> jest aktualną współrzędną x żółwia; <ycor> - jego aktualną współrzędną y; <heading> - jest określonym w stopniach kierunkiem, w którym zwrócony jest żółw; <shownp> przyjmuje wartość logiczną TRUE, jeżeli żółw jest widoczny; <penstate> może wskazywać następujące stany: PD - pióro opuszczone, PE - pióro o kolorze tła, zamazujące, PX - pióro zmieniające kolor każdego napotkanego, uprzednio postawionego punktu, PU - pióro podniesione <pencolour n> wskazuje na numer aktualnie używanego pióra

```
? setpos  15 3Ø
? rt 6Ø
? setpc 3
? pe
? ht
? tf
[15 3Ø 6Ø PE 3 FALSE]
```

*** towards**

Wyprowadza nagłówek, który powoduje ustawienie żółwia w kierunku podanym jako argumenty komendy

```
? seth towards list :x :y
```

Zarządzanie buforem programu

er

(ang. "erase"). Usuwa podaną procedurę (procedury) z bufora

```
? er "square
```

* erall

Usuwa wszystkie procedury i zmienne z bufora

```
? erall
```

ern

(ang. "erase name"). Usuwa podane zmienne z bufora

```
? make "side [1ØØ]
? make "angle [45]
? :side :angle
[1ØØ]
[45]
? ern [side angle]
? :side
side has no value
```

nodes

Wyprowadza liczbę wolnych węzłów bufora zmiennych i procedur

```
? nodes
```

* nonformat

Usuwa formatowanie procedur wraz z komentarzami w celu uzyskania wolnych pozycji bufora na nowe obiekty

```
? noformat
```

* poall

Wyprowadza wszystkie znajdujące się w buforze definicje procedur i zmiennych

```
? poall
```

*** pons**

Wyprowadza nazwy i wartości wszystkich zmiennych globalnych, jakie znajdują się w buforze

```
? pons
medium is 4Ø
small is 2Ø
large is 8Ø
```

*** pops**

Wyprowadza nazwy i definicje wszystkich procedur, jakie znajdują się w buforze

```
? pops
```

recycle

Zwalnia możliwie największą liczbę węzłów w buforze i reorganizuje go

```
? recycle
? nodes
```

Listy atrybutów

glist

(ang. "get list"). Wyprowadza listę wszystkich obiektów w buforze, które w swojej liście atrybutów mają podany w komendzie atrybut

```
? glist ".DEF
```

gprop

(ang. "get property"). Wyprowadza wartość podanego atrybutu obiektu wyspecyfikowanego w komendzie

```
? make "heıgnt "72
? gprop "height ".APV
72
```

plist

(ang. "property list"). Wyprowadza listę atrybutów podanego obiektu

```
? plist "height
[.APV 72]
```

pprop

(ang. "put property"). Wprowadza podaną parę atrybut-wartość na listę atrybutów wyspecyfikowanego w komendzie obiektu

```
? pprop "master ".APV"Scott
? :master
Scott
```

*pps

Wyprowadza niestandardowe pary atrybut-wartość wszystkich obiektów w buforze

```
? pprop "Sally "extension 213
? pps
Sally's extension is 213
? plist "Sally
[extension 213]
```

remprop

(ang. "remove property"). Usuwa podany atrybut z listy atrybutów wyspecyfikowanego w komendzie obiektu

```
? remprop "master ".APV
```

Zbiory Dyskowe

*changef

Zmienia nazwę zbioru w katalogu dysku

```
? dir
[SQUARE CIRCLE STARS]
? changef "boxes "square
? dir
[BOXES CIRCLE STARS]
```

***defaultd**

Wyprowadza nazwę używanego aktualnie napędu dysków

```
? defaultd
A:
```

dir

(ang. "directory"). Wyprowadza listę nazw zbiorów Dr. LOGO znajdujących się na aktualnie używanym bądź podanym dysku; przyjmowane są symbole zastępcze dla znaków.

```
? dir "a: ????????
```

(Używanie symboli zastępczych ???????? opisane jest w części pierwszej rozdziału zatytułowanego "AMSDOS i CP/M". Należy zwrócić uwagę na to, iż symbol zastępczy * nie jest akceptowany przez Dr. LOGO)

*** dirpic**

Wyprowadza listę nazw zbiorów będących obrazami, znajdujących się na aktualnie używanym bądź wyspecyfikowanym dysku. Komenda akceptuje symbole zastępcze

```
? dirpic
[MY_PIC SQUARES STARS NOTES]
```

load

Wczytuje zbiór o podanej nazwie z dysku do bufora programu

```
? load "myfile
? load "b:shapes
```

*** loadpic**

Odtwarza na ekranie obraz zapisany uprzednio na dysku jako zbiór

```
? load "my_pic
? load "b:my_pic
```

save

Zapisuje na dysk zawartość bufora programu do zbioru o podanej nazwie

? sace "shapes

UWAGA! Przed wykonaniem komendy należy do napędu włożyć zformatowany dysk, na którym jest wystarczająca ilość wolnego miejsca. Nigdy nie należy zapisywać zbiorów na oryginalnej dyskietce systemowej. Ryzyko przypadkowego zapisu na te dyskietki powinno być wyeliminowane poprzez zasłonięcie otworu zezwalającego na zapis.

Jeżeli używa się Dr. LOGO w wersji pracującej z systemem operacyjnym CP/M 2.2 nie należy zmieniać dysku w ciągu całej pracy z systemem. Ważne jest zatem, aby robocza wersja dyskietki z systemem operacyjnym i Dr. LOGO miała dużo wolnego miejsca na programy.

* savepic

Zapisuje zawartość ekranu graficznego do zbioru dyskowego o podanej nazwie

? savepic "my pic
? savepic "b:my pic

UWAGA: Przed wykonaniem komendy należy do napędu włożyć zformatowany dysk, na którym jest wystarczająca ilość wolnego miejsca. Nigdy nie należy zapisywać zbiorów na oryginalnej dyskietce systemowej. Ryzyko przypadkowego zapisu na te dyskietki powinno być wyeliminowane poprzez zasłonięcie otworu zezwalającego na zapis.

Jeżeli używa się Dr. LOGO w wersji pracującej z systemem operacyjnym CP/M 2.2 nie należy zmieniać dysku w ciągu całej pracy z systemem. Ważne jest zatem, aby robocza wersja dyskietki z systemem operacyjnym i Dr. LOGO miała dużo wolnego miejsca na programy.

* setd

(ang. "set drive"). Zmienia numer napędu dyskowego współpracującego z systemem

```
? defaultd
A:
? dir
[BOXES CIRCLE STARS]
? setd "b:
? defaultd
B:
? dir
[TRIANGLE HOUSE]
```

Klawiatura i joystick

buttonp

(ang. "button pressed"). Wyprowadza wartość logiczną TRUE, jeżeli przycisk podanego joysticka jest naciśnięty; numery 0 i 1 rozróżniają dwa możliwe do użycia joysticki

```
? to fire
> label "loop
> if (buttonp 0) [pr [fire 0!]]
> if (buttonp 1) [pr [fire 1!]]
> go "loop
> end
```

Pozycja joysticka testowana jest komendą p a d d l e

keyp

Wyprowadza wartość logiczną TRUE jeżeli znak z klawiatury gotowy jest do odczytania przez system

```
? to inkey
> if keyp [op rc] [op "]
> end
```

paddle

Wynikiem jest położenie podanego joysticka. Każdemu położeniu przyporządkowana jest odpowiednia liczba jak następuje:

Wartość	Znaczenie
255	położenie środkowe
0	do góry
1	do góry i w prawo
2	w prawo
3	w dół i w prawo
4	w dół
5	w dół i w lewo
6	w lewo
7	w górę i w lewo

```
? paddle Ø
255
```

Przyciski joysticków testowane są komendą b u t t o n p

rc

("read character"). Wyprowadza pierwszy wprowadzony z klawiatury znak

```
? make "key rc
... naciśnij teraz klawisz X ...
? :key
X
```

rl

("read list"). Wyprowadza listę, która zawiera ciąg znaków z klawiatury; ciąg wejściowy winien być zakończony symbolem CR

```
? make "instr_list rl
repeat 4[ fd 5Ø rt 9Ø]
? :instr_list
 repeat 4[ fd 5Ø rt 9Ø]
```

rq

("read quote"). Wyprowadza słowo, które zawiera ciąg znaków z klawiatury zakończony symbolem CR

```
? make "command rq
repeat 3[ fd 6Ø rt 12Ø]
?  command
repeat 3 [fd 6Ø rt 12Ø]
```

Dźwięk

Komendy sterujące generowaniem dźwięku dotyczą wyłącznie implementacji Dr. LOGO na komputerze AMSTRAD i są analogiczne do odpowiednich komend języka BASIC.

Szczegółowe informacje można znaleźć w cz. 9 rozdziału "Kurs podstawowy".

sound

Komenda wprowadza dźwięk do bufora dźwięków

Format: [<status kanału> <okres tonu> < czas trwania> < głośność> < obwiednia głośności> < obwiednia tonu > < szum>]. Wszystkie parametry po < czas trwania > są opcjonalne

```
? sound [1 2Ø 5Ø]
```

env

Komenda definiuje obwiednię głośności. Format:
[<numer obwiedni> <sekcje obwiedni>]

```
? env [1 1ØØ 2 2Ø]
? sound [1 2ØØ 3ØØ 5 1]
```

ent

Komenda definiuje obwiednię tonu. Format:
[<numer obwiedni> < sekcje obwiedni>]

```
? ent [1 1ØØ 2 2Ø]
? sound [1 2ØØ 3ØØ 5 1 1]
```

release

Uwalnia kanały dźwiękowe wprowadzone w stan wstrzymania komendą s o u n d. Kanały, które należy uwolnić wskazywane są za pomocą argumentu komendy zgodnie z następującą tabelką:

argument	kanały
0	żaden
1	A
2	B
3	

```
5          A 1 C
6          B 1 C
7          A 1 B 1 C

? release 1
```

Komendy sterujące

bye

Kończy sesję pracy z Dr. LOGO

```
? bye
```

co

Przerywa pauzę wprowadzoną przez naciśnięcie [CONTROL] Z, komendę p a u s e lub wartość zmiennej systemowej ERRACT

```
? co
```

go

Powoduje wykonanie jako następnej tej spośród linii wewnątrz procedury, która po wyrażeniu l a b e l ma taki sam wyraz jak argument komendy g o

```
> go "loop
```

if

Powoduje wykonanie jednej z dwu list instrukcji w zależności od wartośoi wyrażenia stojącego za słowem if; instrukcje muszą być listami składającymi się z liter, zamkniętymi nawiasami

```
> if (a > b) [pr [a is bigger]]
>[pr [b is bigger]]
```

label

Identyfikuje linię, która ma być wykonana po komendzie g o z podanym wyrazem

```
> label "loop
```

op

("output"). Wskazuje obiekt, przez który ma być przekazany wynik działania procedury i powoduje jej opuszczenie

```
? op [result]
```

repeat

Powoduje wykonanie ciągu instrukcji zawartych w podanej liście tyle razy, ile wskazuje to pierwszy argument komendy

```
? repeat 4[ fd 5Ø rt 9Ø]
```

run

Komenda powoduje wykonanie podanej listy instrukcji

```
? make "instr_list [fd 4Ø rt 9Ø]
? run : instr_list
```

stop

Powoduje przerwanie działania procedury i powrót do procedury wołającej lub TOPLEVEL (poziom nadrzędny, sygnalizowany znakiem gotowości ?)

```
? stop
```

wait

Przerywa działanie procedury na czas wynikający z argumentu komendy. Czas obliczany jest na podstawie zależności: czas = argument * 1/60 sek.

```
? wait 2ØØ
```

Obsługa sytuacji wyjątkowych

catch

Powoduje ustawienie pułapek programowych w wypadku wystąpienia błędów i pewnych specjalnych warunków, które mogą pojawić się w czasie wykonywania podanej listy komend

```
catch "error [ ] [ ]
```

error

Wyprowadza listę, której elementy opisują ostatni błąd jaki wystąpił w czasie realizacji programu

```
> catch "error [do. until. error]
> show error
```

* notrace

Wyłącza śledzenie wykonywania procedury (patrz t r a c e)

```
? notrace
```

* nowatch

Wyłącza podgląd wszystkich bądź podanych procedur (patrz w a t c h)

```
? nowatch
```

pause

Zawiesza wykonanie procedury w celu ingerencji interpretera bądź edytora

```
> if :size > 5 [pause]
```

throw

Powoduje wykonanie linii określonej przez nazwę wprowadzoną przy poprzednim wyrażeniu c a t c h

```
? throw "TOPLEVEL
```

* trace

Włącza śledzenie wykonania procedury

```
? trace
```

* watch

Włącza podgląd wszystkich bądź podanych procedur

```
? watch
```

Prymitywy systemowe

.contents
Wyświetla zawartość tablicy symboli Dr. LOGO

.deposit
Wpisuje liczbę będącą drugim argumentem do komórki pamięci komputera określonej przez adres będący pierwszym argumentem

.examine
Wyświetla zawartość podanej komórki pamięci komputera

* **.in**
Pobiera aktualną zawartość portu o podanym numerze

* **.out**
Wysyła podaną wartość do portu o wskazanym numerze

Zmienne systemowe

ERRACT
Jeżeli ma wartość TRUE - powoduje wstrzymanie przy wystąpieniu błędu i powrót na poziom TOPLEVEL

FALSE
Zmienna systemowa

REDEFP
Wartość TRUE zezwala na redefiniowanie prymitywów

TOPLEVEL
Komenda t r o w "TOPLEVEL powoduje natychmiastowe przerwanie wszystkich aktywnych procedur

TRUE
Zmienna systemowa

Atrybuty systemowe

.APV
("associated property value"). Wartość zmiennej globalnej

.DFF
Definicja procedury

* .FNL
Koniec linii definicji procedury, zakończonej znakiem CR lub spacjami

* .FMT
Początek linii definicji procedury, zakończonej znakiem CR i spacjami

.PRM
Znacznik prymitywu

* .REM (lub ;)
Uwaga bądź komentarz.

ROZDZIAŁ 7
NIECO UŻYTECZNYCH INFORMACJI ...

Rozdział ten zawiera wiele użytecznych informacji, po trzebnych Ci przy nauce posługiwania się komputerem.

A oto omawiane zagadnienia:

- Lokacje kursora i kody kontrolne (sterujące)
- Przerwania
- Znaki ASCII i znaki graficzne
- Opis klawiszy
- Dźwięk
- Komunikaty o błędach
- Kluczowe słowa języka BASIC
- Szablony
- Złącza
- Drukarki
- Joysticki
- Organizacja dysku
- Rozszerzenie systemu rezydentnego (RSX)
- pamięć
- Emulator terminale w systemie CP/M Plus
- Zbiór znaków w systemie CP/M Plus

Szczegółowy opis języka BASIC i opis firmowy komputer CPC6128 znaleźć można w opracowaniach AMSOFT SOFT967 i SOF

Część 1: Lokacje kursora i kody kontrolne (sterujące) w systemie BASIC

W wielu programach aplikacyjnych kursor tekstowy może umieszczony poza aktualnie wykorzystywanym oknem ekranu ta że staje się niewidoczny. Różne operacje powodują jednak powtórne ulokowanie kursora w wyświetlanym polu ekranu i t jeszcze przed wykonaniem operacji, operacje takie to:

1. Wpisanie znaku
2. Narysowanie znaku kursora
3. Użycie jed[illegible] kodów kontrolnych, oznaczonych gwiazdk w umiesz[illegible] kodów.

Powrót kursora do wykorzystywanego okna ekranu następuje zgodnie z poniższymi regułami:

1. Jeżeli kursor znajduje się na prawo od prawego skraju ekranu, przesuwany jest do pierwszej, najbardziej na lewo wysuniętej kolumny następnej, niższej linii.

2. Jeżeli kursor znajduje się na lewo od lewego skraju ekranu, przesuwany jest do ostatniej, najbardziej na prawo wysuniętej kolumny poprzedniej, wyższej linii.

3. Jeżeli kursor znajduje się powyżej górnego skraju ekranu, treść wyświetlana na ekranie jest przesuwana w dół o jedną linię a kursor umieszczany jest w najwyższej linii.

4. Jeżeli kursor znajduje się poniżej dolnego skraju ekranu, treść wyświetlana na ekranie jest przesuwana w górę o jedną linię a kursor umieszczany jest w najniższej linii. Badanie położenia i przesuwanie kursora przeprowadzane jest przy tym w podanej wyżej kolejności. Pozycja kursora poza wyświetlanym oknem ekranu może być zerowa lub ujemna, co oznacza położenie na lewo lub powyżej okna.

Znaki o wartościach od 0 do 31, wysyłane do ekranu tekstowego nie powodują wyświetlenia znaku na ekranie lecz są interpretowane jako ZNAKI KONTROLNE i nie mogą być używane nierozważnie. Niektóre z tych znaków zmieniają znaczenie jednego lub kilku następujących po nich znaków, przyjmowanych jako parametry wysłanego znaku kontrolnego.

Znaki kontrolne, wysyłane do ekranu graficznego powodują jedynie wyświetlenie konwencjonalnego symbolu, związanego z ich funkcją, jeżeli wysyłane są za pomocą klawiatury (np. & 07 'BEL' - [CTRL] G). Znaki takie spełniają natomiast przypisane im funkcje kontrolne, gdy przekazywane są przez polecenia programowe:
PRINT CHR$ (&07), lub PRINT "␇" (gdzie ␇ uzyskuje się przez wprowadzenie [CTRL]G wewnątrz instrukcji PRINT)

Znaki kontrolne zestawione są w tabeli, gdzie podano ich wartość szesnastkową (&XX) i dziesiętną oraz opisano ich działanie w systemie BASIC.

Znaki oznaczone * powodują powrót kursora na odpowiednią pozycję w wyświetlanym oknie ekranu jeszcze przed wykonaniem przynależnej im operacji, chociaż mogą także pozostawić kursor w położeniu niewidocznym na ekranie.

Znaki kontrolne w systemie BASIC

Wartość		Nazwa	Parametr	Działanie
&00	0	NUL		Żadne; znak ignorowany
&01	1	SOH	0 do 255	Wyświetla symbole podane jako parametry; pozwala to na wyświetlanie symboli o wartości od 0 do 31
&02	2	STX		Wyłącza kursor tekstowy. Odpowiednik rozkazu CURSOR z parametrem ⟨przełącznik użytkownika⟩ równym 0
&03	3	ETX		Włącza kursor tekstowy. Odpowiednik rozkazu CURSOR z parametrem ⟨przełącznik użytkownika⟩ równym 1. Aby jednak wyświetlić kursor w ramach programu BASIC (inaczej niż automatyczne włączenie kursora w przypadku oczekiwania przez BASIC na znaki z klawiatury), należy użyć rozkazu CURSOR z parametrem ⟨przełącznik systemowy⟩ równym 1
&04	4	EOT	0 do 2	Ustawia tryb pracy ekranu. Parametry przyjmowane są modulo 4. Odpowiednik rozkazu MODE
&05	5	ENQ	0 do 255	Wysyła parametr dla kursora graficznego
&06	6	ACK		Włącza ekran tekstowy (patrz &15 NAK nieco dalej)
&07	7	BEL		Nadaje sygnał dźwiękowy. Uwaga, kasuje kolejki dźwiękowe
&08	8	*BS		Cofa kursor o jeden znak
&09	9	*TAB		Przesuwa kursor w przód o jeden znak
&0A	10	*LF		Przesuwa kursor w dół o jedną linię
&0B	11	*VT		Przesuwa kursor w górę o jedną linię
&0C	12	FF		Czyści okno tekstowe i ustawia kursor w lewym górnym rogu okna. Odpowiednik rozkazu CLS
&0D	13	*CR		Przesuwa kursor do lewego końca linii
&0E	14	SO	0 do 15	Określa atrament papieru. Parametr przyjmowany modulo 16. Odpowiednik rozkazu PAPER
&0F	15	SI	0 do 15	Określa atrament pióra. Parametr przyjmowany modulo 16. Odpowiednik rozkazu PEN

Wartość		Nazwa	Parametr	Działanie
&10	16	*DLE		Usuwa bieżący znak. Wypełnia pole znaku atramentem papieru.
&11	17	*DC1		Czyści linię od lewego skraju okna do aktualnej pozycji znaku, włącznie z tym znakiem. Zapełnia pola znaków atramentem papieru
&12	18	*DC2		Czyści linię od aktualnej pozycji znaku, włącznie z tym znakiem, do prawego skraju okna. Zapełnia pola znaków atramentem papieru
&13	19	*DC3		Czyści linie od początku okna do aktualnej pozycji znaku, włącznie z tym znakiem. Zapełnia pola znaków atramentem papieru
&14	20	*DC4		Czyści linie od aktualnej pozycji znaku, włącznie z tym znakiem, do końca okna. Zapełnia pola znaków atramentem papieru
&15	21	NAK		Wyłącza ekran tekstowy. Ekran nie reaguje na żaden wysyłany do niego znak, dopóki nie zostanie wysłany do ekranu znak ACK (&06)
&16	22	SYN	0 do 1	Określa tryb transparentny: 0 wyłącza, 1 włącza. Parametr modulo 2
&17	23	ETB	0 do 3	Określa tryb użycia atramentu graficznego: 0 - tryb normalny 1 - tryb XOR 2 - tryb AND 3 - tryb OR Parametr modulo 4.
&18	24	CAN		Wymienia ze sobą atramenty pióra i papieru
&19	25	EM	0 do 255 0 do 255 0 do 255 0 do 255 0 do 255 0 do 255 0 do 255 0 do 255 0 do 255	Zakłada tablicę znaku, definiowanego przez użytkownika. Odpowiednik rozkazu SYMBOL. Przyjmuje dziewięć parametrów. Pierwszy parametr określa numer definiowanego znaku, następne osiem określają treść tablicy. Najbardziej znaczący bit pierwszego bajtu odpowiada lewemu górnemu punktowi znaku; najmniej znaczący bit ostatniego bajtu odpowiada dolnemu prawemu punktowi znaku

Wartość		Nazwa	Parametr	Działanie
& 1A	26	SUB	1 do 80 1 do 80 1 do 25 1 do 25	Definiuje okno. Odpowiednik rozkazu WINDOW. Dwa pierwsze parametry określają lewy i prawy skraj okna - mniejsza wartość jest przyjmowana jako lewy, większa wartość - jako prawy skraj okna. Dwa dalsze parametry określają górny i dolny skraj okna - mniejsza wartość jest przyjmowana jako górny, większa wartość - jako dolny skraj okna
& 1B	27	ESC		Nie ma żadnego działania. Znak ignorowany
& 1C	28	FS	0 do 15	Przypisuje atramentowi dwa kolory. Pierwszy parametr (modulo 16) podaje numer atramentu, dwa następne (modulo 32) określają kolory (wartość 27 do 31 dają kolory niezdefiniowane). Odpowiednik rozkazu INK.
& 1D	29	GS	0 do 31	Przypisuje ramce dwa kolory. Dwa parametry (modulo 32) określają kolory (wartości 27 do 31 dają kolory niezdefiniowane). Odpowiednik rozkazu BORDER
& 1E	30	RS		Przesuwa kursor do lewego górnego brzegu okna
& 1F	31	US	1 do 80 1 do 25	Przesuwa kursor do określonej pozycji okna. Odpowiednik rozkazu LOCATE. Pierwszy parametr określa numer kolumny a drugi - numer linii

CPC6128 jest wyposażony w specjalnie opracowany, szybko działający system operacyjny, który "kieruje ruchem" wewnątrz komputera, od wejścia do wyjścia.

System operacyjny przede wszystkim pośredniczy pomiędzy elementami sprzętowymi komputera a interpreterem języka BASIC - na przykład przy realizacji funkcji migotania atramentu, gdzie BASIC deklaruje parametry - a system operacyjny przyjmuje te parametry wraz z poleceniem wykonania określonego zadania i zmienia odpowiednio kolory atramentu w zadanych przedziałach czasowych.

System operacyjny, nazywany "oprogramowaniem firmowym" ("firmware") zawiera procedury programowe w kodzie maszynowym, które są wywoływane przez rozkazy wyższego poziomu w języku BASIC.

Jeżeli masz zamiar wpisywać za pomocą instrukcji POKE w obszary pamięci, używane przez system operacyjny lub wywoływać za pomocą instrukcji CALL podprogramy, zapisz najpierw swój program na dysku, bo łatwo możesz go stracić!

Całe obszerne oprogramowanie firmowe CPC6128 jest opisane w opracowaniu SOFT968 a jego omówienie przekracza zakres tego podręcznika.

W przypadku pisania programów, zawierających obszerne części w kodzie maszynowym, niezbędne jest użycie asemblera. Polecić tu można pakiet programowy firmy AMSOFT o nazwie DEVPAC, zawierający relekasalny asembler Z80, edytor, disasembler i monitor.

Część 2: Przerwania

W 6128 szeroko wykorzystywany jest system przerwań Z80, co umożliwia wykonywanie przez system operacyjny kilku zadań jednocześnie. Przerwania używane są np. przez opisane wcześniej instrukcje AFTER i EVERY. Priorytet obsługi poszczególnych zdarzeń czasowych jest następujący:

Przerwanie programu BASIC klawiszami [ESC] [ESC]
Zegar 3
Zegar 2 (i trzy kanały kolejek dźwiękowych)
Zegar 1
Zegar 0

Przerwania mogą być włączane do programów po wzięciu pod uwagę możliwości wystąpienia pośrednich stanów zmiennych w chwili przerwania. Podprogramy obsługi przerwań nie powinny wywoływać żadnych niepożądanych zmian stanu zmiennych głównego programu.

Kolejki dźwiękowe wykorzystują niezależne przerwania o równym priorytecie. Przerwanie dźwiękowe nie może być zakłócone przez żadne inne przerwanie tego rodzaju. Umożliwia to procedurom obsługi przerwań dźwiękowych rozdzielać zmienne bez obawy wystąpienia opisanych wyżej efektów.

Po odblokowaniu systemu przerwań kolejek dźwiękowych (przy użyciu ON SQ GOSUB), przerwanie wystąpi natychmiast, jeżeli

kolejka dźwiękowa kanału nie jest pełna lub gdy zakończy się bieżący dźwięk i jest miejsce w kolejce na następną. Przyjęcie przerwania blokuje system przerwań tak, że procedura obsługi musi zawierać polecenie odblokowania tego systemu, jeżeli wymagane jest przyjmowanie dalszych przerwań.

Zarówno próba wydawania dźwięku jak i sprawdzanie statusu kolejki także blokuje system przerwań dźwiękowych.

Część 3: Znaki ASCII i znaki graficzne w systemie BASIC

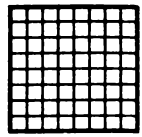
ASCII

Poniżej zamieszczono tabelę, zawierającą pełny zestaw standardowych znaków ASCII; podano wartość znaku w zapisie dziesiętnym /DEC/, ósemkowym /OCTAL/ i szestnastkowym /HEX/ oraz nazwę lub symbol znaku. Poszczególne znaki przedstawiono także szczegółowo w następnym punkcie.

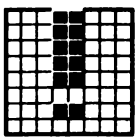
DEC	OCTAL	HEX	ASCII characters
0	000	00	NUL ([CTRL]@)
1	001	01	SOH ([CTRL]A)
2	002	02	STX ([CTRL]B)
3	003	03	ETX ([CTRL]C)
4	004	04	EOT ([CTRL]D)
5	005	05	ENQ ([CTRL]E)
6	006	06	ACK ([CTRL]F)
7	007	07	BEL ([CTRL]G)
8	010	08	BS ([CTRL]H)
9	011	09	HT ([CTRL]I)
10	012	0A	LF ([CTRL]J)
11	013	0B	VT ([CTRL]K)
12	014	0C	FF ([CTRL]L)
13	015	0D	CR ([CTRL]M)
14	016	0E	SO ([CTRL]N)
15	017	0F	SI ([CTRL]O)
16	020	10	DLE ([CTRL]P)
17	021	11	DC1 ([CTRL]Q)
18	022	12	DC2 ([CTRL]R)
19	023	13	DC3 ([CTRL]S)
20	024	14	DC4 ([CTRL]T)
21	025	15	NAK ([CTRL]U)
22	026	16	SYN ([CTRL]V)
23	027	17	ETB ([CTRL]W)
24	030	18	CAN ([CTRL]X)
25	031	19	EM ([CTRL]Y)
26	032	1A	SUB ([CTRL]Z)
27	033	1B	ESC
28	034	1C	FS
29	035	1D	GS
30	036	1E	RS
31	037	1F	US
32	040	20	SP
33	041	21	!
34	042	22	"
35	043	23	#
36	044	24	$
37	045	25	%
38	046	26	&
39	047	27	'
40	050	28	(
41	051	29	)
42	052	2A	*
43	053	2B	+
44	054	2C	,
45	055	2D	-
46	056	2E	.
47	057	2F	/
48	060	30	0
49	061	31	1
50	062	32	2
51	063	33	3
52	064	34	4
53	065	35	5
54	066	36	6
55	067	37	7
56	070	38	8
57	071	39	9
58	072	3A	:
59	073	3B	;
60	074	3C	<
61	075	3D	=
62	076	3E	>
63	077	3F	?
64	100	40	@
65	101	41	A
66	102	42	B
67	103	43	C
68	104	44	D
69	105	45	E
70	106	46	F
71	107	47	G
72	110	48	H
73	111	49	I
74	112	4A	J
75	113	4B	K
76	114	4C	L
77	115	4D	M
78	116	4E	N
79	117	4F	O
80	120	50	P
81	121	51	Q
82	122	52	R
83	123	53	S
84	124	54	T
85	125	55	U
86	126	56	V
87	127	57	W
88	130	58	X
89	131	59	Y
90	132	5A	Z
91	133	5B	[
92	134	5C	\
93	135	5D	]
94	136	5E	^
95	137	5F	_
96	140	60	`
97	141	61	a
98	142	62	b
99	143	63	c
100	144	64	d
101	145	65	e
102	146	66	f
103	147	67	g
104	150	68	h
105	151	69	i
106	152	6A	j
107	153	6B	k
108	154	6C	l
109	155	6D	m
110	156	6E	n
111	157	6F	o
112	160	70	p
113	161	71	q
114	162	72	r
115	163	73	s
116	164	74	t
117	165	75	u
118	166	76	v
119	167	77	w
120	170	78	x
121	171	79	y
122	172	7A	z
123	173	7B	{
124	174	7C	|
125	175	7D	}
126	176	7E	~

Zestaw znaków graficznych w systemie BASIC

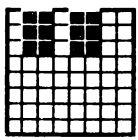
Poniżej zamieszczono rysunki znaków graficznych, używanych w języku BASIC. Znaki narysowano w polu matrycy 8 x 8 punktów, używanej do wyświetlania znaków na ekranie monitora 6128. W celu uzyskania specjalnych efektów, użytkownik może także definiować własne znaki, zastępujące przedstawione na rysunkach. Patrz punkt "Znaki definiowane przez użytkownika" w rozdziale "W wolnej chwili..."



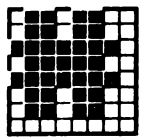
32 &H20
&X00100000



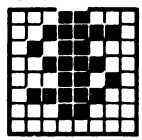
33
&H21
&X00100001



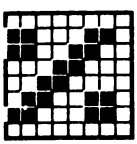
34
&H22
&X00100010



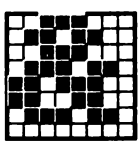
35
&H23
&X00100011



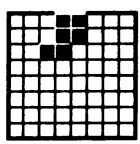
36
&H24
&X00100100



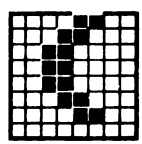
37
&H25
&X00100101



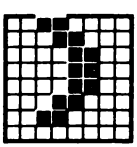
38
&H26
&X00100110



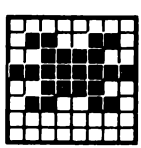
39
&H27
&X00100111



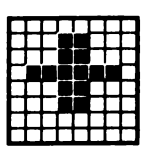
40
&H28
&X00101000



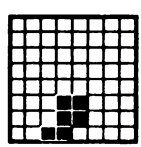
41
&H29
&X00101001



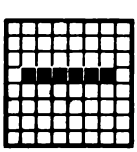
42
&H2A
&X00101010



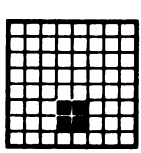
43
&H2B
&X00101011



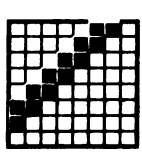
44
&H2C
&X00101100



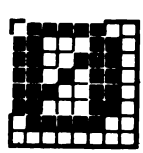
45
&H2D
&X00101101



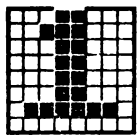
46
&H2E
&X00101110



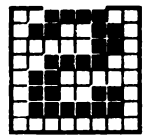
47
&H2F
&X00101111



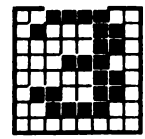
48
&H30
&X00110000



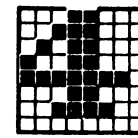
49
&H31
&X00110001



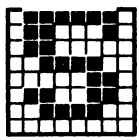
50
&H32
&X00110010



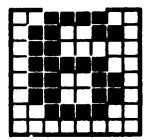
51
&H33
&X00110011



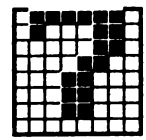
52
&H34
&X00110100



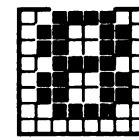
53
&H35
&X00110101



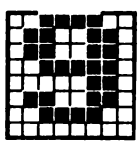
54
&H36
&X00110110



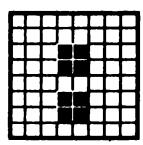
55
&H37
&X00110111



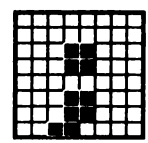
56
&H38
&X00111000



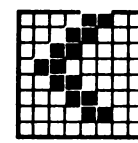
57
&H39
&X00111001



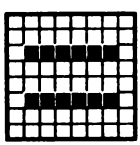
58
&H3A
&X00111010



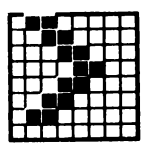
59
&H3B
&X00111011



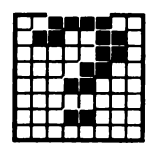
60
&H3C
&X00111100



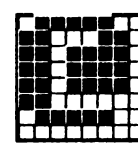
61
&H3D
&X00111101



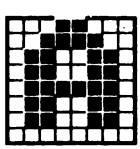
62
&H3E
&X00111110



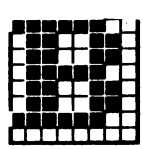
63
&H3F
&X00111111



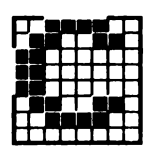
64
&H40
&X01000000



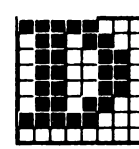
65
&H41
&X01000001



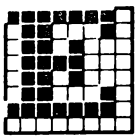
66
&H42
&X01000010



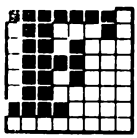
67
&H43
&X01000011



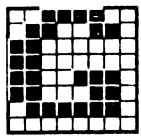
68
&H44
&X01000100



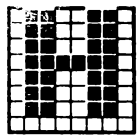
69
&H45
&X01000101



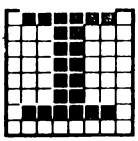
70
&H46
&X01000110



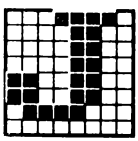
71
&H47
&X01000111



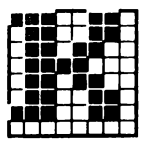
72
&H48
&X01001000



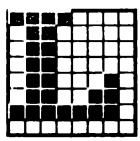
73
&H49
&X01001001



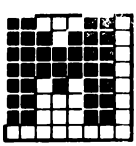
74
&H4A
&X01001010



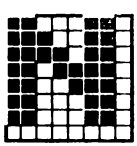
75
&H4B
&X01001011



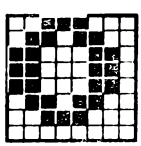
76
&H4C
&X01001100



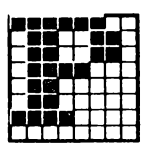
77
&H4D
&X01001101



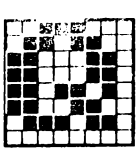
78
&H4E
&X01001110



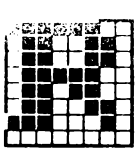
79
&H4F
&X01001111



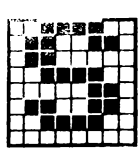
80
&H50
&X01010000



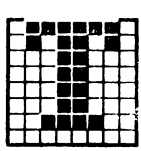
81
&H51
&X01010001



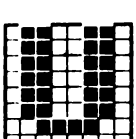
82
&H52
&X01010010



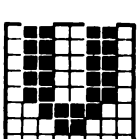
83
&H53
&X01010011



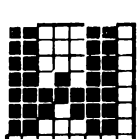
84
&H54
&X01010100



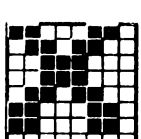
85
&H55
&X01010101



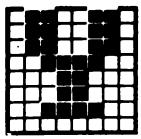
86
&H56
&X01010110



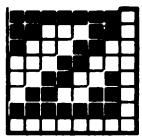
87
&H57
&X01010111



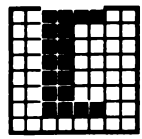
88
&H58
&X01011000



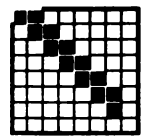
89
&H59
&X01011001



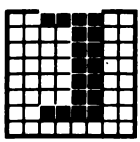
90
&H5A
&X01011010



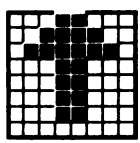
91
&H5B
&X01011011



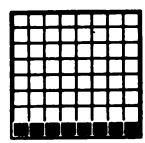
92
&H5C
&X01011100



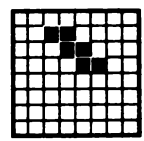
93
&H5D
&X01011101



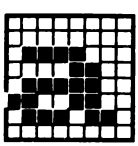
94
&H5E
&X01011110



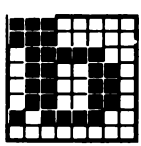
95
&H5F
&X01011111



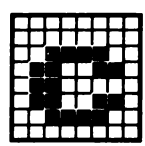
96
&H60
&X01100000



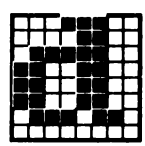
97
&H61
&X01100001



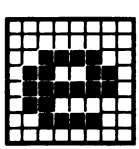
98
&H62
&X01100010



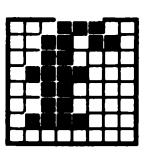
99
&H63
&X01100011



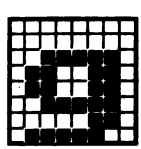
100
&H64
&X01100100



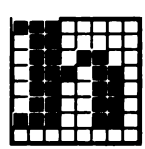
101
&H65
&X01100101



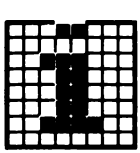
102
&H66
&X01100110



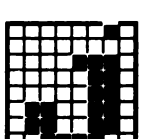
103
&H67
&X01100111



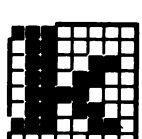
104
&H68
&X01101000



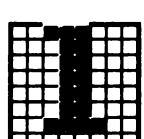
105
&H69
&X01101001



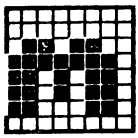
106
&H6A
&X01101010



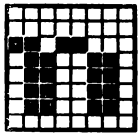
107
&H6B
&X01101011



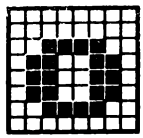
108
&H6C
&X01101100



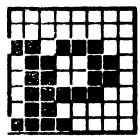
109
&H6D
&X01101101



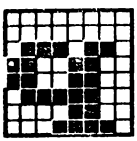
110
&H6E
&X01101110



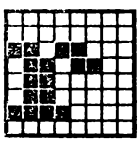
111
&H6F
&X01101111



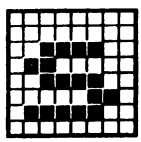
112
&H70
&X01110000



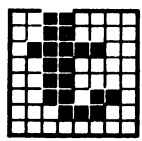
113
&H71
&X01110001



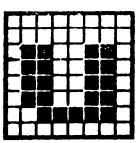
114
&H72
&X01110010



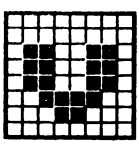
115
&H73
&X01110011



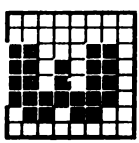
116
&H74
&X01110100



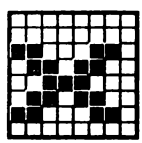
117
&H75
&X01110101



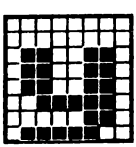
118
&H76
&X01110110



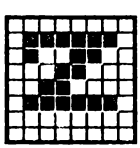
119
&H77
&X01110111



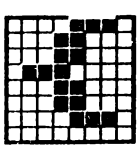
120
&H78
&X01111000



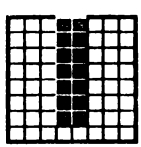
121
&H79
&X01111001



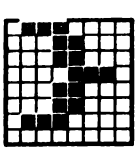
122
&H7A
&X01111010



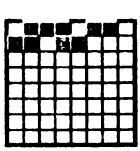
123
&H7B
&X01111011



124
&H7C
&X01111100

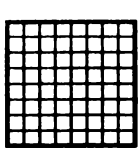


125
&H7D
&X01111101

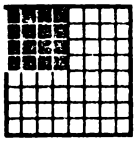


126
&H7E
&X01111110

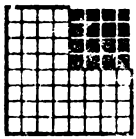
127
&H7F
&X01111111



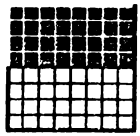
128
&H80
&X10000000



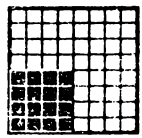
129
&H81
&X10000001



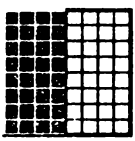
130
&H82
&X10000010



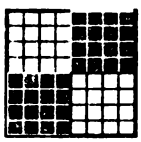
131
&H83
&X10000011



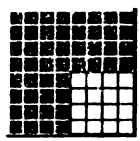
132
&H84
&X10000100



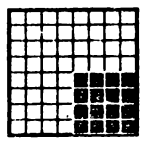
133
&H85
&X10000101



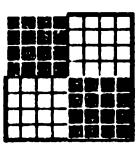
134
&H86
&X10000110



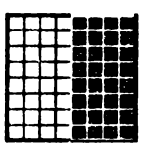
135
&H87
&X10000111



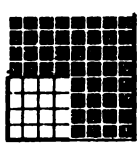
136
&H88
&X10001000



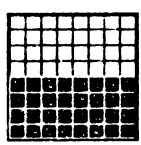
137
&H89
&X10001001



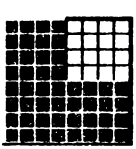
138
&H8A
&X10001010



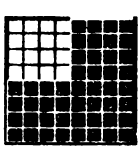
139
&H8B
&X10001011



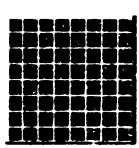
140
&H8C
&X10001100



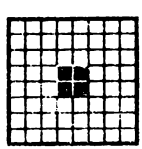
141
&H8D
&X10001101



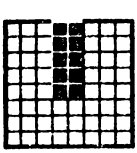
142
&H8E
&X10001110



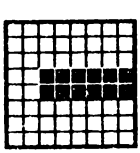
143
&H8F
&X10001111



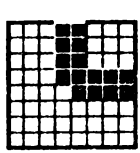
144
&H90
&X10010000



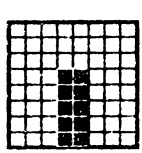
145
&H91
&X10010001



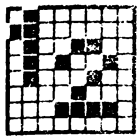
146
&H92
&X10010010



147
&H93
&X10010011

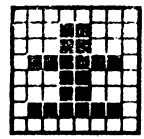


148
&H94
&X10010100

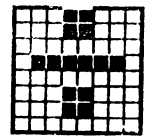


169
&HA9
&X10101001

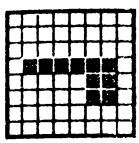
170
&HAA
&X10101010



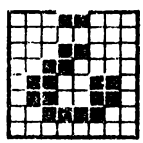
171
&HAB
&X10101011



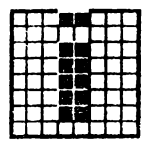
172
&HAC
&X10101100



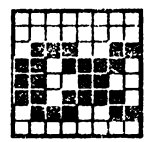
173
&HAD
&X10101101



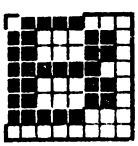
174
&HAE
&X10101110



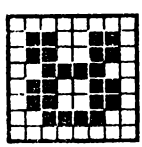
175
&HAF
&X10101111



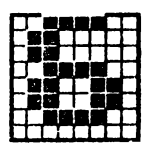
176
&HB0
&X10110000



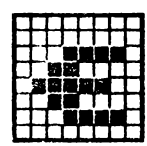
177
&HB1
&X10110001



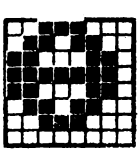
178
&HB2
&X10110010



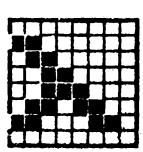
179
&HB3
&X10110011



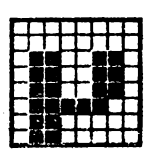
180
&HB4
&X10110100



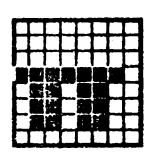
181
&HB5
&X10110101



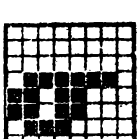
182
&HB6
&X10110110



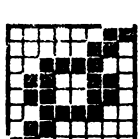
183
&HB7
&X10110111



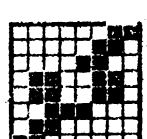
184
&HB8
&X10111000



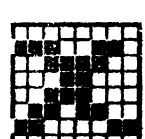
185
&HB9
&X10111001



186
&HBA
&X10111010

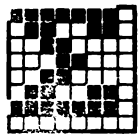


187
&HBB
&X10111011

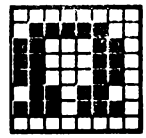


188
&HBC
&X10111100

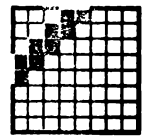
189
&HBD
&X10111101



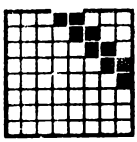
190
&HBE
&X10111110



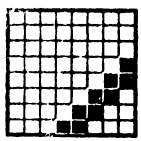
191
&HBF
&X10111111



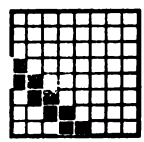
192
&HC0
&X11000000



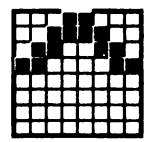
193
&HC1
&X11000001



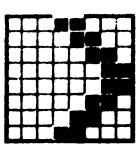
194
&HC2
&X11000010



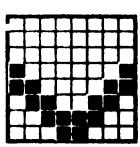
195
&HC3
&X11000011



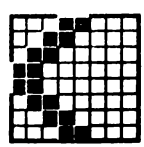
196
&HC4
&X11000100



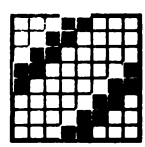
197
&HC5
&X11000101



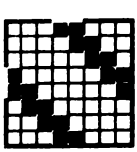
198
&HC6
&X11000110



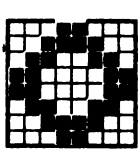
199
&HC7
&X11000111



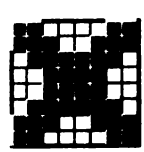
200
&HC8
&X11001000



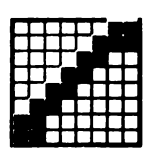
201
&HC9
&X11001001



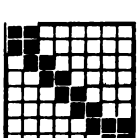
202
&HCA
&X11001010



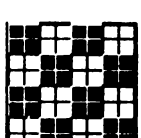
203
&HCB
&X11001011



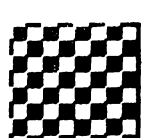
204
&HCC
&X11001100



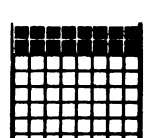
205
&HCD
&X11001101



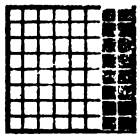
206
&HCE
&X11001110



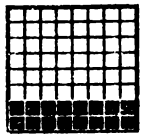
207
&HCF
&X11001111



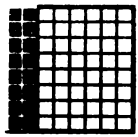
208
&HD0
&X11010000



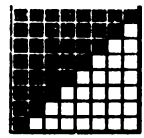
209
&HD1
&X11010001



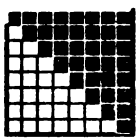
210
&HD2
&X11010010



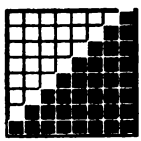
211
&HD3
&X11010011



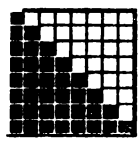
212
&HD4
&X11010100



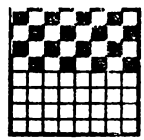
213
&HD5
&X11010101



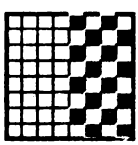
214
&HD6
&X11010110



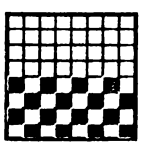
215
&HD7
&X11010111



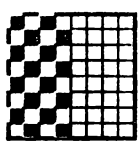
216
&HD8
&X11011000



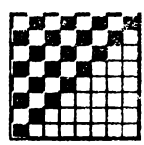
217
&HD9
&X11011001



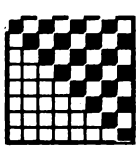
218
&HDA
&X11011010



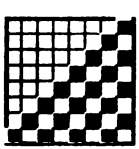
219
&HDB
&X11011011



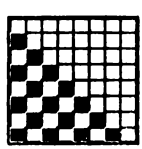
220
&HDC
&X11011100



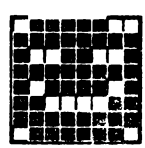
221
&HDD
&X11011101



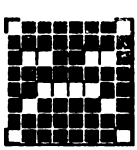
222
&HDE
&X11011110



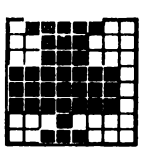
223
&HDF
&X11011111



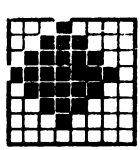
224
&HE0
&X11100000



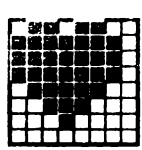
225
&HE1
&X11100001



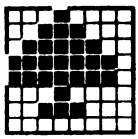
226
&HE2
&X11100010



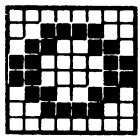
227
&HE3
&X11100011



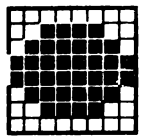
228
&HE4
&X11100100



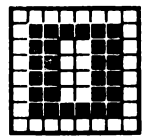
229
&HE5
&X11100101



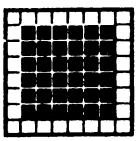
230
&HE6
&X11100110



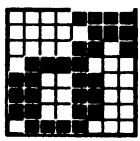
231
&HE7
&X11100111



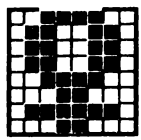
232
&HE8
&X11101000



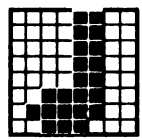
233
&HE9
&X11101001



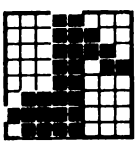
234
&HEA
&X11101010



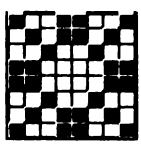
235
&HEB
&X11101011



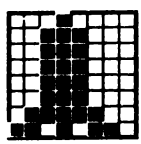
236
&HEC
&X11101100



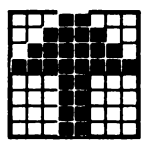
237
&HED
&X11101101



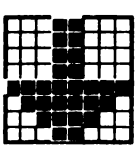
238
&HEE
&X11101110



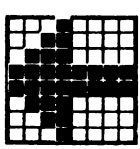
239
&HEF
&X11101111



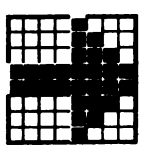
240
&HF0
&X11110000



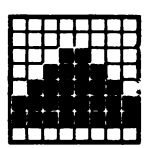
241
&HF1
&X11110001



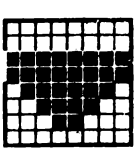
242
&HF2
&X11110010



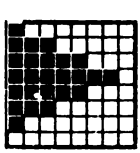
243
&HF3
&X11110011



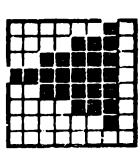
244
&HF4
&X11110100



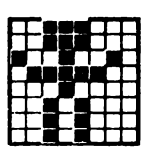
245
&HF5
&X11110101



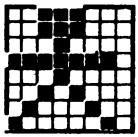
246
&HF6
&X11110110



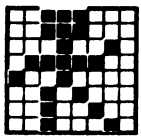
247
&HF7
&X11110111



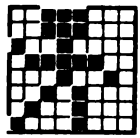
248
&HF8
&X1111[illegible]00



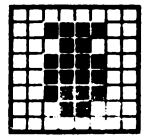
249
&HF9
&X11111001



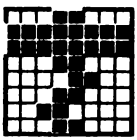
250
&HFA
&X11111010



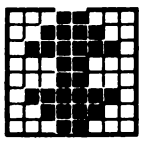
251
&HFB
&X11111011



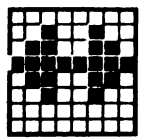
252
&HFC
&X11111100



253
&HFD
&X11111101



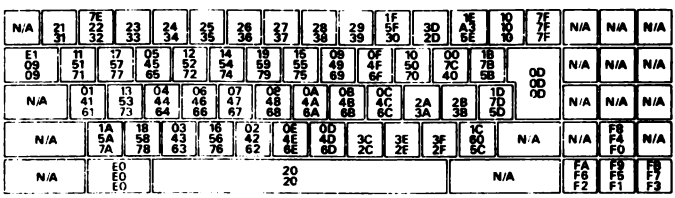
254
&HFE
&X11111110

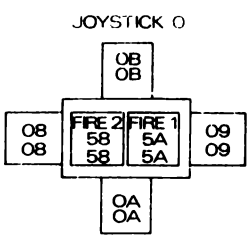


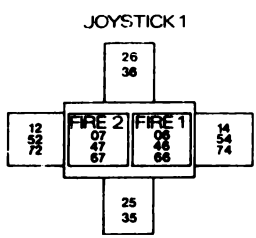
255
&HFF
&X11111111

Część 4: Dane dotyczące klawiszy

W części tej zestawiono dane związane z klawiszami. Domyślnie przyjmowane wartości ASCII / w zapisie szestnastkowym/ przypisane poszczególnym klawiszom klawiatury i joystickom. Klawisze oznaczone N/A nie wysyłają kodów ASCII.

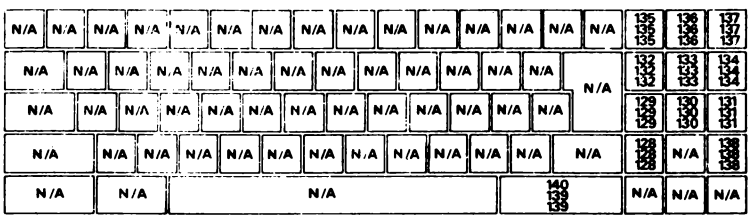






Znaki dodatkowe, domyślnie przyjmowane lokacje i wartości.

Klawisze oznaczone N/A nie wysyłają znaków dodatkowych; znaki dodatkowe 13 do 41 /141 do 159/ mają domyślną wartość zero i mogą mieć inną wartość przypisaną im za pomocą rozkazu KEY oraz zostać przyporządkowane określonemu klawiszowi za pomocą rozkazu KEY DEF.



EXPANSION CHARACTER	DEFAULT SETTING	
	CHARACTER	ASCII VALUE
0(128)	0	&30
1(129)	1	&31
2(130)	2	&32
3(131)	3	&33
4(132)	4	&34
5(133)	5	&35
6(134)	6	&36
7(135)	7	&37
8(136)	8	&38
9(137)	9	&39
10(139)	.	&2E
11(139)	[RETURN]	&0D
12(140)	RUN"[RETURN]	&52 &55 &4E &22 &0D

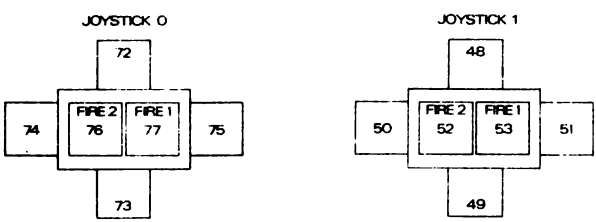
Note: Expansion characters 13 to 31 (141 to 159) have a null value by default. They have values assigned using the BASIC command KEY, and are assigned to keys using the command KEY DEF.

ZNAKI DODATKOWE	WARTOSC DOMYSLNA ZNAK	KOD ASCII
0(128)	0	&30
1(129)	1	&31
2(130)	2	&32
3(131)	3	&33
4(132)	4	&34
5(133)	*5	&35
6(134)	6	&36
7(135)	7	&37
8(136)	8	&38
9(137)	9	&39
10(138)	.	&2E
11(139)	[RETURN]	&0D
12(140)	RUN"[RETURN]	&52 &55 &4E &22 &0D

ZNAKI DODATKOWE	WARTOSC DOMYSLNA ZNAK	KOD ASCII
0(128)	0	&30
1(129)	1	&31
2(130)	2	&32
3(131)	3	&33
4(132)	4	&34
5(133)	5	&35
6(134)	6	&36
7(135)	7	&37
8(136)	8	&38
9(137)	9	&39
10(138)	.	&2E
11(139)	[RETURN]	&0D
12(140)	RUN"[RETURN]	&52 &55 &4E &22 &0D

Numery porządkowe przypisane poszczególnym klawiszom klawiatury i joystickom.

66	64	65	57	56	49	48	41	40	33	32	25	24	16	79	10	11	3
68	67	59	58	50	51	43	42	35	34	27	26	17	18		20	12	4
70	69	60	61	53	52	44	45	37	36	29	28	19			13	14	5
21		71	63	62	55	54	46	38	39	31	30	22	21		15	0	7
23		9				47						6			8	2	1



Część 5: Dźwięk.

Nuty; częstotliwość i okres tonu.

W części tej zamieszczono tabele nut /NOTE/ zwykłej z.ównoważonej skali temperowanej w pełnym zakresie ośmiu oktaw. Podano częstotliwość /FREQUENCY/ i okres /PERIOD/ każdego dźwięku.

Wytwarzane częstotliwości nie odpowiadają dokładnie żądanym częstotliwościom poszczególnych dźwięków, ponieważ deklarowany okres tonu musi być liczbą całkowitą. Wynikający z tego względny błąd częstotliwości, obliczony na podstawie różnicy między częstotliwością wymaganą i wytwarzaną podano w rubryce RELATIVE ERROR.

NOTE	FREQUENCY	PERIOD	RELATIVE ERROR	
C	16.352	3822	-0.007%	
C#	17.324	3608	+0.007%	
D	18.354	3405	-0.007%	
D#	19.445	3214	-0.004%	
E	20.602	3034	+0.009%	Octave-4
F	21.827	2863	-0.016%	
F#	23.125	2703	+0.009%	
G	24.500	2551	-0.002%	
G#	25.957	2408	+0.005%	
A	27,500	2273	+0.012%	
A#	29.135	2145	-0.008%	
B	30.868	2025	+0.011%	

NOTE	FREQUENCY	PERIOD	RELATIVE ERROR	
C	32.703	1911	-0.007%	
C#	34.648	1804	+0.007%	
D	36.708	1703	+0.022%	
D#	38.891	1607	-0.004%	
E	41.203	1517	+0.009%	
F	43.654	1432	+0.019%	Octave-3
F#	46.249	1351	-0.028%	
G	48.999	1276	+0.037%	
G#	51.913	1204	+0.005%	
A	55.000	1136	-0.032%	
A#	58.270	1073	+0.039%	
B	61.735	1012	-0.038%	

NOTE	FREQUENCY	PERIOD	RELATIVE ERROR	
C	65.406	956	+0.046%	
C#	69.296	902	+0.007%	
D	73.416	851	-0.037%	
D#	77.782	804	+0.058%	
E	82.407	758	-0.057%	
F	87.307	716	+0.019%	Octave -2
F#	92.499	676	+0.046%	
G	97.999	638	+0.037%	
G#	103.826	602	+0.005%	
A	110.000	568	-0.032%	
A#	116.541	536	-0.055%	
B	123.471	506	-0.038%	

NOTE	FREQUENCY	PERIOD	RELATIVE ERROR	
C	130.813	478	+0.046%	
C#	138.591	451	+0.007%	
D	146.832	426	+0.081%	
D#	155.564	402	+0.058%	
E	164.814	379	-0.057%	
F	174.614	358	+0.019%	Octave -1
F#	184.997	338	+0.046%	
G	195.998	319	+0.037%	
G#	207.652	301	+0.005%	
A	220.000	284	-0.032%	
A#	233.082	268	-0.055%	
B	246.942	253	-0.038%	

NOTE	FREQUENCY	PERIOD	RELATIVE ERROR	
C	261.626	239	+0.046%	Middle C
C#	277.183	225	-0.215%	
D	293.665	213	+0.081%	
D#	311.127	201	+0.058%	
E	329.628	190	+0.206%	
F	349.228	179	+0.019%	Octave 0
F#	369.994	169	+0.046%	
G	391.995	159	-0.277%	
G#	415.305	150	-0.328%	
A	440.000	142	-0.032%	International A
A#	466.164	134	-0.055%	
B	493.883	127	+0.356%	

NOTE	FREQUENCY	PERIOD	RELATIVE ERROR	
C	523.251	119	-0.374%	
C#	554.365	113	+0.229%	
D	587.330	106	-0.390%	
D#	622.254	100	-0.441%	
E	659.255	95	+0.206%	
F	698.457	89	-0.543%	Octave 1
F#	739.989	84	-0.548%	
G	783.991	80	+0.350%	
G#	830.609	75	-0.328%	
A	880.000	71	-0.032%	
A#	932.328	67	-0.055%	
B	987.767	63	-0.435%	

NOTE	FREQUENCY	PERIOD	RELATIVE ERROR	
C	1046.502	60	+0.462%	
C#	1108.731	56	-0.662%	
D	1174.659	53	-0.390%	
D#	1244.508	50	-0.441%	
E	1318.510	47	-0.855%	
F	1396.913	45	+0.574%	
F#	1479.978	42	-0.548%	Octave 2
G	1567.982	40	+0.350%	
G#	1661.219	38	+0.992%	
A	1760.000	36	+1.357%	
A#	1864.655	34	+1.417%	
B	1975.533	32	+1.134%	

NOTE	FREQUENCY	PERIOD	RELATIVE ERROR	
C	2093.004	30	+0.462%	
C#	2217.461	28	-0.662%	
D	2349.318	27	+1.469%	
D#	2489.016	25	-0.441%	
E	2637.021	24	+1.246%	
F	2793.826	22	-1.685%	
F#	2959.955	21	-0.548%	Octave 3
G	3135.963	20	+0.350%	
G#	3322.438	19	+0.992%	
A	3520.000	18	+1.357%	
A#	3729.310	17	+1.417%	
B	3951.066	16	+1.134%	

2 S y n t a x E r r o r
B ł ą d s y n t a k t y c z n y (składni)
BASIC nie rozumie wprowadzonej linii, gdyż konstrukcja linii nie jest zgodna z regułami języka

3 U n e x p e c t e d r e t u r n
N i e o c z e k i w a n y r e t u r n
Instrukcja RETURN znaleziona poza podprogramem

4 D A T A e x h a u s t e d
w y c z e r p a n y z b i ó r D A T A
Instrukcja READ nakazuje pobranie więcej danych niż zawierają instrukcje DATA

5 I m p r o p e r a r g u m e n t
N i e w ł a ś c i w y a r g u m e n t
Błąd wynikający z wielu przyczyn. Wartość argumentu funkcji lub parametru instrukcji (rozkazu) w jakiś sposób niewłaściwa.

6 O v e r f l o w
P r z e p e ł n i e n i e
Wynik operacji arytmetycznej przekracza dopuszczalny zakres. Może to być przekroczenie zakresu przyjmowanych liczb zmiennoprzecinkowych, jeżeli wynik jakiejś operacji osiąga wartość większą niż ok. 1.7 E 38 lub też błąd taki może być wynikiem próby zamiany liczby zmiennoprzecinkowej na 16-bitową liczbę całkowitą ze znakiem.

7 M e m o r y f u l l
P a m i ę ć z a p e ł n i o n a
Wykonywany program lub jego zmienne wymagają większego obszaru pamięci, albo też struktury kontrolne programu są zbyt głęboko zagnieżdżone (zagnieżdżone instrukcje GOSUB, WHILE lub FOR).

Błąd tego rodzaju może powstać także w przypadku gdy za pomocą rozkazu MEMORY przydzieli się zbyt mały zakres pamięci lub próbuje się określić zbyt duży jego zakres. Należy zauważyć, że każdy otwarty zbiór dyskowy wymaga wydzielenia w pamięci odpowiedniego bufora dla obsługi tego zbioru, co ogranicza zakres pamięci, jaki może być używany przez program.

8 L i n e d o e s n o t e x i s t
L i n i a n i e i s t n i e j e
Nie można znaleźć linii, do której jest odwołanie

9 S u b s c r i p t o u t o f r a n g e
I n d e k s p o z a z a k r e s e m
Jeden z indeksów w odwołaniu do tablicy (macierzy) jest zbyt duży lub zbyt mały

10 A r r a y a l r e a d y d i m e n s i o n e d
T a b l i c a j u ż z w y m i a r o w a n a
Jedna z tablic w instrukcji DIM została już zadeklarowana.

11 D i v i s i o n b y z e r o
D z i e l e n i e p r z e z z e r o
Może wystąpić przy dzieleniu liczb rzeczywistych, dzieleniu liczb całkowitych, dzieleniu modulo lub przy potęgowaniu

12 I n v a l i d d i r e c t c o m m a n d
N i e p r a w i d ł o w y r o z k a z b e z p o ś r e d n i
Ostatnio wprowadzony rozkaz nie może być użyty w trybie bezpośrednim

13 T y p e m i s m a t c h
N i e z g o d n o ś ć t y p ó w
Wprowadzono wartość liczbową w miejscu, gdzie powinna być wartość tekstowa lub odwrotnie, lub też wprowadzono nieprawidłowo sformułowaną liczbę w rozkazie READ albo INPUT

14 S t r i n g s p a c e f u l l
M i e j s c e n a c i ą g i z a p e ł n i o n e
Wprowadzono tak wiele ciągów, że nie ma więcej wolnego miejsca, nawet po uporządkowaniu.

15 S t r i n g t o o l o n g
C i ą g z b y t d ł u g i
Długość ciągu przekracza 255 znaków. Może się zdarzyć po połączeniu razem kilku ciągów.

16 String expression too complex
Wyrażenie tekstowe zbyt złożone
Wyrażenie tekstowe może wytwarzać pewną liczbę przejściowych wartości tekstowych. Gdy liczba tych wartości przekracza rozsądną granicę, zgłaszany jest w/w błąd

17 Cannot Continue
Nie można kontynuować
Z jakiejś przyczyny program nie może być wznowiony przez CONT. Weź pod uwagę, że CONT jest przeznaczony do wznawiania programu po instrukcji STOP, po [ESC] [ESC] lub po błędzie i jakiekolwiek wprowadzane w międzyczasie zmiany w programie uniemożliwiają wznowienie.

18 Unknown user function
Nieznana funkcja użytkownika
Nie wykonano DEF FN dla wywoływanej właśnie FN

19 RESUME missing
Pominięte RESUME
Osiągnięty został koniec programu podczas wykonywania procedury obsługi błędu (tj. w procedurze ON ERROR GOTO)

20 Unexpected RESUME
Nieoczekiwane RESUME
RESUME jest dopuszczalne tylko w procedurze obsługi błędu (tj. w procedurze ON ERROR GOTO)

21 Direct command found
Znaleziono rozkaz bezpośredni
Znaleziono linię bez numeru w programie wprowadzonym ze zbioru

22 Operand missing
Opuszczony operand
Wprowadzono niekompletne wyrażenie

23 Line too long
Linia zbyt długa
Po przekształceniu do postaci wewnętrznej języka BASIC linia zrobiła się zbyt długa

24 E O F m e t
   N a p o t k a n o E O F
Próbowano czytać za znakiem końca strumienia wejściowego (EOF = End Of File, Koniec zbioru)

25 F i l e t y p e e r r o r
   Z ł y t y p z b i o r u
Czytany zbiór nie jest odpowiedniego typu. OPENIN jest przeznaczone tylko do otwarcia zbiorów tekstowych ASCII. Podobnie, LOAD, RUN itp. są przeznaczone tylko do obsługi zbiorów tworzonych przez SAVE.

26 N E X T m i s s i n g
   O p u s z c z o n y N E X T
Brakuje rozkazu NFXT wymaganego przez rozkaz FOR. Numer linii podawany przy komunikacie odpowiada instrukcji FOR, dla której stwierdzono błąd.

27 F i l e a l r e a d y o p e n
   Z b i ó r j u ż o t w a r t y
Wprowadzono OPFNIN lub OPENOUT przed zamknięciem wcześniej otwartego zbioruk

28 U n k n o w n c o m m a n d
   N i e z n a n y r o z k a z
BASIC nie rozpoznaje rozkazu zewnętrznego, tj. rozkazu poprzedzonego przez kreskę

29 W E N D m i s s i n g
   O p u s z c z o n y W E N D
Nie znaleziono WFND wymaganego przez rozkaz WHILE.

30 U n e x p e c t e d W E N D
   N i e o c z e k i w a n y W E N D
Znaleziono WEND poza pętlą WHILE lub WEND, który nie pasuje do aktualnie wykonywanej pętli WHILE.

31 F i l e n o t o p e n
   Z b i ó r n i e o t w a r t y
(Patrz następny punkt "Błędy działania dysku")

32 B r o k e n i n
Z a ł a m a n i e
(Patrz następny punkt "Błędy działania dysku")

Błędy działania dysku w systemie AMSDOS

Podczas jakiejkolwiek operacji na zbiorach dyskowych wystąpić może kilka rodzajów błędów. Każdy z tych błędów jest sygnalizowany przez BASIC jako ERRor (błąd) numer 32. Bardziej szczegółowe informacje o błędzie uzyskać można za pomocą funkcji DERR, wprowadzonej po wykryciu błędu o tym numerze. Funkcja DERR podaje następujące wartości:

Błąd AMSDOS	Wartość DERR	Przyczyna błędu
0	0 lub 22	Naciśnięto ESC
14	142 (128+14)	Strumień nie jest w odpowiednim stanie
15	143 (128+15)	Osiągnięto fizyczny koniec zbioru
16	144 (128+16)	Zły rozkaz, najczęściej błędna nazwa zbioru
17	145 (128+17)	Zbiór już istnieje
18	146 (128+18)	Zbiór nie istnieje
19	147 (128+19)	Skorowidz dysku pełny
20	148 (128+20)	Dysk całkowicie zapełniony
21	149 (128+21)	Wymieniono dysk z otwartymi zbiorami
22	150 (128+22)	Zbiór jest "tylko do czytania" (Read/Only)
26	154 (128+26)	Wykryto logiczny koniec zbioru

Wartości DERR są z reguły większe od 128, gdyż zgłoszenie błędu przez AMSDOS powoduje wpisanie 1 na pozycji najstarszego (7-ego) bitu.

Inne wartości funkcji DERR sygnalizują błędy, zgłaszane przez układ kontrolera dysku. W takim przypadku wartość 1 wpisywana jest pozycji 6-go bitu, bit 7-my wskazuje, czy błąd był zgłaszany przez AMSDOS, a pozostałe bity określają rodzaj błędu. Znaczenie poszczególnych bitów jest przy tym następujące:

Bit	Znaczenie
0	Nie znaleziony znacznik adresu (address mark)
1	Zapis niemożliwy - dysk zabezpieczony przed zapisem
2	Brak danych - nie można znaleźć żądanego sektora dysku
3	Napęd dyskowy nie jest gotowy do pracy - brak dysku w napędzie dyskowym
4	Dane nałożone na siebie (overrun error)
5	Błąd danych - błąd sumy kontrolnej (CRC)
6	Zawsze "1", co oznacza błąd sygnalizowany przez kontroler dysku
7	"1", gdy błąd był już zgłaszany przez AMSDOS

ERR może mieć także wartość 31 w przypadku próby dostępu do zbioru, który nie został wcześniej otwarty.

Funkcje ERR i DERR mogą być użyte w procedurze obsługi błędu ON ERROR GOTO, gdzie sprawdza się czy ERR ma wartość 31 lub 32, a w przypadku 32 za pomocą funkcji DERR uzyskuje się bardziej szczegółowe informacje o przyczynie powstania błędu. Na przykład:

```
1Ø  ON ERROR GOTO 1ØØØ
2Ø  OPENOUT "myfile.asc"
3Ø  WRITE # 9 "test-data"
4Ø  CLOSEOUT
5Ø  END
1ØØØ    amsdoserr = (DERR AND &7F): REM wymaskuj bit 7
1Ø1Ø    IF ERR<31  THEN END
1Ø2Ø    IF ERR=31 THEN PRINT "jesteś pewien, że poprawnie
        napisałeś linię 20?": END
1Ø3Ø    IF amsdoser = 2Ø THEN PRINT "dysk zapełniony, użyj
        nowy dysk" : END
1Ø4Ø    IF amsdoser = &XØ1ØØ1ØØØ THEN PRINT "włóż dysk do
        napędu dyskowego, następnie naciśnij jakikolwiek klawisz:
        WHILE INKEY$ = " " : WEND:RESUME
1Ø5Ø    END
```

Część 7: Słowa kluczowe języka BASIC

W części tej zamieszczono pełną listę słów kluczowych języka BASIC AMSTRAD CPC 6128.
Słowa te są zastrzeżone i NIE mogą być używane jako nazwy zmiennych.

ABS, AFTER, AND, ASC, ATN, AUTO

BIN$, BORDER

CALL, CAT, CHAIN, CHR$, CINT, CLEAR, CLG, CLOSEIN, CLOSEOUT, CLS, CONT, COPYCHR$, COS, CREAL, CURSOR

DATA, DEC$, DEF, DEFINT, DEFREAL, DEFSTR, DEG, DELETE, DERR, DI, DIM, DRAW, DRAWR

EDIT, EI, ELSE, END, ENT, ENV, EOF, ERASE, ERL, ERR, ERROR, EVERY, EXP

FILL, FIX, FN, FOR, FRAME, FRE

GOSUB, GOTO, GRAPHICS

HEX$, HIMEM

IF, INK, INKEY, INKEY$, INP, INPUT, INSTR, INT

JOY

KEY

LEFT$, LEN, LET, LINE, LIST, LOAD, LOCATE, LOG, LOG10, LOWER$

MASK, MAX, MEMORY, MERGE, MID$, MIN, MOD, MODE, MOVE, MOVER

NEXT, NEW, NOT

ON, ON BREAK, ON ERROR GOTO 0, ON SQ, OPENIN, OPENOUT, OR, ORIGIN, OUT

PAPER, PEEK, PEN, PI, PLOT, PLOTR, POKE, POS, PRINT

RAD, RANDOMIZE, READ, RELEASE, REM, REMAIN, RENUM, RESTORE, RESUME, RETURN, RIGHT$, RND, ROUND, RUN

SAVE, SGN, SIN, SOUND, SPACE$, SPC, SPEED, SQ, SQR, STEP, STOP, STR$, STRING$, SWAP, SYMBOL

TAB, TAG, TAGOFF, TAN, TEST, TESTR, THEN, TIME, TO, TROFF, TRON

UNT, UPPER$, USING

VAL, VPOS

WAIT, WEND, WHILE, WIDTH, WINDOW, WRITE

XOR, XPOS

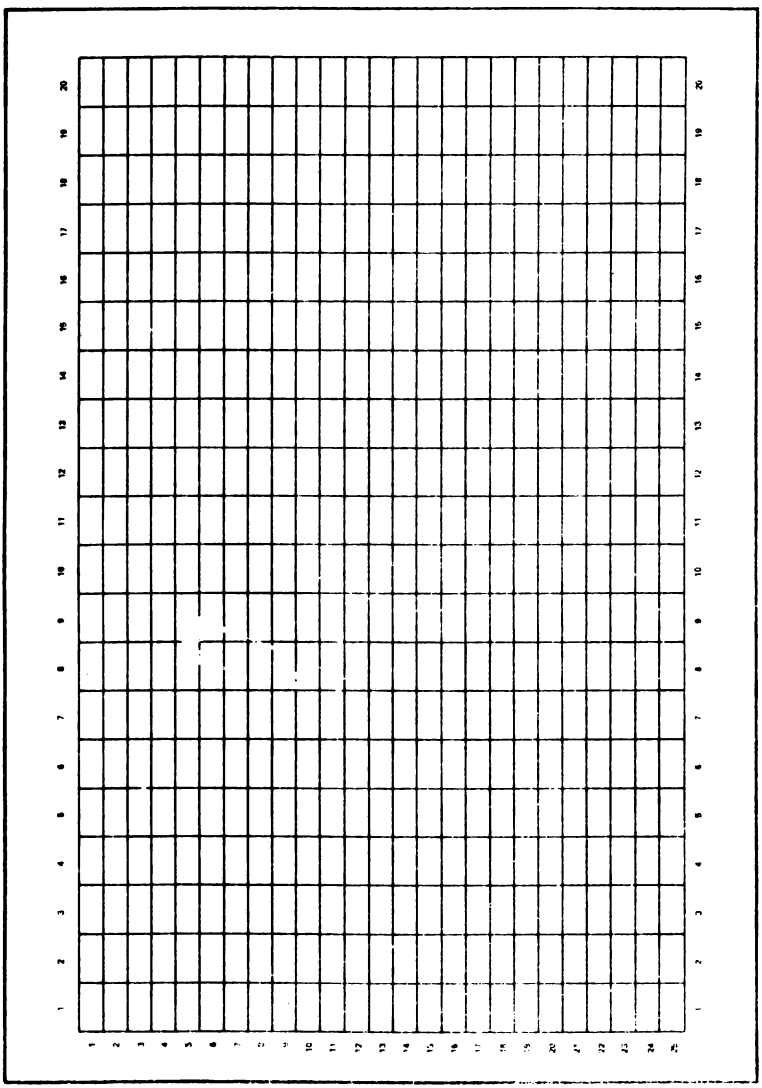
YPOS

ZONE

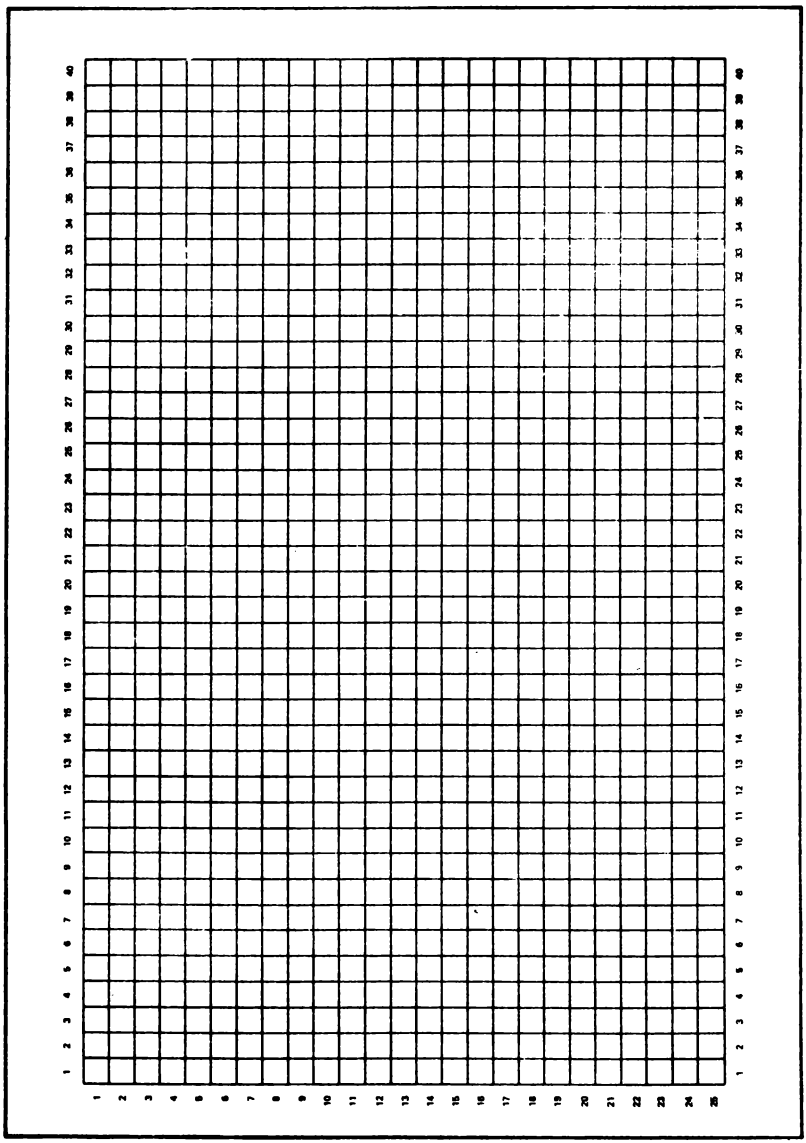
Część 8: Szablony

W części tej zamieszczono szablony, ułatwiające planowanie rozmieszczenia tekstu i okien w płaszczyźnie ekranu w trzech różnych trybach pracy /Text and window planner - MODE Ø, 1, 2/ oraz przebieg obwiedni dźwięku lub muzyki /Sound envelope/music planner/.

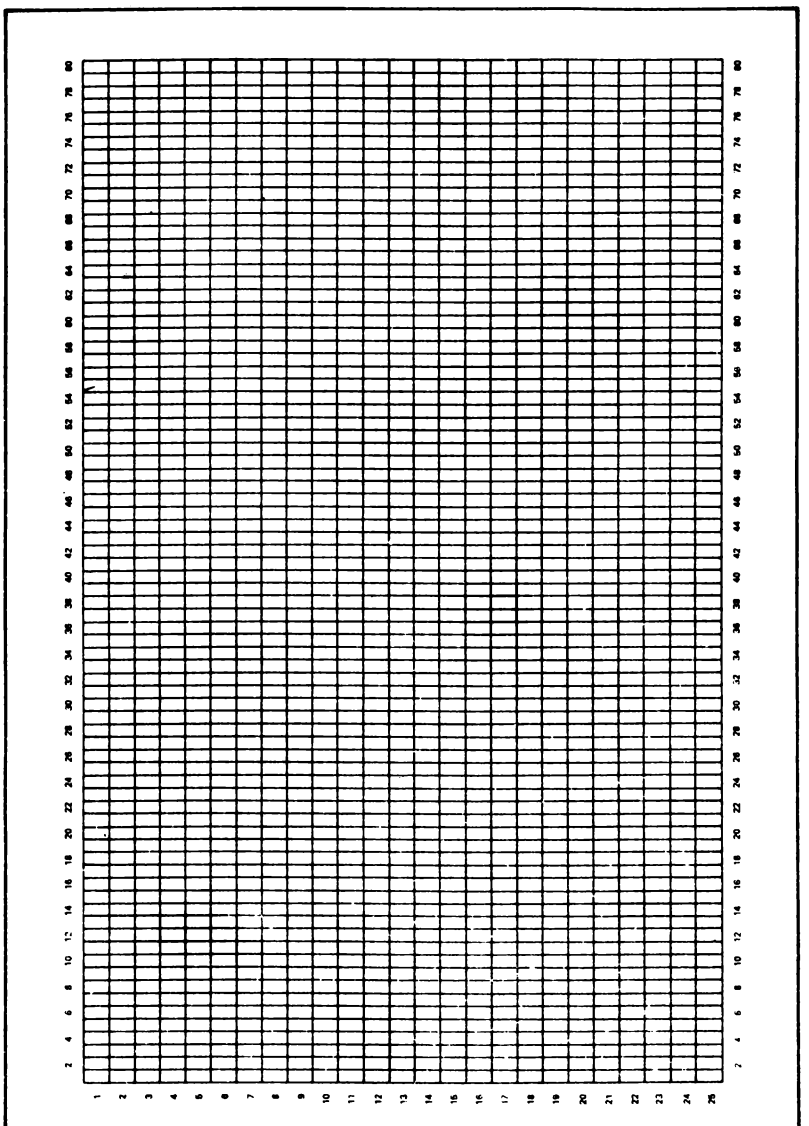
Text and window planner - MODE 0 (20 columns)



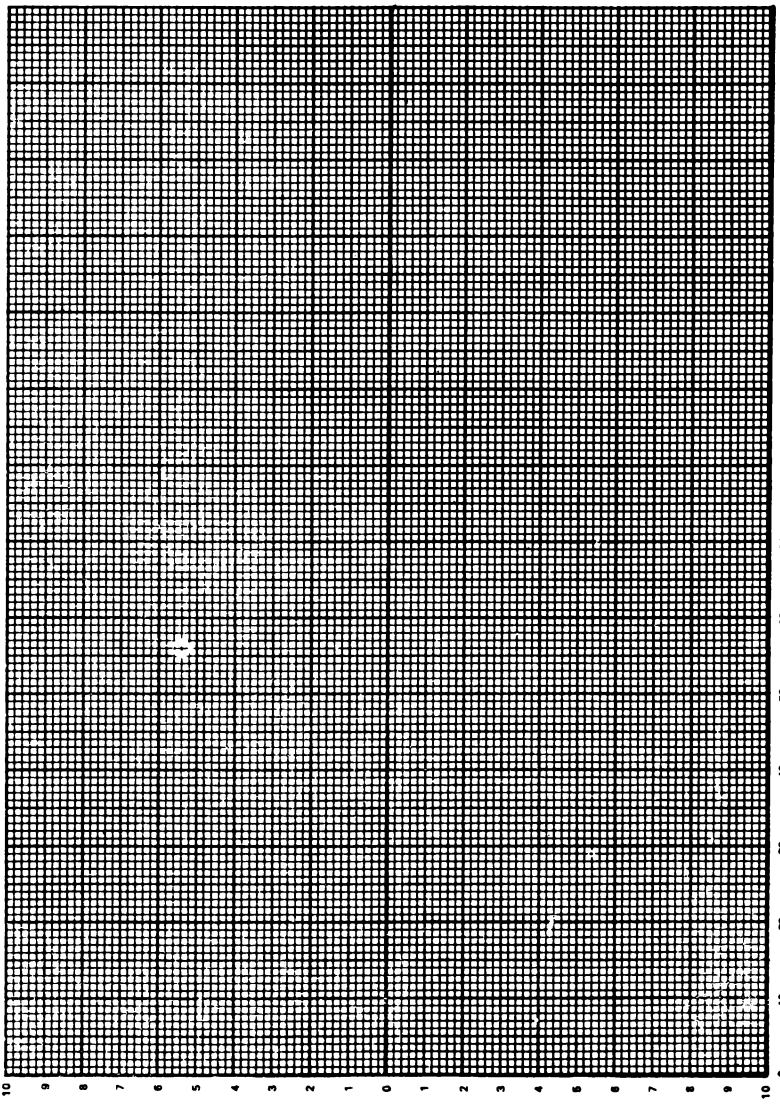
Text and window planner - MODE 1 (40 columns)



Text and window planner - MODE 2 (80 columns)



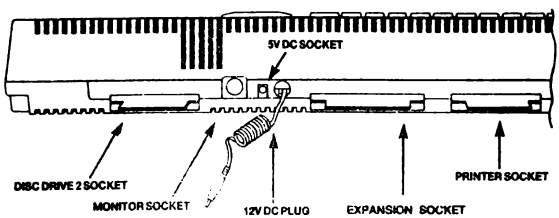
Sound envelope/music planner

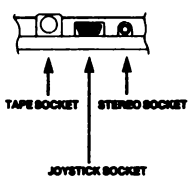


Część 9: Złącza

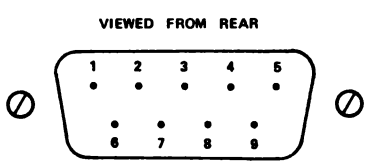
W części tej zamieszczono rysunki przedstawiające rozmieszczenie oraz opisy złącz, przeznaczonych do przyłączania urządzeń peryferyjnych.

CPC6128 Input/Output Sockets





Joystick Socket



PIN 1	UP	PIN 6	FIRE 2
PIN 2	DOWN	PIN 7	FIRE 1
PIN 3	LEFT	PIN 8	COMMON
PIN 4	RIGHT	PIN 9	COM 2
PIN 5	SPARE		

CPC6128 Input/Output Sockets

przedstawia rozmieszczenie gniazd wejściowych i wyjściowych:

5V DC Socket - gniazdo 5V napięcia stałego

12V DC Socket - gniazdo 12V napięcia stałego

Monitor Socket - gniazdo przyłączenia monitora

Disc Drive 2 Socket - złącza drugiego napędu dyskowego

Expansion Socket - złącze rozszerzające

Printer Socket - złącze drukarki

Tape Socket - gniazdo magnetofonowe

Stereo Socket - wyjście stereo

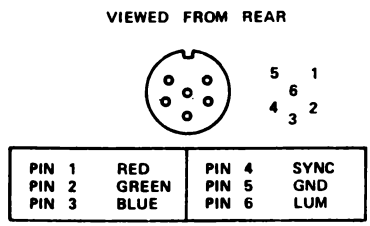
Joystick Socket - gniazdo joysticka

Joystick Socket

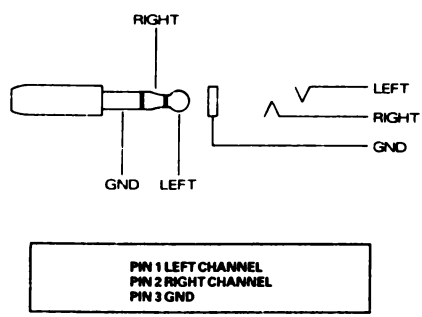
Gniazdo joysticka - widok z tyłu

1 w górę
2 w dół
3 w lewo
4 w prawo
5 zapasowy
6 spust 2
7 spust 1
8 wspólny
9 wspólny 2

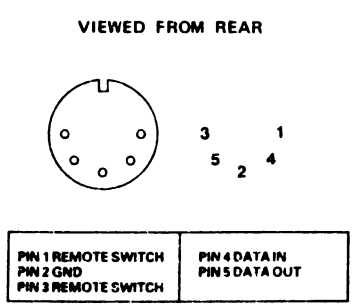
Monitor Socket



Stereo Socket



Tape Socket



Monitor Socket

Gniazdo monitora - widok z tyłu

1 czerwony	4 synchronizacja
2 zielony	5 masa
3 niebieski	6 luminacja

Stereo Socket

Złącze Stereo

LEFT - kanał lewy

RIGHT - kanał prawy

GND - masa

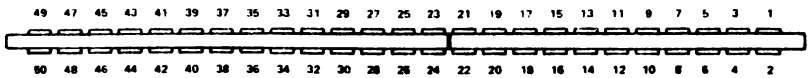
Tape Socket

Gniazdo magnetofonowe - widok z tyłu

1 zdalne włączanie	4 wejście
2 masa	5 wyjście
3 zdalne włączanie	

Expansion Socket

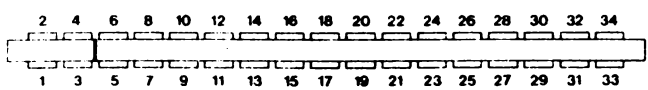
VIEWED FROM REAR



PIN 1 SOUND	PIN 18 A0	PIN 35 $\overline{INT}$
PIN 2 GND	PIN 19 D7	PIN 36 $\overline{NMI}$
PIN 3 A15	PIN 20 D6	PIN 37 $\overline{BUSR2}$
PIN 4 A14	PIN 21 D5	PIN 38 $\overline{BUSAK}$
PIN 5 A13	PIN 22 D4	PIN 39 READY
PIN 6 A12	PIN 23 D3	PIN 40 $\overline{BUS\ RESET}$
PIN 7 A11	PIN 24 D2	PIN 41 $\overline{RESET}$
PIN 8 A10	PIN 25 D1	PIN 42 $\overline{ROMEN}$
PIN 9 A9	PIN 26 D0	PIN 43 $\overline{ROMDIS}$
PIN 10 A8	PIN 27 + 5v	PIN 44 $\overline{RAMRD}$
PIN 11 A7	PIN 28 $\overline{MREQ}$	PIN 45 RAMDIS
PIN 12 A6	PIN 29 $\overline{M1}$	PIN 46 CURSOR
PIN 13 A5	PIN 30 $\overline{RFSH}$	PIN 47 L. PEN
PIN 14 A4	PIN 31 $\overline{IORQ}$	PIN 48 $\overline{EXP}$
PIN 15 A3	PIN 32 $\overline{RD}$	PIN 49 GND
PIN 16 A2	PIN 33 $\overline{WR}$	PIN 50
PIN 17 A1	PIN 34 $\overline{HALT}$	

Disc Drive 2 Socket

VIEWED FROM REAR



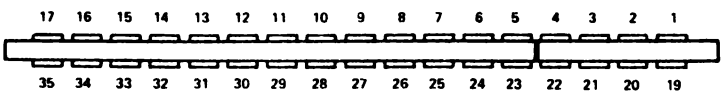
PIN 1 $\overline{READY}$	PIN 18 $\overline{GND}$
PIN 2 $\overline{GND}$	PIN 19 MOTOR ON
PIN 3 SIDE 1 SELECT	PIN 20 GND
PIN 4 $\overline{GND}$	PIN 21 N/C
PIN 5 READ DATA	PIN 22 $\overline{GND}$
PIN 6 $\overline{GND}$	PIN 23 DRIVE SELECT 1
PIN 7 WRITE PROTECT	PIN 24 GND
PIN 8 $\overline{GND}$	PIN 25 N/C
PIN 9 TRACK 0	PIN 26 $\overline{GND}$
PIN 10 $\overline{GND}$	PIN 27 INDEX
PIN 11 WRITE GATE	PIN 28 GND
PIN 12 $\overline{GND}$	PIN 29 N/C
PIN 13 WRITE DATA	PIN 30 GND
PIN 14 $\overline{GND}$	PIN 31 N/C
PIN 15 $\overline{STEP}$	PIN 32 $\overline{GND}$
PIN 16 $\overline{GND}$	PIN 33 N/C
PIN 17 DIRECTION SELECT	PIN 34 GND

Expansion Socket
Złącze rozszerzające - widok z tyłu

Disc Drive 2 Socket
Złącze drugiego napędu dyskowego - widok z tyłu

Printer Port

VIEWED FROM REAR



PIN	1	$\overline{\text{STROBE}}$	PIN	19	GND
PIN	2	D0	PIN	20	GND
PIN	3	D1	PIN	21	GND
PIN	4	D2	PIN	22	GND
PIN	5	D3	PIN	23	GND
PIN	6	D4	PIN	24	GND
PIN	7	D5	PIN	25	GND
PIN	8	D6	PIN	26	GND
PIN	9	GND	PIN	27	GND
PIN	11	BUSY	PIN	28	GND
PIN	14	GND	PIN	33	GND
PIN	16	GND	All other pins		NC

Printer Port
Złącze drukarki - widok z tyłu
/łączówki nie opisane - puste/

Część 10: Drukarki

Dołączanie drukarki

Do CPC 6128 może być dołączona standardowa drukarka z łączem typu "Centronics".

Kabel przyłączeniowy wykonać można przez proste połączenie łącsówek o tych samych numerach złącsa PRINTER (DRUKARKA) z tyłu komputera i złącsa wejściowego drukarki. Złącze umieszczone w komputerze ma o dwie końcówki mniej, niż złącze drukarki, gdyż przesnaczone jest do standardowego złącsa krawędziowego do płytek drukowanych.

Rozmieszczenie końcówek złącsa szczegółowo przedstawiono w części 9 tego rozdziału.

Kabel należy wykonać tak, aby końcówka 1 złącza komputera była połączona s końcówką 1 złącsa drukarki, itd. Końcówki 18 i 36 złącsa drukarki powinny pozostać NIE połączone.

Proszę zauważyć, że chociaż górny rząd złącsa PRINTER (DRUKARKA) komputera zawiera tylko 17 końcówek, dolne końcówki numerowane są od numeru 19 (a nie 18). Numerację taką wprowadzono w tym celu aby wszystkie druty w kablu przyłączeniowym drukarki łączyły końcówki o dokładnie TYCH SAMYCH NUMERACH złącsa krawędziowego komputera i gniazda drukarki.

Komputer używa sygnału BUSY (zajęty - końcówka 11) do synchronizacji z drukarką i czeka, jeżeli drukarka nie jest gotowa do współpracy.

Skierowanie informacji wychodzącej z komputera do drukarki nie wymaga używania żadnego specjalnego zbioru dyrektyw, wystarczy jedynie określić strumień logiczny # 8.

Chociaż wyjście PRINTER (DRUKARKA) komputera CPC 6128 przeznaczone jest przede wszystkim do dołączenia niesbyt drogich drukarek mozaikowych, to przy użyciu odpowiedniego interfejsu możliwa jest także współpraca z drukarkami rozetkowymi, drukarkami (ploterami) graficznymi i wielokolorowymi drukarkami strumieniowymi. Kluczem do kompatybilności jest standardowe łącze równoległe.

Odpowiednie oprogramowanie, zaimplementowane w jednostce drukującej AMSTRAD DMP1 ułatwia operowanie grafiką punktową oraz drukowanie kompletnego obrazu ekranu.

Konfiguracja drukarki

Oprogramowanie drukarki AMSTRAD DMP1 umożliwia wyświetlanie znaków specjalnych na ekranie i drukowanie ich przez drukarkę nawet wtedy, gdy kody znaków, przeznaczonych do wyświetlania na ekranie różnią się od kodów, jakie należałoby wysłać do drukarki. Większość takich znaków może być drukowana tylko wtedy, gdy drukarka zostanie przełączona do pracy w wybranym języku obcym. Na przykład:

```
PRINT CHR$ (&A0)
      ^
PRINT #8, CHR$ (&A0)
      ^
```

powoduje wyświetlenie na ekranie i wydrukowanie przez drukarkę znaku akcentu, używanego w języku francuskim. Drukarka drukuje nakazany znak pomimo, iż kod tego znaku jest w DMP1 równy &5E. Oprogramowanie drukarki rozpoznaje kod A0 jako jeden z kodów, umieszczonych w tablicy tłumaczącej znaki specjalne i kod ten jest zamieniony na &5E tak, że drukowany jest taki sam znak, jaki wyświetlany jest na ekranie. Kod &5E powoduje drukowanie przez DMP1 znaku akcentu, bez względu na to, jaki język wybrano jako język pracy drukarki. Inne znaki specjalne są tłumaczone zgodnie z poniższą tablicą:

CHR$	Znak na ekranie	Tłumaczenie w drukarce	angielski W.Bryt.	angielski USA	francuski	niemiecki	hiszpański
&A0	^	&5E	^	^	^	^	^
&A2	¨	&7B	+	+	+	+	¨
&A3	£	&23	£	#	#	#	Pt
&A6	§	&40	+	+	+	§	+
&AE	¿	&5D	+	+	+	+	¿
&AF	¡	&5B	+	+	+	+	¡

\+ Znak drukowany zgodnie z opisem, zamieszczonym na str. instrukcji obsługi drukarki DMP1

Przedstawione powyżej tłumaczenie znaków przyjmowane jest domyślnie i może być zmienione zgodnie z życzeniami użytkownika. Szczegóły podaje instrukcja firmowa (SOFT 968).

Część 11: Joysticki

Oprogramowanie systemowe komputera umożliwia posługiwanie się jednym lub dwoma joystickami. Joysticki są traktowane jako część klawiatury i ich stan może być badany za pomocą instrukcji INKEY i INKEY$.

Należy zauważyć, że w większości przypadków jako główny "język spustowy" joysticka jest przyjmowany przez 6128 "spust" ("fire") 2.

Funkcje JOY (0) i YOY (1) umożliwiają bezpośrednie badanie stanu pierwszego lub drugiego joysticka. Funkcja podaje wartość, której poszczególne bity opisują stan joysticka podczas ostatniego czytania klawiatury.

Wartości funkcji JOY dla obu joysticków podano w poniższej tablicy. Podano tam także wartości, jakie powinny być używane w rozkazach, które wymagają jako parametr numer klawisza (tj. INKEY i KEY DEF).

STAN JOY-STICK-KA	FUNKCJA JOY		NUMER KLAWISZA		
	Ustawiony bit	Wartość funkcji	Joystick 1	Joystick 2	Odpowiadający klawisz
Góra	0	1	72	48	6
Dół	1	2	73	49	5
Lewo	2	4	74	50	R
Prawo	3	8	75	51	T
Spust 1	4	16	76	52	G
Spust 2	5	32	77	53	F

Należy zauważyć, że w przypadku otrzymania numeru klawisza, odpowiadającego DRUGIEMU joystickowi nie można odróżnić, czy wartość ta wynika z działania joysticka czy odpowiedniego klawisza klawiatury, podanego w ostatniej kolumnie tablicy. Oznacza to, że klawiatura może być używana jako zamiennik drugiego joysticka.

Część 12: Organizacja dysku

BIOS umożliwia obsługę dysków o trzech różnych formatach: format SYSTEMowy, format TYLKO DANE (DATA ONLY) i format IBM. W systemie AMSDOS format dysku jest automatycznie rozpoznawany w czasie każdego dostępu do dysku z nie otwartymi zbiorami. Dla umożliwienia automatycznego rozpoznawania formatu, w każdym formacie stosowana jest odmienna numeracja sektorów.

Dyski 3-calowe są dyskami dwustronnymi, ale tylko jedna strona jest używana w danym momencie, zależnie od tego w którą stronę obrócił dysk użytkownik w czasie wkładania dysku. Każda strona dysku może być inaczej sformatowana.

Wspólne dla wszystkich formatów:

- Dysk jednostronny (każda strona dysku 3-calowego jest traktowana jako osobny dysk)
- Fizyczny rozmiar sektora równy 512 bajtów
- 40 ścieżek numerowanych od 0 do 39
- Blok CP/M o wielkości 1024 bajty
- 64 pozycje skorowidzu (directory)

Format SYSTEMowy

- 9 sektorów na ścieżce, numerowanych od 841 do 849
- 2 ścieżki zarezerwowane

Format systemowy jest formatem podstawowym, ponieważ CP/M może być wprowadzony tylko z dysku o takim formacie. W systemie CP/M 2.2 wymagane jest także umieszczenie dysku w napędzie przy każdej reinicjalizacji systemu. Ścieżki zarezerwowane są używane w następujący sposób:

	CP/M 2.2	CP/M Plus
Ścieżka 0 sektor 41	program wprowadzający CP/M 2.2	program wprowadzający CP/M Plus
Ścieżka 0 sektor 42	sektor opisujący konfigurację sprzętu	nie używane
Ścieżka 0, sektory 43 do 47	nie używane	
Ścieżka 0, sektory 48 do 49	CCP i BDOS	
Ścieżka 1, sektory 41 do 49		

Format SPRZEDAŻNY (VENDOR format) jest specjalną odmianą formatu systemowego, w której nie umieszczono żadnego oprogramowania systemowego na ścieżkach 0 i 1. Format ten jest przeznaczony do użycia przy dystrybucji oprogramowania.

Format TYLKO DANE (DATA ONLY)

- 9 sektorów na ścieżce, numerowanych od C1 do C9
- 0 zarezerwowanych ścieżek

Format nie zalecany do stosowania w systemie CP/M 2.2, gdyż nie umożliwia reinicjalizacji systemu. Jednakże gdy używa się tylko CP/M Plus lub AMSDOS, format taki oferuje nieco więcej miejsca na dysku.

Format IBM (tylko w systemie CP/M 2.2)

- 8 sektorów na ścieżce numerowanych od 1 do 8
- 1 zarezerwowana ścieżka

Format ten jest logicznie taki sam, jak format jednostronny używany przez CP/M w komputerach IBM PC. CPV 6128 może czytać i zapisywać dyski o formacie IBM, nie może jednakże formatować lub kopiować takich dysków.

Część 13: Rozszerzenia systemu rezydentnego (Resident System eXtensions - RSXs)

Rozkazy zewnętrzne zostały wprowadzone w rozdziale 5 (o systemie AMSDOS). Ogólnie, rozkazy takie dają możliwość rozszerzenia repertuaru rozkazów języka BASIC przez dodanie nowych rozkazów, poprzedzonych znakiem | . Procedury realizujące zestaw rozkazów systemu AMSDOS są umieszczone w pamięci ROM i wszelkie niezbędne zabiegi, umożliwiające używanie tych rozkazów są wykonywane automatycznie w czasie inicjalizacji pracy 6128 w języku BASIC.

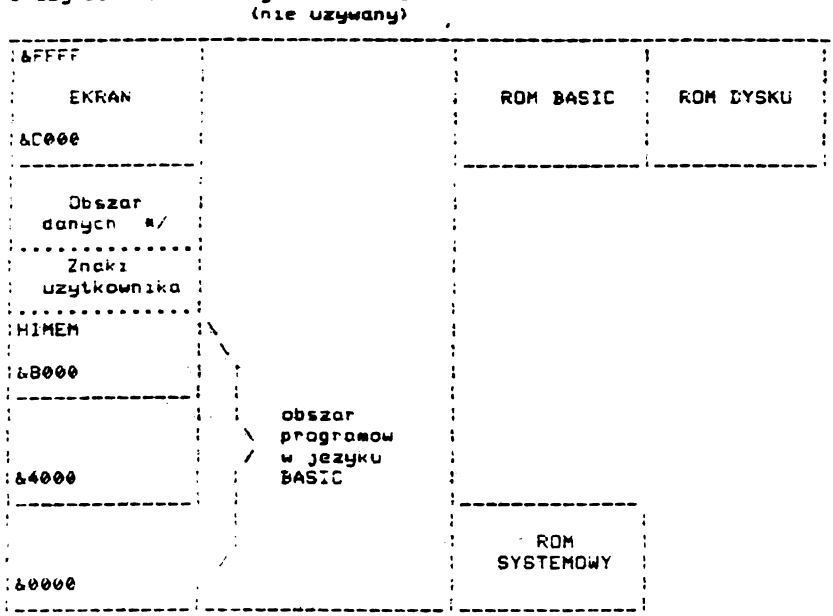
Jest także możliwe dodanie dalszych rozkazów zewnętrznych w czasie pracy w języku BASIC przez umieszczenie odpowiednich procedur maszynowych w pamięci RAM. Rozkazy takie są określane mianem "RSX" i mogą być używane dokładnie tak samo, jak rozkazy

zewnętrzne realizowane przez procedury umieszczone w pamięci ROM. Procedury RSX muszą być wprowadzane z dysku /lub magnetofonu/ za każdym razem, gdy inicjalizuje się /lub reinicjalizuje/ pracę 6128 w języku BASIC. RSX są stosowane przede wszystkim do obsługi niektórych typów urządzeń peryferyjnych jak np. pióro świetlne czy syntetyzer mowy.

W rozdziale 8 opisano użycie RSX, umożliwiającego korzystanie z drugiego banku pamięci 64 KB komputera 6128.

Część 14: Pamięć

CPC 6128 zawiera 128 KB pamięci RAM i 48 KB pamięci ROM. Pamięć ta jest wykorzystywana przez BASIC 1.1 w przedstawiony ppniżej sposób. Pierwsze 64 KB pamięci RAM jest nominalnie podzielone na cztery bloki po 16 KB, oznaczone jako Blok 0 do Blok 3. Blok 3 używany jest przez ekran a górna część Bloku 2 jest wypełniona przez zmienne systemowe jak pokazano na rysunku.



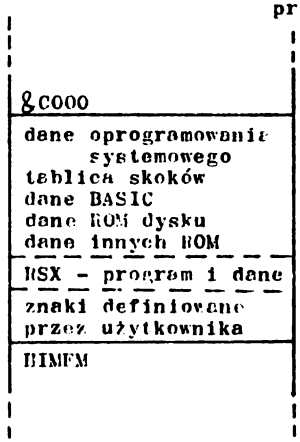
*/ Dane oprogramowania systemowego, blok skokow, dane BASIC dane ROM DYSKU, dane innych ROM.
Wielkosc tego obszaru zalezy od liczby dodanych zewnetrznie ROM - &A6FC bez zewnetrznych ROM.

Znaki definiowane przez użytkownika są początkowo umieszczone bezpośrednio powyżej HIMEM. HIMEM może być zmienione przez rozkaz MEMORY a także jest automatycznie obniżane o 4 KB w celu utworzenia bufora w przypadku otwarcia zbiorów przez AMSDOS. Liczba znaków definiowanych przez użytkownika może być zmieniona tylko wtedy, gdy HIMEM pozostało niezmienione od chwili ich zdefiniowania (chyba, że ostatnio zadeklarowano "brak znaków definiowanych przez użytkownika" za pomocą polecenia SYMBOL AFTER 256). Przy inicjalizacji języka BASIC znaki definiowane przez użytkownika są ustawiane w taki sam sposób, jak po rozkazie SYMBOL AFTER 240.

Jest zatem wskazane, by zadeklarować brak znaków definiowanych przez użytkownika przed zmianą HIMEM a następnie ustabilizować te znaki w nowej pozycji. Umożliwia to w wykonywanych później programach zmieniać deklarację SYMBOL AFTER.

Poniższy przykład pokazuje taki sposób postępowania gdy HIMEM jest obniżane w wzwiązku z wprowadzaniem RSX:

```
1ØØ  SYMBOL AFTER 256  'deklaracja braku znaków definiowanych
                        przez użytkownika
11Ø  rsxaddress = HIMEM - rsxlength
12Ø  MEMORY rsxaddress - 1
13Ø  LOAD "rsxcode", rsxaddress
14Ø  CALL rsxaddress   'inicjalizacja RSX
15Ø  SYMBOL AFTER 14Ø  'odtworzenie znaków definiowanych
                        przez użytkownika
```



Mapa Bloku 1
po wprowadzeniu RSX

Dodatkowe układy wejścia/wyjścia

Większość adresów urządzeń wejścia/wyjścia jest zarezerwowana przez system, w szczególności - nie mogą być używane adresy poniżej &7FFF.

Zaleca się, aby część A0 - A7 adresu odzwierciedlała typ zewnętrznego urządzenia wejścia/wyjścia a linie A8 i A9 mogą być dekodowane w celu selekcjonowania wewnętrznych rejestrów urządzenia. Z pozostałych linii adresowych należy dekodować tylko A10: stan niski i A11 do A15: stan wysoki. W ten sposób każde urządzenie może mieć rejestry adresowane jako &F8??, &F9??, &FA?? i &FB?? gdzie ?? ma wartość w zakresie DC do DF dla urządzeń sprzęgów komunikacyjnych i wartość od E0 do FE dla innych urządzeń peryferyjnych użytkownika.

Do adresowania należy odpowiednio używać instrukcji Z80, które powodują wysłanie zawartości rejestru B na górną połówkę magistrali adresowej (A15-A8).

Dodatkowe (boczne) pamięci ROM

Przewidziano możliwość wybierania dodatkowych zewnętrznych pamięci ROM zamiast dowolnej części takiej pamięci, umieszczonej wewnątrz komputera. Odpowiedni układ wyboru pamięci należy umieścić w module, dołączanym do złącza rozszerzającego, gdzie wyprowadzone są wszystkie wymagane przy tym sygnały.

Część 15: Emulator terminala CP/M Plus

W części 1 tego rozdziału zamieszczono tablicę znaków kontrolnych i opisano ich działanie. Działania takie mają miejsce przy wysyłaniu tekstu na ekran przez BASIC lub CP/M 2.2 i wybrane zostały zarówno ze względu na łatwość użycia jak i możliwości procedur obsługi ekranu oprogramowania systemowego. Działania takie są specyficzne dla komputerów firmy AMSTRAD i wszystkie programy muszą być dostosowane do ich użycia.

W profesjonalnym i handlowym otoczeniu programowym systemu CP/M Plus normalnie oczekuje się określonego zakresu "standardo-

wych" możliwości procedur edycji i wizualizacji tekstu aby programy były łatwe do przenoszenia i możliwe do zainstalowania w różnych typach komputerów. Implementacja CP/M Plus w 6128 zawiera emulator terminala o możliwościach bardzo podobnych do monitora alfanumerycznego typu Zenith Z19/Z29. Procedury instalacyjne programów przeznaczonych do stosowania w systemie CP/M Plus powinny zawierać standardową opcję dla tego typu terminala.

Możliwości oferowane przez emulator terminala systemu CP/M Plus zawierają wiele z poprzednio opisanych możliwości procedur edycji tekstu oprogramowania firmowego chociaż wymagane są odmienne kody kontrolne. Ponadto występuje tutaj znaczna liczba nowych i bardziej wymyślnych operacji.

Znaki z zakresu od &20 do &FF wyświetlane są w miejscu aktualnego położenia kursora. Gdy kursor nie jest w skrajnie prawej kolumnie, przesuwa się w prawo o jedną pozycję. Jeżeli kursor jest w skrajnym prawym położeniu i dozwolone jest przenoszenie, przesuwa się do skrajnie lewej kolumny następnej linii, w razie potrzeby rolując w górę zawartość ekranu.

Znaki z zakresu &00 do &1F są interpretowane jako znaki kontrolne w następujący sposób:

&07	BEL	(Dzwonek).Krótki sygnał dźwiękowy
&08	BS	(Backspace). Przesuwa w lewo o jedną kolumnę. Jeżeli kursor jest w skrajnie lewej kolumnie w nienajwyższej linii i dozwolone jest przenoszenie, przesuwa się do prawej kolumny wyższej linii
&0A	LF	(Linefeed). Przesuwa kursor o jedną linię w dół, w razie potrzeby rolując w górę zawartość ekranu
&0D	CR	(Cariage return). Przesuwa kursor do lewego skraju bieżącej linii
&1B	ESC	(Escape). Zmienia znaczenie następnego lub następnych znaków

Rozpoznawane są podane niżej znaki lub sekwencje znaków umieszczone za znakiem ESC . Inne znaki wprowadzone po ESC wyświetlane są w pozycji kursora, który przesuwa się do przodu. Ta właściwość może być wykorzystane do wyświetlania znaków

odpowiadających kodom kontrolnym z zakresu 00 do 1F. Należy jednak zauważyć, że w wielu językach aplikacyjnych kod kontrolny &09 (TAB) powoduje wprowadzenie określonej liczby spacji (tabulację) i sekwencja [ESC][TAB] często nie wyświetla znaku o kodzie &09.

[ESC] 0 Wyłącza linię systemową. Komunikaty systemu dyskowego wyświetlane są w pozycji kursora. Można normalnie wykorzystywać do wyświetlania tekstu najniższą (25) linię ekranu

[ESC] 1 Włącza linię systemową. Komunikaty systemu dyskowego wyświetlane są w najniższej linii ekranu

[ESC] 2 ⟨n⟩ Zmienia zestaw znaków (patrz część 16 tego rozdziału). ⟨n⟩ jest parametrem określającym język, maskowanym przez &07. Niektóre matryce znakowe z zakresu &20 do &7F są wymieniane z innymi znakami z zakresu &80 do &FF. Działanie takie jest podobne do stosowanego w drukarkach, które mają programowo wybierane zestawy znaków kilku różnych języków

⟨n⟩ = 0	USA	⟨n⟩ = 4	Dania
⟨n⟩ = 1	Francja	⟨n⟩ = 5	Szwecja
⟨n⟩ = 2	Niemcy	⟨n⟩ = 6	Włochy
⟨n⟩ = 3	Wielka Brytania	⟨n⟩ = 7	Hiszpania

[ESC] 3 ⟨m⟩ Zmienia tryb pracy ekranu ⟨m⟩ = tryb pracy ekranu +&20. Wartość jest maskowana przez &3 tak, aby określić tryb pracy w zakresie od 0 do 2. Tryb 3 jest ignorowany. Usuwana jest zawartość ekranu, ale pozycja kursora pozostaje niezmieniona

[ESC] A Kursor w górę. W najwyższej linii nie działa.

[ESC] B Kursor w dół. W najniższej linii nie działa.

[ESC] C Kursor w przód. W skrajnie prawej kolumnie nie działa.

[ESC] D Kursor w tył. W skrajnie lewej kolumnie nie działa

[ESC] E Kasuje stronę. Pozycja kursora nie zmienia się. Kasuje cały ekran, nawet gdy ustawiony tryb 24x80. (Inne sekwencje ESC działają tylko w obszarze 24x80 gdy ustawiony jest taki tryb wyświetlania)

[ESC] H Kursor w położenie początkowe w lewym górnym rogu ekranu

[ESC] I Kasuje do końca strony od znaku w pozycji kursora włącznie. Pozycja kursora pozostaje niezmieniona

[ESC] K Kasuje do końca linii od znaku w pozycji kursora włącznie. Pozycja kursora nie zmieniana.

[ESC] L Dołącza linię. Linia z kursorem i wszystkie linie niżej są rolowane w dół. W dawnej pozycji linii z kursorem wprowadzona zostaje pusta linia. Pozycja kursora nie zmieniana.

[ESC] M Usuwa linię. Linia z kursorem i wszystkie linie niżej są rolowane w górę. Linia najniższa pozostaje pusta. Nie zmienia pozycji kursora.

[ESC] N Usuwa znak. Wszystkie znaki na prawo od kursora są przesuwane w lewo o jedną pozycję. Znak na końcu linii jest kasowany. Pozycja kursora nie zmienia się

[ESC] Y ⟨r⟩⟨c⟩ Przesuwa kursor do zadanej pozycji. Jeżeli określono pozycję poza zakresem ekranu, kursor jest przesuwany do skraju ekranu. ⟨r⟩ = wiersz (row) + &20, ⟨c⟩= kolumna (column) + &20. Górny lewy róg ekranu to wiersz 0 kolumna 0

[ESC] b ⟨cp⟩ Ustawia kolor znaków. Oddziaływuje na wszystkie znaki na ekranie ⟨cp⟩ jest parametrem określającym kolor; jego wartość jest maskowana przez &3F i następnie traktowana jako zbiór trzech liczb 2-bitowych z których każda określa intensywność jednego z trzech kolorów podstawowych: niebieskiebo (bity 0,1), czerwonego (bity 2,3) i zielonego (bity 4,5). W 6128 rozróżniane są trzy poziomy intensywności określane przez cztery możliwe do wyspecyfikowania wielkości następująco:

CPC 6128	Intensywność zerowa	Pół intensywności	Pełna intensywność
Bity kolorów	00 binarnie	01 lub 10 binarnie	11 binarnie

[ESC] c <cp> Ustawia kolor tła. Oddziaływuje na tło i obramowanie ekranu. Kolory są określane jak wyżej

[FSC] d Kasuje od początku strony do znaku na pozycji kursora łącznie z tym znakiem. Nie zmienia pozycji kursora

[ESC] e Włącza kursor. Aby zapobiec brzydkiemu błyskaniu, kursor nie jest włączany podczas normalnego wysyłania tekstu lecz 1/10 sekundy po wpisaniu ostatniego znaku

[ESC] f Wyłącza kursor

[ESC] j Zapamiętuje pozycję kursora

[FSC] k Odtwarza pozycję kursora zapamiętaną po ESC j

[FSC] l Kasuje linię. Nie zmienia pozycji kursora

[ESC] o Kasuje linię od początku do znaku na pozycji kursora łącznie z tym znakiem. Nie zmienia pozycji kursora

[ESC] p Włącza tryb wyświetlania negatywowego. Wymienia wzajemnie kolory wyświetlania znaków i tła.

[ESC] q Wyłącza tryb wyświetlania negatywowego

[FSC] r Włącza tryb pracy z linią podkreślającą (nie wprowadzone w CPC 6128)

[FSC] u Wyłącza tryb pracy z linią podkreślającą (nie wprowadzone w CPC 6128)

[FSC] v Przenoszenie na końcu linii

[ESC] w Obcinanie na końcu linii

[ESC] x Włącza tryb 24x80. Niektóre programy aplikacyjne mogą wymagać takiego "standardowego" formatu ekranu. Rozkaz ten powinien zezwolić na pracę z takim ekranem bez względu na całkowite wymiary ekranu, które mogą zależeć od sprzętu i kraju oraz bez względu na to, czy linia systemowa jest włączona czy nie. Treść ekranu jest kasowana

[ESC] y Wyłącza tryb 24x80. Treść ekranu jest kasowana

Część 16: Zbiór znaków w systemie CP/M Plus

W części 10 tego rozdziału opisano tablicę zmieniającą znaki drukarki. Zadaniem realizowanym przy użyciu takiej tablicy jest zmiana niektórych znaków ze zbioru znaków języka BASIC lub systemu CP/M 2.2 tak, aby mogły być drukowane przez drukarkę przystosowaną do drukowania w różnych językach alfabetu. Możliwość ta jest nieco ograniczona ze względu na to, że bardzo mało specyficznych obcojęzycznych znaków, które mogę być drukowane przez drukarkę występuje w zbiorze znaków języka BASIC.

Chociaż opisana procedura zmiany znaków przez drukarkę realizowana jest także w systemie CP/M Plus zbiór znaków tego systemu został rozszerzony tak, aby zapewnić prawie całkowitą zgodność między znakami wyświetlanymi na ekranie a znakami drukowanymi przez drukarkę /jednym symbolem nie wyświetlanym na ekranie jest znak waluty szwedzkiej, zastąpiony przez znak S/. Tablica "CP/M Plus International Character Set" /zbiór znaków obcojęzycznych systemu CP/M Plus/ zamieszczona poniżej potwierdza takie uporządkowanie.

	23	24	40	5B	5C	5D	5E	60	7B	7C	7D	7E
USA	#	$	@	[	\	]	↑	`	{	|	}	~
France	#	$	à	°	ç	§	↑	`	é	ù	è	¨
Germany	#	$	§	Ä	Ö	Ü	↑	`	ä	ö	ü	ß
UK	£	$	@	[	\	]	↑	`	{	|	}	~
Denmark	#	$	@	Æ	Ø	Å	↑	`	æ	ø	å	~
Sweden	#	$	É	Ä	Ö	Å	Ü	é	ä	ö	å	ü
Italy	#	$	@	°	\	é	↑	ù	à	ò	è	ì
Spain	Pt	$	@	¡	Ñ	¿	↑	`	¨	ñ	}	~

CP/M Plus International Character Set

Przystosowanie sprzętu do pracy w języku obcym wymaga dwóch operacji:

1. Przystosowania drukarki do pracy w żądanym języku /często za pomocą odpowiedniego przełącznika, chociaż niektóre drukarki dopuszczają wysłanie odpowiednich kodów kontrolnych/

2. Wprowadzenia żądanego zbioru znaków ekranu bądź za pomocą polecenia nierezydentnego:

LANGUAGE n

bądź przez wysłanie

ESC 2 n

do emulatora terminala.

W praktyce inicjalizacja taka może być przeprowadzona jako część operacji PROFILE.SUB przy użyciu poleceń LANGUAGE i SETLIST. CP/M Plus jest dostarczany przez wytwórcę w podstawowej wersji anglojęzycznej amerykańskiej /różni się ona od wersji brytyjskiej wyświetlaniem znaku po użyciu klawiszy SHIFT 3/. Do opracowywania tekstów w innych językach korzystne jest odpowiednie dostosowanie sprzętu.

Oprogramowanie 7-bitowe ...

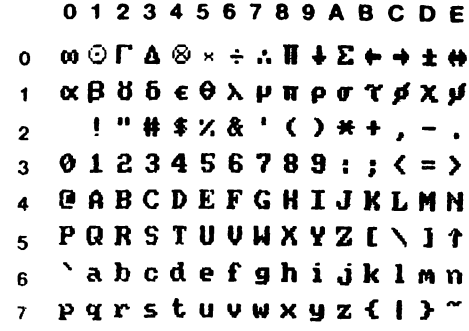
Chociaż możliwość pracy w językach obcych jest bardzo użyteczna, niezbyt korzystne jest przy tym to, że "normalne" (czyli "amerykańskie") znaki, które są zastąpione przez znaki obcojęzyczne nie mogą być nadal wyświetlane. Jest to zwykły i konieczny kompromis przy posługiwaniu się oprogramowaniem 7-bitowym. Prawie całe dostępne oprogramowanie (włącznie z większością programów użytkowych CP/M Plus, procesorów tekstu i języków) operuje tylko 7-bitowym zbiorem znaków. W Wielkiej Brytanii może być stosunkowo łatwo akceptowane, że znak # znika zastąpiony przez znak £ nie tylko przy posługiwaniu się procesorami tekstu (gdzie jest to uważane za pożądane) lecz także przy listowaniu programu, np. LIST #8, gdzie jest to niepożądane. Jednakże, w razie potrzeby każdy może sobie łatwo uzmysłowić, o co chodzi.

Niefortunnie, w innych językach obcych zastępowane są także takie znaki jak pionowa kreska i nawiasy kwadratowe i chociaż stosowanie specyficznych znaków obcojęzycznych zwiększa czytelność tekstu, to w sytuacjach, w których programy użytkowe wymagają kresek i nawiasów, jak np. DIR [FULL], czytelność i (w przypadku zmiany główek z opisami klawiszy) łatwość wprowadzania znaków jest znacząco obniżona. Należy pamiętać, że programy aplikacyjne wykorzystują wartości znaków ASCII, niezależnie od tego jakie kształty mają odpowiadające im znaki wyświetlane na ekranie. Mankamentem 7-bitowego oprogramowania jest po prostu zbyt mało różnych wartości w 7-bitowym kodzie ASCII.

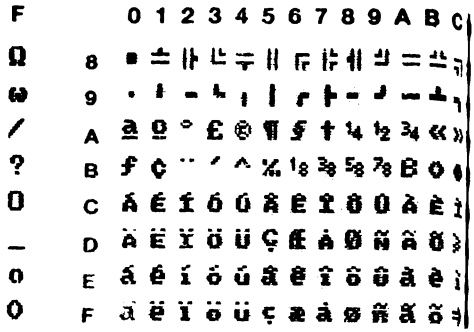
Praca ze zbiorem znaków 8-bitowych ...

Zbiór znaków języka BASIC ma 256 różnych symboli z wartościami 128 do 255 (&80 do &FF) zawierającymi różne symbole graficzne, używane głównie w grach i zastosowaniach domowych (tańczące ludziki, kiery/piki/kara/trefle itp.). CP/M Plus także dysponuje zestawem 256 znaków, lecz drugie 128 znaków są inne niż w języku BASIC i odpowiadają międzynarodowemu i profesjonalnemu charakterowi tego systemu. Poniżej

zamieszczono pełną tabelę tego zestawu znaków /The Standard CP/M Plus Character Set /USA/ - standardowy zbiór znaków systemu CP/M Plus /USA//.



Characters 0 to 127 (&00 to &7F)



Characters 128 to 255 (&80 to

The Standard CP/M Plus Character Set (USA)

Programy przystosowane do operowania znakami 8-bitowymi mogą używać ten zestaw znaków, umożliwiający stosowanie wszystkich przewidzianych w nim znaków obcojęzycznych jednocześnie, bez konieczności wprowadzania zmiany języka. Ponadto dostępne są jednocześnie wszystkie znaki typu "kresek i nawiasów".

Proszę pamiętać, że aktualnie bardzo mało jest dostępnych programów 8-bitowych i znaki z zakresu 0 do 31 i 128 do 255 będą pojawiać się w postaci przedstawionej w tablicach tylko na ekranie. Drukarki, z drugiej strony, mogą mieć swoje własne i odmienne koncepcje, w jaki sposób powinny wyglądać takie znaki.

ROZDZIAŁ 8

WIĘCEJ O PROGRAMIE **BANK MENAGER**

Dodatkowe oprogramowanie umożliwia dostęp do drugiego banku 64 KB pamięci RAM w języku BASIC.

Omawiane zagadnienia:

- przechowywanie obrazów ekranu

- działanie z pseudo-zbiorami

Mapa wykorzystania pamięci w języku BASIC 1.1 (zamieszczona w Części 14 Rozdziału 7) pokazuje, że 64 KB ze 128 KB pamięci RAM jest nie używane. BASIC i firmowe oprogramowanie systemowe rezydują w pamięci ROM, co łącznie z ROM dyskowym rozszerza używaną pamięć z 64 KB do 112 KB (64 KB RAM i 48 KB ROM).

Każda sekcja 16 KB jest nazywana "blokiem" a jakakolwiek konkretna konfiguracja czterech bloków, tworzących łącznie 64 KB, jest nazywana "bankiem". Technika selekcji bloków określana jest mianem "przełączania banku".

Mikroprocesor Z80 może adresować bezpośrednio tylko 64 KB pamięci i system operacyjny zawiera instrukcje włączające ROM z firmowym oprogramowaniem systemowym zamiast Bloku 0 pamięci RAM oraz włączające ROM z programem BASIC lub ROM dyskowy w miejsce Bloku 3 pamięci RAM w celu umożliwienia pracy z tymi pamięciami ROM. Przełączanie to następuje automatycznie w przypadku, gdy wymagany jest ROM z programem BASIC lub firmowym oprogramowaniem systemowym. Przełączanie banku pamięci RAM stanowi proste rozszerzenie tej koncepcji tak, aby wymieniane były między sobą bloki pamięci RAM zamiast bloków pamięci RAM i ROM. Przełączanie takie jest realizowane przez odpowiedni program w języku asemblera.

Program taki o nazwie BANK_MAN.BAS jest umieszczony na Stronie 1 pakietu dysków systemowych. Uruchomienie tego programu w systemie BASIC powoduje zainstalowanie rozszerzenia systemowego (RSX), umożliwiającego administrowanie bankami pamięci. Rozszerzenie to jest określane mianem "BANK MENAGER".

Jedną z możliwości użycia drugiej części 54 KB pamięci RAM jest chwilowe przechowywanie treści obrazów ekranu. Może to być wykorzystane np. przez program "Scren Designer" - "Projektant

Obrazu" do przechowywania wielu różnych zawartości ekranu lub w grach wizyjnych do przechowywania wcześniej przygotowanych różnych obrazów ekranu.

Inną możliwością użycia drugiej części 64 KB pamięci RAM jest rozszerzenie przestrzeni roboczej zmiennych, co może być uważane bądź za rozszerzenie przestrzeni tablic ciągów bądź też po prostu za "dysk typu RAM".

Część 1: Przechowywanie obrazów ekranu

Wybierz swój ekran ...

BANK MENAGER umożliwia wyłączenie Bloku 1 i włączenie w jego miejsce jednego z czterech bloków drugich 64 KB pamięci RAM.

Należy zauważyć, że każdy z bloków drugich 64 KB pamięci wprowadzany jest w tę samą przestrzeń adresową (&4000 do &7FFF). Zawartość Bloku 1 (być może środkowa część aktualnie działającego programu w języku BASIC!) jest przechowywana i odtwarzana po zakończeniu akcji przez BANK MENAGER. Są także możliwe trzy inne sposoby przełączenia banku (odmienne od pięciu możliwych konfiguracji, wynikających z opisanego wyżej sposobu), lecz są one używane tylko przy implementacji CP/M Plus.

BANK MENAGER wykonuje dwa rozkazy, umożliwiające przenoszenie informacji traktowanej jako zawartość ekranu z jednego bloku pamięci do innego, Bloki 4 do 7 są przy tym włączane i wyłączane automatycznie zgodnie z wymaganiami a po zakończeniu wykonywania rozkazu Blok 1 włączany jest na swoje miejsce.

Rozkaz:

|SCRFENSWAP,[<sekcja ekranu>],<numer ekranu>,<numer ekranu

wymienia zawartość dwóch bloków, natomiast:

|SCRFENCOPY,[<sekcja ekranu>],<docelowy numer ekranu>,
<źródłowy numer ekranu>

kopiuje zawartość jednego bloku do innego.

Wprowadzenie opcjonalnego parametru, nazwanego <sekcja ekranu> powoduje kopiowanie tylko 1/64 bloku (256 bajtów z 16 KB). <sekcja ekranu> może mieć wartość z zakresu od 0 do 63. Taki tryb pracy jest użyteczny gdy chce się wprowadzać

jakieś inne działania związane z ruchem na ekranie. Wymiana zawartości ekranu może trwać około 150/300 sekundy (150 okresów funkcji TIME).

Parametr <numer ekranu> przyjmuje wartości 1 (dla normalnie wyświetlonego ekranu) lub 2,3,4 i 5. Operacje kopiowania i wymiany prowadzone z ekranem numer 1 są znacznie szybsze. Należy unikać rolowania ekranu, co wyjaśniano przy omawianiu przechowywania treści ekranu na dysku. Powinno się zapewnić takie warunki, aby obraz ekranu był tworzony i wyświetlany z ekranem nr 1 ustawionym w tej samej sprzętowej pozycji. Najprostszą (domyślnie przyjmowaną) pozycją jest pozycja ustawiana rozkazem MODE.

Wypróbuj rozkazy przełączania ekranu ...

Najpierw uruchom program BANK MENAGER ze Strony 1 pakietu dysków systemowych:

```
RUN "BANKMAN"
```

Następnie napisz:

```
MODE 1
```

Ekran został wyczyszczony. Teraz napisz:

```
' To jest ekran oryginalny
|SCREENCOPY,3,1 ' prześlij ekran oryginalny do pamięci 3
CLS
```

Ekran znowu został wyczyszczony. Teraz napisz:

```
'To jest ekran pośredni
|SCREENCOPY,2,1 'prześlij ekran pośredni do pamięci 2
|SCREENSWAP,2,3 'wymień pamięci 2 i 3
|SCREENCOPY,1,3 'Odtwórz ekran pośredni z pamięci 3
|SCREENCOPY,1,2 'Odtwórz ekran oryginalny z pamięci 2
```

Na zakończenie tego tematu, ostatnia część Rozdziału 9 zawiera obszerny program pod nazwą "Screen-Designer" - "Projektant Obrazu", w którym wykorzystywane są możliwości przełączania ekranu, zapewniane przez program BANK MENAGER.

Część 2: Działanie z pseudo-zbiorami

Jeszcze trochę o zbiorach ...

W przypadku używania jako "dysk typu RAM", drugie 64 KB pamięci jest dzielone na "zbiory RAM", zawierające określoną liczbę rekordów o stałej długości. Długość rekordu może być od 0 do 255 bajtów, przy czym 2 bajty zalecane są jako minimum. Po ustaleniu długości "rekordu RAM", każdy rekord jest oznaczony "numerem rekordu RAM". Jest także dopuszczalne zapisywanie "zbioru RAM" przy użyciu jednej długości rekordu a odczytywanie go z powrotem przy użyciu innej długości.

UWAGA: Zbiór RAM zawierać może jedynie dane; NIE MA żadnej możliwości, aby zawierał instrukcje programu.

Podobnie jak w przypadku swobodnego dostępu do zbiorów dyskowych, stosuje się tutaj koncepcję "bieżącego numeru rekordu". Prowadzi to w rezultacie do korzystania z domyślnego numeru rekordu, co jest szczególnie użyteczny przy automatycznym przesuwaniu się przez zbiór RAM.

Polecenie:

|BANKOPEN, ⟨długość rekordu RAM⟩

określa stałą długość wszystkich rekordów i wprowadza bieżący numer rekordu równy zeru; NIE ZMIENIA przy tym w żaden sposób (nie czyści) zawartości pamięci.

Polecenie:

|BANKWRITE, @ ⟨kod powrotny⟩ . ⟨wyrażenie tekstowe⟩ , ⟨numer rekordu RAM⟩

wpisuje ⟨wyrażenie tekstowe⟩ do zbioru RAM.

⟨numer rekordu RAM⟩ określa rekord, do którego następuje wpisywanie. Jeżeli parametr ten jest pominięty, używany jest bieżący numer rekordu. Bieżący numer rekordu jest następnie ustawiany tak, aby określał następny rekord.

Jeżeli ⟨wyrażenie tekstowe⟩ nie wypełnia całkowicie rekordu, stare znaki (które nie zostały zastąpione wprowadzonymi właśnie znakami) pozostają do końca rekordu. Jeżeli ⟨wyrażenie tekstowe⟩ jest dłuższe od rekordu, znaki nadmiarowe są pomijane (odrzucane) aby zapobiec przelaniu się do następnego rekordu.

<kod powrotny> jest zmienną typu liczby całkowitej, która przyjmuje wartość równą numerowi rekordu, do którego wpisywano wyrażenie, jeżeli operacja zakończona została prawidłowo lub podaje ujemny kod błędu, jeżeli operacja wpisywania nie udała się z jakiś przyczyn:

-1 Błąd końca zbioru. Adres żądanego numeru rekordu przekracza 64 KB

-2 Błąd przełączenia banku (nie powinien nigdy wystąpić)

Przykłady:

```
|BANKOPEN,10
|BANKWRITE,∂r%,"123 testing",Ø
|BANKWRITE,∂r%,w$
```

Polecenie:

```
|BANKREAD, <kod powrotny> , <zmienna tekstowa> ,
                          <numer rekordu RAM>
```

czyta rekord ze zbioru RAM do <zmiennej tekstowej>

<numer rekordu RAM> podaje który rekord ma być czytany. Jeżeli parametr ten został pominięty, używany jest bieżący numer rekordu. Bieżący numer rekordu jest następnie ustawiany tak, aby określał następny rekord.

Jeżeli zawartość rekordu nie wypełniła całkowicie <zmiennej tekstowej>, stare znaki (które nie zostały zastąpione wprowadzonymi właśnie znakami) pozostają do końca <zmiennej tekstowej>. Jeżeli zawartość rekordu jest dłuższa od długości <zmiennej tekstowej>, znaki nadmiarowe są pomijane (odrzucane), ponieważ nie jest możliwe zwiększenie długości zmiennej tekstowej podczas wykonywania rozkazu zewnętrznego.

<kod powrotny> jest zmienną typu liczby całkowitej, która przyjmuje wartość równą numerowi rekordu, z którego czytano, jeżeli operacja została przeprowadzona prawidłowo, lub podaje ujemny kod błędu, jeżeli operacja czytania nie udała się z jakiś przyczyn:

-1 Błąd końca zbioru. Adres żądanego numeru rekordu przekracza 64 KB

-2 Błąd przełączenia banku (nie powinien nigdy wystąpić)

Przykład:

```
|BANKREAD,)r%,1S,0
```

Wyszukiwanie ...

Jest możliwe przeszukiwanie przechowywanych rekordów w celu znalezienia określonego ciągu znaków.

Polecenie:

|BANKFIND, <kod powrotny> , <poszukiwany ciąg> ,
[, <początkowy numer rekordu > [, <końcowy numer rekordu>]]

przeszukuje wszystkie zadane rekordy RAM. <początkowy numer rekordu > określa, od którego rekordu należy zacząć poszukiwanie. Jeżeli parametr ten jest opuszczony, przyjmowany jest bieżący numer rekordu.

Przeszukiwanie jest prowadzone krokami o <długości rekordu RAM > poprzez całe drugie 64 KB pamięci aż do znalezienia odpowiednika.

Jeżeli określono <końcowy numer rekordu> , przeszukiwanie kończy się po zbadaniu tego rekordu (chyba że wcześniej znaleziono odpowiednik).

Jeżeli poszukiwanie było skuteczne, bieżący numer rekordu staje się równy numerowi rekordu, w którym znaleziono odpowiednik (w przeciwnym wypadku pozostaje niezmieniony).

<kod powrotny> określa zmienną typu liczby całkowitej która przyjmuje wartość równą numerowi rekordu, w którym znaleziono odpowiednik (jeżeli poszukiwanie było skuteczne) lub podaje ujemny kod błędu, jeżeli poszukiwanie było z jakiś powodów nieskuteczne:

-1 Koniec zbioru. Adres początkowego rekordu przekracza 64 KB lub numer rekordu początkowego jest większy od numeru rekordu końcowego

-2 Błąd przełączania banku (nie powinien nigdy wystąpić)

-3 Nie znaleziono odpowiednika.

<poszukiwany ciąg> może zawierać znaki zastępcze, sygnalizowane przez znaki puste - CHR$ (Ø) a porównanie może być prowadzone przy przyjęciu albo <długości rekordu RAM> albo długości <poszukiwanego ciągu> , zależnie od tego co jest krótsze.

Przykłady:

```
|BANKFIND,∂r%,"123 test",Ø
|BANKFIND,∂r%,fS,100,200
```

Strzeż się niezgodności ...

Oczywiste błędy, takie jak zła liczba parametrów, sygnalizowane są jako "B a d c o m m a n d" - "Z ł y r o z k a z". Jednakże procedury obsługi rozkazów zewnętrznych nie wykrywają błędów w rodzaju "T y p e m i s m a t c h" - "N i e - z g o d n o ś ć t y p u" i użytkownik musi być pewien, że stosowany jest właściwy typ parametrów.

Program, zamieszczony poniżej używa rozkazów operujących z "dyskiem typu RAM" do założenia i przeszukiwania bazy danych, zawierającej anagramy słów 7-literowych. Program wyszukuje odpowiedniki i dopuszcza używania znaków zastępczych.

Dla przykładu anagramami słowa FIGURES, które odpowiadają wzorcowi ?RUGS?? (dwa ostatnie ?? mogą być zresztą pominięte są FRUGSIE, FRUGSEI, IRUGSFE, IRUGSEF, ERUGSFI i IRUGSIF.

Tworzenie bazy danych wymaga nieco czasu, lecz 64 KB to dużo pamięci do wypełnienia!

Pamiętaj, aby przed uruchomieniem programu uruchomić najpierw program "BANKMAN" (RUN "BANKMAN")!

```
10 'ANAGRAMY autor ROLAND PERRY
20 'copyright (c) AMSOFT 1985
30 '
40 'Pamietaj wczesniej uruchomic program BANKMAN
50 '*****************************************
60 '
70 MODE 2
80 DEFINT a-z
90 r%=0:|BANKOPEN,7
100 INPUT"Jakie 7-literowe slowo mam przetwarzac ";s$
110 IF LEN(s$)<>7 THEN 100
120 PRINT"Prosze czekac..."
130 LOCATE 1,5:PRINT"Przetwarzam:"
140 FOR c1=1 TO 7
150 FOR c2=1 TO 7
160 IF c2=c1 THEN 370
170 FOR c3=1 TO 7
180 IF c3=c2 OR c3=c1 THEN 360
190 FOR c4=1 TO 7
200 IF c4=c3 OR c4=c2 OR c4=c1 THEN 350
210 FOR c5=1 TO 7
220 IF c5=c4 OR c5=c3 OR c5=c2 OR c5=c1 THEN 340
230 FOR c6=1 TO 7
240 IF c6=c5 OR c6=c4 OR c6=c3 OR c6=c2 OR c6=c1 THEN 330
250 FOR c7=1 TO 7
260 IF c7=c6 OR c7=c5 OR c7=c4 OR c7=c3 OR c7=c2 OR c7=c1 THEN 320
270 o$=MID$(s$,c1,1)+MID$(s$,c2,1)+MID$(s$,c3,1)+MID$(s$,c4,1)
    +MID$(s$,c5,1)+MID$(s$,c6,1)+MID$(s$,c7,1)
280 LOCATE 12,5:PRINT x;o$
290 |BANKWRITE,@r%,o$
300 IF r%<0 THEN 100
310 x=x+1
320 NEXT c7
330 NEXT c6
340 NEXT c5
350 NEXT c4
360 NEXT c3
370 NEXT c2
380 NEXT c1
390 lastrec=r%
400 REM teraz spojrzmy na wynik
410 r%=0:g$=SPACE$(7)
420 PRINT:INPUT"Jaki odpowiednik sobie zyczysz: uzyj ? jako
    znak zastepczy: ";m$
430 m$=LEFT$(m$,7)
440 FOR x=1 TO LEN(m$)
450 IF MID$(m$,x,1)="?" THEN MID$(m$,x,1)=CHR$(0)
460 NEXT
470 |BANKFIND,@r%,m$,0,lastrec
480 IF r%<0 THEN GOTO 420
490 |BANKREAD,@r%,g$
500 PRINT g$,
510 |BANKFIND,@r%,m$,r%+1,lastrec
520 GOTO 490
```

ROZDZIAŁ 9

W WOLNEJ CHWILI ...

W rozdziale tym podano niektóre podstawowe informacje o komputerach ogólnie i o 6128 w szczególności. Nie musisz czytać tego rozdziału przed przystąpieniem do pracy z komputerem, pozwoli ci on jednak zrozumieć trochę lepiej co się dzieje "pod pokrywką".

Część 1. OGÓLNIE MÓWIĄC ...

O co chodzi?

Nawet jeżeli jedynym powodem, dla którego kupiłeś 6128, była możliwość korzystania z wyrafinowanych gier komputerowych, prawdopodobnie ciekaw jesteś niektórych cech komputera kryjących się pod nazwą "sprzęt" (hardware).

Sprzęt jest to coś, co można wziąć w ręce i przenieść, to jest: główna klawiatura komputera, monitor, kable łączące itp. Faktycznie jest to wszystko to, co nie jest "oprogramowaniem" (software) - programem, podręcznikiem, informacją zapisaną na dysku lub taśmie magnetycznej.

Niektóre cechy zachowania się komputera wynikają ze specyfiki sprzętu - na przykład wyświetlanie kolorów na odbiorniku telewizyjnym (monitorze). Zadaniem oprogramowania jest wykorzystanie możliwości sprzętowych do wytworzenia na ekranie określonych znaków i kształtów.

Sprzęt w rzeczywistości kieruje strumień elektronów na pokrytą warstwę luminescencyjną wewnętrzną powierzchnią kineskopu i powoduje rozświetlenie kineskopu - oprogramowanie dodaje do tego uporządkowanie i inteligencję, mówiąc sprzętowi, kiedy i jak należy to robić. Oprogramowanie dodaje taktowanie, sterowanie i narzuca kolejność czynności dających efekt odlatującego statku kosmicznego lub, bliżej ziemi, efekt litery pojawiającej się na ekranie wtedy, kiedy naciskasz klawisz.

Coż więc powoduje, że jeden komputer jest lepszy od drugiego?

Sprzęt bez oprogramowania jest bezwartościowy. Oprogramowanie bez sprzętu - również. Wartość komputera zaczyna się wówczas, kiedy sprzęt i oprogramowanie spotykają się razem, w celu wykonywania różnych zadań. Istnieją bardzo podstawowe cechy, które mogą być używane do oceny możliwości zarówno sprzętu, jak i oprogramowania.

Ogólnie przyjęte obecnie cechy porównywalne komputerów są następujące:

1. Rozdzielczość ekranu - najmniejszy, rozróżnialny punkt ekranu. Składają się na nią czynniki zawierające liczbę kolorów dostępnych dla programisty, liczbę różnych plamek, które można uzyskać na ekranie i liczbę znaków tekstowych, które mogą być wyświetlane na ekranie.
   Przekonasz się, że twój 6128 wypada bardzo korzystnie, w porównaniu z innymi komputerami o podobnej cenie, pod każdym względem.

2. Interpreter BASIC'a
   W chwili obecnej każdy komputer domowy jest wyposażony w interpreter BASIC'a, który pozwala użytkownikowi rozpocząć tworzenie programów wykorzystujących właściwości sprzętu. Wbudowany w twoją maszynę język programowania (BASIC) jest ogromnie skomplikowanym i zawiłym programem, który rozwinął się dzięki doświadczeniom wielu ludzi od czasu, kiedy został "wymyślony" w USA. BASIC (Beginers All-purpose Symbolic Instruction Code) jest z pewnością najszerzej w świecie używanym językiem komputerowym i, podobnie jak inne języki, ma swoje lokalne "dialekty".
   Wersja zastosowana w 6128 jest jedną z najbardziej podobnych do innych wersji BASIC'a, pozwala ona również na uruchamianie programów napisanych do pracy pod kontrolą systemu zarządzającego dyskami CP/M. Jest to bardzo szybka implementacja BASIC'a - innymi słowy szybko wykonuje obliczenia - i chociaż może cię nie obchodzić, że jeden komputer mnoży 3 razy 5 i wyświetla wynik w ciągu 0,05 sekundy, a drugi robi to w 0,075 sekundy - jeżeli program rysujący kształty na ekranie

może wywoływać wiele tysięcy razy proste operacje arytmetyczne, różnica pomiędzy 0,05 i 0,075 sekundy może urosnąć do istotnej różnicy w szybkości działania.
Czasem będziesz słyszał termin "kod maszynowy". Kod maszynowy jest szeregową formą kodu instrukcji, która jest przekazywana do procesora. Program napisany w kodzie maszynowym działa 5 do 15 razy szybciej niż taki sam program wykonywany przez interpreter BASIC'a. Z drugiej strony napisanie programu w kodzie maszynowym zajmuje 5 do 50 razy więcej czasu niż napisanie równoważnego programu w języku BASIC.
BASIC w twoim Amstradzie jest jednym z najszybszych i najlepiej wyposażonych BASIC'ów stosowanych w komputerach domowych i ma wiele właściwości, które pozwalają doświadczonemu programiście przezwyciężyć niektóre wewnętrzne spowolnienia interpretera "języka wysokiego poziomu" i uzyskać zdumiewająco dynamiczne efekty wizualne i muzyczne.

3. Możliwości rozszerzenia
W większości komputerów zwraca się wiele uwagi na możliwość dołączenia dodatkowych urządzeń sprzętowych: drukarki, joysticka, zewnętrznego napędu dysków. Paradoksem jest, że niektóre z najbardziej udanych komputerów domowych muszą być połączone nawet z prostą drukarką lub joystickiem poprzez specjalne układy sprzęgające zwane interfejsami.
Nabywca nie zawsze myśli o swoich potrzebach w przyszłości, a przecież maszyna wyposażona we właściwie sterowany, równoległy port wyjściowy do drukarki (typu Centronix) oraz port wejściowy joysticka może być istotnie tańsza w rzeczywistych warunkach.
Komputer 6128 ma wbudowany port do drukarki typu Centronix, port do dodatkowego napędu dysków, złącze do połączenia z magnetofonem (ze sterowaniem silnika), złącza do bezpośredniego dołączenia dwóch joysticków, wyjście do stereofonicznego wzmacniacza akustycznego i wyjście magistrali systemowych, które może być użyte do dołączenia łącza szeregowego (AMSTRAD model RS232C), MODEM'u, syntetyzera mowy (AMSTRAD model SSA2), pióra świetlnego itd.

4. Dźwięk

Właściwości dźwiękowe komputera decydują o tym, czy brzmi on jak mucha w pustej puszce po kakao, czy wytwarza zadowalającą muzykę elektronicznego instrumentu muzycznego Komputer 6128 ma 3 kanałowy, 8 oktawowy generator dźwięku, który może wytwarzać dźwięk bardzo dobrej muzycznie jakości, z pełną kontrolą obwiedni głośności i tonu. Co więcej, dźwięk jest rozłożony w konfiguracji stereofonicznej: jeden kanał daje dźwięk z lewej strony, drugi kanał z prawej, trzeci kanał daje efekt dźwięku usytuowanego w środku. Zapewnia to istotną możliwość pisania programów, które tworzą efekty dźwiękowe nadążające za ruchem na ekranie.

Ostatecznie sam zdecydujesz, które z tych właściwości są najważniejsze dla ciebie. Mamy nadzieję, że spróbujesz wszystkich, żeby uzyskać jak najwięcej za pomocą twojego komputera.

Dlaczego nie potrafi?

Użytkownicy często dziwią się dlaczego, przy całej potędze nowoczesnej technologii, nawet tak zaawansowana maszyna jak 6128, najwidoczniej nie jest w stanie wykonywać zadań w sposób obserwowany na przykład w telewizji. Dlaczego komputer nie potrafi animować rysunku postaci kroczącej przez ekran w naturalny sposób? - dlaczego wszystkie komputery przedstawiają ruch "patykowatych" figur?

Odpowiedź jest prosta, a zarazem złożona. Prosta odpowiedź jest taka, że nie powinieneś łudzić się, że ekran twojego komputera ma cokolwiek z subtelności ekranu telewizyjnego. Telewizja działa z wykorzystaniem "liniowej" informacji, która może opisywać w rzeczywistości nieskończoną liczbę stanów pomiędzy największą jaskrawością i największą ciemnią wszystkich kolorów zakresu widzialnego. Oznacza to, że w przypadku komputera, pamięć pełnego obrazu telewizyjnego byłaby około dwadzieścia razy większa od pamięci ekranu komputera domowego.

To tylko część problemu, ponieważ animacja takiego obrazu wymagałaby przetwarzania tej niezwykłej ilości pamięci z bardzo dużą szybkością (50 razy na sekundę). To można zrobić - ale tylko

za pomocą maszyny, która kosztuje kilka tysięcy razy więcej niż komputer domowy, przynajmniej jak na razie.

Aż do czasu, kiedy cena bardzo szybkiej pamięci spadnie radykalnie (w końcu to nastąpi), małe komputery muszą zadowalać się stosunkowo małą pamięcią przeznaczoną do sterowania obrazem ekranu, co objawia się mniejszym zróżnicowaniem i ruchami mniej płynnymi. Przemyślana konstrukcja sprzętu i dobre programowanie mogą znacznie poprawić sytuację, ale ciągle będziemy daleko od tanich komputerów, które mogą wytwarzać obrazy naśladujące życie i płynne ruchy, w takim samym stopniu, jak może to robić przeciętny film rysunkowy.

Ta klawiatura wygląda swojsko ...

Dlaczego nie możesz włączyć swojego komputera i napisać na ekranie dowolnego tekstu?

Niech cię nie zwiedzie fakt, że komputer wygląda jak maszyna do pisania z elektronicznym wyświetlaczem. Ekran nie jest elektronicznym papierem - to "konsola operacyjna" - żargon, który oznacza, że ekran jest środkiem komunikacji z językiem programowania (i programami) w pamięci maszyny.

Dopóki nie powiesz przekorze o co ci chodzi, komputer będzie próbował interpretować wszystkie znaki, które wpisujesz klawiaturą tak, jakby to były instrukcje programowe. Kiedy naciśniesz klawisz RETURN , komputer przejrzy to, co było wpisane i, jeżeli nie miało to żadnego sensu dla wbudowanego BASIC'a, odrzuci wpisaną linię tekstu z komunikatem:

Syntax error (błąd składni)

Tym niemniej, może się zdarzyć, że programem aktualnie czynnym w twoim komputerze jest program przetwarzania tekstu (word processor). W tym przypadku możesz wpisywać przypadkowe słowa, naciskać [RETURN] i pisać dalej, jak gdyby ekran był kawałkiem elektronicznego papieru w elektronicznej maszynie do pisania. Ale, żeby tak było, musiałeś wpierw wprowadzić do pamięci maszyny program przetwarzania tekstu.

Komputer wydaje się być kombinacją sprzętu znanego z domu lub biura, takiego jak ekran telewizyjny i klawiatura. Musisz

pamiętać, że podobieństwa są wyłącznie zewnętrzne i, że komputer jest kombinacją zwyczajnie wyglądającego sprzętu o całkowicie odmienionych właściwościach.

Kto się boi żargonu?

Podobnie jak wszystkie specjalistyczne dziedziny działalności człowieka, dziedzina komputerów wytworzyła swój własny żargon jako skrótową formę przekazywania skomplikowanych znaczeń, których wyrażenie w "zrozumiałym języku" wymaga użycia wielu słów. To nie przemysł rozwiniętej technologii jest winien chowania się za zasłoną dymną "pustych słów", żargonu i terminologii - większość z nas występuje przeciwko barierom zrozumienia wznoszonym przez wszystkie główne grupy zawodowe.

Główna różnica pomiędzy żargonem prawniczym na przykład, a żargonem komputerowym polega na tym, że żargon prawniczy używa znanych słów w sposób niezrozumiały, podczas gdy komputerowy raczej tworzy nowe słowa. Większość ludzi, którzy wzrastali w przyjaźni z terminologią komputerową, przyzwyczaja się używać słów w sposób możliwie najprostszy, po to, aby minimalizować złożoność porozumienia. Nie daj się zwieść zwolennikom "zrozumiałego języka", to nie jest dziedzina literatury, to nauka ścisła i, pozostawiając na boku budowę słów, struktura komunikacji jest tutaj bardzo jasna i prosta i nie jest w najmniejszym stopniu myląca czy mętna. Nauczyciele informatyki nie zdążyli jeszcze uczynić sztuki z prób analizy dokładnych znaczeń, które chciał przekazać programista w konstrukcji swojego programu.

Powiedziawszy to, trzeba dodać, że pomijając pytanie czy znaczenie programu komputerowego jest oczywiste, jest wiele cech programu, które mogą być uważane za eleganckie lub niechlujne i, teraz, kiedy początkowy szum dookoła rewolucji mikrokomputerowej opada, coraz więcej wagi przykłada się do formalnego podejścia do konstrukcji programów.

Programowanie jest szybko pojmowane przez wielu młodych ludzi, którzy doceniają precyzję i prostotę pomysłów i sposób w jaki mogą one być przekazywane - wśród dziesięciolatków znajdziesz niewielu prawników, za to mnóstwo programistów.

Podstawy BASIC'a

Właściwie wszystkie komputery domowe można programować w BASIC'u, języku, który pozwala pisać programy najbardziej zbliżone do zrozumiałego języka (angielskiego). BASIC w chwili obecnej nie ma jakiegoś szczególnego znaczenia jako stopień wśród mniej lub bardziej wyrafinowanych języków i wiele bardzo złożonych programów o wielkich możliwościach jest pisanych właśnie w BASIC'u.

Tym niemniej nazwa, bez wątpienia, przyciąga wielu nowicjuszy obietnicą pozycji startowej w labiryncie języków programowania komputerów i to zadecydowało o jego uniwersalności.

BASIC jest językiem komputerowym, który interpretuje pewne, dozwolone rozkazy, a następnie dokonuje operacje na danych podczas wykonywania programu. W przeciwieństwie do słownika przeciętnego człowieka zawierającego 5000÷8000 słów (plus wszystkie odmiany), słownik BASIC'a zawiera ich tylko około 200. Programy komputerowe pisane w BASIC'u muszą przestrzegać ścisłych zasad dotyczących użycia tych słów. Składnia jest ściśle określona i każda próba porozumienia z komputerem przy użyciu wyrażeń literackich lub potocznych (to jest zrozumiałym językiem) kończy się zimnym komunikatem:

Syntax error

Nie jest to takim ograniczeniem jak z początku się wydaje, ponieważ język BASIC (składnia) jest skonstruowany głównie w celu manipulacji liczbami - danymi numerycznymi. Słowa są zasadniczo rozwinięciem zwykłych operatorów matematycznych +/- itd. i najważniejszym faktem, z którego powinien zdać sobie sprawę nowicjusz jest to, że komputer może pracować wyłącznie z danymi liczbowymi. Informacja przekazywana do układu scalonego CPU (Central Processing Unit - centralna jednostka przetwarzania jest zawsze w postaci danych numerycznych.

Numer proszę ...

Jeżeli komputer został użyty do przechowania dzieł wszystkich Shakespeare'a, w systemie nie znajdziesz ani jednego słowa czy litery. Każda cząstka informacji jest najpierw przetworzona na liczbę, którą komputer może przesyłać i manipulować jak potrzeba.

BASIC zamienia słowa na liczby,którymi komputer następnie manipuluje używając tylko dodawania, odejmowania i działań logicznych, które pozwalają porównywać dane i selekcjonować je według pewnych cech - innymi słowy sprawdzać czy jedna liczba jest większa lub równa drugiej oraz wypełniać określone zadania, jeżeli jedna liczba lub inna spełnia pewne kryteria.

Za pośrednictwem programu komputer rozkłada każde zadanie na prosty ciąg operacji Tak/Nie.

Jeżeli ten proces wydaje ci się uciążliwy, to masz rację, ponieważ odkryłeś pierwszą i najważniejszą prawdę o komputerze. Komputer jest głównie narzędziem do rozwiązywania najprostszych i powtarzających się zadań, bardzo szybko i z absolutną precyzją. Tak więc BASIC interpretuje instrukcje podane w formie programu i tłumaczy je na język zrozumiały dla CPU. Tylko dwa stany są zrozumiałe dla logiki komputera - "tak" i "nie", reprezentowane w zapisie binarnym przez "1" i "0". W logice Boole'a odpowiadają one pojęciom "prawda" i "fałsz" - nie ma tu takich pojęć jak "być może" czy "prawdopodobnie".

Proces przełączania pomiędzy tymi dwoma wykluczającymi się stanami jest istotą pojęcia "cyfrowy". W świecie rzeczywistym większość ruchów odbywa się stopniowo, od jednego stabilnego stanu do innego w sposób liniowy. Innymi słowy, przejście dokonywane jest po ścieżce wzdłuż linii łączącej dwa stany - w idealnym cyfrowym świecie, przełączanie od jednego stanu do następnego dokonywane jest w nieskończenie krótkiej chwili - w rzeczywistości występuje małe opóźnienie, nazywane czasem propagacji - i akumulacja tych czasów propagacji powoduje, że komputer potrzebuje trochę czasu na przetworzenie informacji, zanim może dać odpowiedź na postawione zadanie.

W każdym razie, komputer musi czekać przez skończony okres czasu na zakończenie jednego zadania, zanim może zacząć korzystać z wyników tego zadania - tak więc pewne sztuczne opóźnienie i tak musi być wprowadzone. Przetwarzanie cyfrowe jest "czarno-białe" i stopnie przejścia przez różne "odcienie szarości" NIE mają żadnego znaczenia. Odwrotnie, liniowe czy "analogowe" przetwarzanie JEST związane z różnymi odcieniami szarości.

Jeżeli ostateczną odpowiedzią jest 0 lub 1, to nie ma możliwości, żeby była to odpowiedź "prawie dobra". Fakt, że komputer czasem popełnia błędy przy przetwarzaniu danych liczbowych, wynika z ograniczenia długości liczb powodującego skrócenie lub "obcięcie" zbyt długich liczb w celu dopasowania ich do przeznaczonego na nie miejsca w pamięci, prowadzące do błędów zaokrąglenia, na przykład 999,999,999 staje się równe 1,000,000,000.

Jak, w świecie gdzie dostępne są tylko dwie cyfry: 0 i 1, można przedstawiać liczby większe niż 1?

Bity i bajty

Przyzwyczailiśmy się rozumieć liczby zapisywane w systmie dziesiętnym, gdzie punktem odniesienia jest liczba 10 - to jest mamy dziesięć cyfr w zakresie 0 do 9. System, w którym są tylko dwie cyfry 0 i 1 nazywa się systemem binarnym, a pozycje tych cyfr w zapisie liczby nazywa się bitami (Binary digIT - cyfra binarna).

Zależność pomiędzy bitami i zapisem dziesiętnym jest prosta. Zazwyczaj deklarujemy maksymalną liczbę bitów aktualnie używaną w komputerze przez uzupełnienie liczby z przodu nieznaczącymi zerami, tak, że długość liczby jest zawsze taka sama: na przykład 7 w systemie dziesiętnym, jest równe:
00111 w systemie binarnym
... jeżeli stosujemy zapis 5-bitowy.

W systemie binarnym cyfry mogą być traktowane jak wskaźniki w kolumnach pokazujące czy dana potęga 2 występuje czy nie; 1 = tak, 0 = nie.

$2^0 = 1$
$2^1 = 2 = 2 = 2(2^0)$
$2^2 = 4 = 2x2 = 2(2^1)$
$2^3 = 8 = 2x2x2 = 2(2^2)$
$2^4 = 16 = 2x2x2x2 = 2(2^3)$

... tak więc kolumny wyglądają tak:

2^4	2^3	2^2	2^1	2^0
1	0	0	1	1

(16 + 0 + 0 + 2 + 1) = 19 (dziesiętnie)

Termin "bajt" oznacza informację 8-bitową i został wprowadzony jako skrót językowy. Maksymalna liczba, jaka może być zapisana w bajcie jest więc (binarnie) 11111111 - lub (dziesiętnie) 255. To daje 256 różnych wariancji, wliczając 00000000, które jest równie dobrą daną dla komputera.

Komputery mają tendencję do manipulowania danymi będącymi wielokrotnościami 8 bitów. 256 nie jest zbyt wielką liczbą i po to, żeby uzyskać zadowalający sposób obsługi pamięci używa się dwóch bajtów, do jej adresowania, co przyjmuje formę tablicy, w której każdy element jest zlokalizowany przez adres poziomy i pionowy:

	0	1	2	3	4	5	6	7	8	9
0										
1										
2										
3										
4										
5						1		1		
6										
7										
8										
9										

Narysowana tablica ma (10x10) pozycji przy użyciu liczb adresowych z zakresu 0 do 9. Wartość elementu na pozycji 5, 3 jest równa "1" - podobnie jak elementu 5, 5.

Tablica binarna 256x256 może obsłużyć 65536 indywidualnych lokacji przy użyciu 8-bitowych adresów poziomych i pionowych. Tak więc nasze "0" i "1" stało się zdolne do zidentyfikowania jednego z 65536 różnych elementów.

Następnym skrótem językowym stosowanym do informacji binarnej jest kilobajt (kB lub K), który jest równy 1024 bajtów. 1024 jest potęgą dwójki najbliższą liczby, której dotyczy normalnie stosowany termin "kilo" (1000). To tłumaczy, dlaczego komputer z "64K" pamięcią, w rzeczywistości ma pamięć 65536 bajtową (64x1024).

Na szczęście interpreter BASIC'a wykonuje za ciebie wszystkie potrzebne konwersje i, jest całkiem możliwe stać się biegłym programistą, nie rozumiejąc zupełnie liczb binarnych, chociaż, docenienie znaczenia liczb binarnych pomoże ci rozszyfrować wiele "magicznych" lub ważnych liczb, które nieuchronnie pojawią się w trakcie poznawania nauki o komputerach.

Warto wysilić się trochę i przyswoić sobie znajomość systemu binarnego i różnych znaczących liczb, jak 255, 1024 itp., ponieważ jest zupełnie nieprawdopodobne, żeby system binarny przestał być "kołyską" wszelkich operacji komputerowych w najbliższej przyszłości. Pewność działania i prostota wynikająca z przyjęcia tylko dwóch stanów stabilnych ma przewagę nad niezwykle zwiększoną kompleksowością, która byłaby efektem przyjęcia jakiejkolwiek innej liczby bazowej.

Tym niemniej ...

Prosty i elegancki zapis binarny jest rozwlekły i skłania do niedokładności, ponieważ nie może być łatwo czytany jednym spojrzeniem. System binarny ma kilka pokrewnych systemów liczenia, które służą programistom jako skrót zapisu. Jednym z takich systemów, szeroko używanym w programowaniu mikrokomputerów, jest system heksadecymalny (HEX).

W systemie tym liczbą bazową jest 16 i tyle jest cyfr:

Dziesiętnie:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Heksadecymalnie:

0 1 2 3 4 5 6 7 8 9 A B C D E F

Ponieważ 15 jest liczbą binarną 4-bitową: 1111, system heksadecymalny pozwala zapisać 8 bitów w postaci dwóch cyfr HEX (dwóch bloków 4 bitowych). Pierwsza cyfra wskazuje liczbę pełnych jednostek po "16", druga wskazuje "resztę" - i tu zaczyna się wyłaniać absolutna elegancja systemu binarnego i heksadecymalnego.

Oto tablica tych systemów:

Dziesiętnie	Binarnie	Heksadecymalnie
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5

6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

Liczba 8-bitowa np. 11010110 (& D6 hex) może być podzielona na dwie grupy i traktowana jako dwie 4-bitowe liczby. W niniejszym podręczniku liczba heksadecymalna jest poprzedzona symbolem "&", to jest & D6. Zapis heksadecymalny jest najchętniej używany przez programistów programujących w języku asemblera. Program w języku asemblera jest już bardzo blisko programu napisanego bezpośrednio w kodzie maszynowym, ponieważ język asemblera pozwala programiście używać prostych "mnemoników" literowych odpowiadających "liczbom" ostatecznego kodu maszynowego.

Przy używaniu zapisu hex, należy pamiętać, że pierwsza cyfra oznacza liczbę 16-tek, w ostatecznej liczbie, a druga cyfra resztę, którą należy dodać, żeby otrzymać ostateczną liczbę dziesiętną. Wielką pokusą jest traktować liczbę & D6 jak 13+6 lub 136, ale naprawdę to jest (13x16) + (6) = 214.

Jest to taki sam proces jaki stosujesz czytając liczbę dziesiętną taką jak "89" - to jest (8x10) + (9). Tak się składa (dzięki przyzwyczajeniu do systemu dziesiętnego), że mnożenie przez 10 jest dla ciebie znacznie łatwiejsze, chyba, że masz dużą wprawę w mnożeniu przez 16.

Jeżeli dotarłeś do tego miejsca i nie jesteś zbyt oszołomiony, to jesteś na dobrej drodze do uchwycenia podstawowych zasad komputera. Możesz się nawet dziwić o co ta cała wrzawa - i będziesz miał rację. Komputer jest urządzeniem bardzo prostym koncepcyjnie i ideowo; tylko wykonuje swoje zadania bardzo szybko (miliony razy na sekundę) i ma ogromną pojemność pamiętania zarówno danych wejściowych, jak i pośrednich wyników wielu tysięcy bardzo prostych sum po drodze do ostatecznego rezultatu.

Jeżeli chcesz studiować teorię swojego komputera, dostępne są dosłownie tysiące książek na ten temat. Niektóre mają tendencje do "robienia wody z mózgu", ale niektóre rzeczywiście poprowadzą cię właściwie odkrywając prostotę i podstawowe zależności, które istnieją pomiędzy systemami liczbowymi i zasadą działania twojego komputera.

Część 2: O CPC6128 w szczególności ...

W części tej przedstawiono niektóre specyficzne właściwości komputera 6128. Podstawowe wiadomości na ten temat można znaleźć w rozdziale "Kurs podstawowy" oraz w rozdziale "Kompletna lista słów kluczowych BASIC AMSTRAD CPC 6128"

W części tej omówiono następujące tematy:

Zestaw znaków
ASCII
Zmienne
Funkcje definiowane przez użytkownika
Formatowanie wydruku
Okna
Przerwania
Dane
Dźwięk
Grafika
Grafika wykorzystująca dodatkową pamięć

Trochę o znakach ...

Używając klawiatury twojego 6128 nie powinieneś przyjmować za rzecz oczywistą, że na ekranie pojawiają się rozpoznawalne litery i cyfry. Przecież już mówiliśmy o tym, że twój komputer nie jest maszyną do pisania. To co rzeczywiście się dzieje jest wynikiem włączania przez ciebie kombinacji przełączników elektrycznych. Sygnały elektryczne powstające wtedy, kiedy naciskasz na klawisz, są tłumaczone przez układy komputera na rozkład punktów świecących na ekranie. Rozpoznajemy ten rozkład jako literę, cyfrę lub inny znak z "zestawu znaków" 6128.

Niektóre znaki nie są bezpośrednio dostępne z klawiatury, lecz wyłącznie przez użycie rozkazu PRINT CHR$ (<numer>). Jest tak, ponieważ każdy element przechowywany w komputerze występuje w jednostce zwanej bajtem - 1, jak powiedziano w części 1. tego rozdziału, bajt może mieć 256 różnych wartości. Ponieważ komputer musi używać całego bajtu na przechowanie jednego znaku (czy tego chcemy czy nie - jest to najmniejsza jednostka jaką uznaje 6128), lepiej byłoby zużytkować wszystkie możliwe kombinacje

(256), niż zadowolić się po prostu około 96 znakami "standardowymi" maszyn do pisania - i zrezygnować z pozostałych 160 możliwości.

"Standardowy" zestaw znaków, nazywany jest "podzestawem". W świecie komputerów występuje on najczęściej w postaci systemu ASCII (American Standard Code for Information Interchange). Jest to głównie system zapewniający "wspólny język" różnych komputerów przy wymianie informacji. W rozdziale zatytułowanym "Nieco użytecznych informacji ..." zamieszczono wykaz znaków ASCII i dodatkowych znaków komputera 6128 wraz z odpowiadającymi im liczbami kodowymi.

Jak się tam dostać ...

Zapoznałeś się już prawdopodobnie z programem:

```
10 FOR n=32 TO 255
20 PRINT CHR$ (n);
30 NEXT
```

... który wyświetla na ekranie zestaw znaków. Zastanówmy się nad zasadniczą treścią tego małego programu.

Pierwszą rzeczą, którą należy zauważyć jest to, że komputerowi nie rozkazano PRINT "abcdefghijklmn... itd."; zamiast tego napisano PRINT CHR$ (n). Tutaj n jest wygodną skrótową literą oznaczającą "zmienną". Zmienna jest pozycją informacji komputerowej, która "zmienia się" stosownie do instrukcji podanych w programie. (Wybór litery n na zmienną jest arbitralny - może to być jakakolwiek litera lub słowo z wyjątkiem ciągu liter tworzących słowo kluczowe).

Jak można rozpoznać zmienną ...

Liczba np. 5 jest stała, znajduje się pomiędzy 4 i 6 - to nie jest zmienna. Znak n jest również stały - jest to jedna z liter alfabetu.

Jak więc komputer rozpoznał różnicę? Gdyby n było zadeklarowane jako znak alfabetu, wpisalibyśmy n w cudzysłowie

"n", na co komputer odpowiedziałby komunikatem "Syntax error" - ponieważ komputer nie rozumie rozkazu FOR "n" = 32 TO 255.

W prosty sposób, przez napisanie n bez cudzysłowu, powiedzieliśmy komputerowi, że n jest zmienną. W definicji rozkazu FOR podano, że za FOR powinna występować zmienna - komputer zakłada, że za FOR jest zmienna, cokolwiek by to nie było.

Powiedzieliśmy również komputerowi, że n=32 TO (do) 255. W ten sposób zadeklarowaliśmy zakres zmiennej. W rzeczywistości jest to sekwencja wartości zaczynająca się od 32, kończąca na 255.

Po zadeklarowaniu zmiennej powinniśmy poinstruować komputer co ma z nią zrobić - czyni to linia 20:

```
20 PRINT CHR$ (n);
```

Określa ona, że dla każdej wartości n komputer powinien znaleźć pamięci znak odpowiadający tej liczbie i wydrukować go na ekranie.

Średnik na końcu linii mówi komputerowi, że nie powinien przesuwać miejsca kolejnego wydruku na początek następnej linii (w przeciwnym wypadku każdy nowy znak będzie drukowany w pierwszej kolumnie nowej linii).

Linia 30 informuje komputer, że po zakończeniu zadania z pierwszą wartością n (32), powinien wrócić do linii, w której jest rozkaz FOR i zrobić to samo z następną (NEXT) wartością zmiennej n. Proces ten znany jest jako "zakładanie pętli" i jest jednym z najbardziej podstawowych i istotnych w programowaniu i działaniu komputera. Oszczędza on wpisywania długich, powtarzających się sekwencji - szybko nauczysz się go stosować przy programowaniu.

Kiedy pętla FOR NEXT dochodzi do krańcowej wartości zadeklarowanego zakresu (255), jej wykonywanie się kończy i komputer szuka następnej linii po linii 30 - ale nie ma takiej, więc po prostu kończy wykonywanie programu i wraca do trybu poleceń bezpośrednich wyświetlając komunikat Ready (gotowe). Dla ciebie jest to znak, że komputer jest gotów przyjmować dalsze instrukcje - możesz na przykład wpisać znowu RUN i powtórzyć wykonanie programu. Program jest bezpiecznie schowany

w pamięci i pozostanie tam dopóty, dopóki nie rozkażesz komputerowi usunąć go - lub nie wyłączysz zasilania.

Program ten trafnie ilustruje podstawową zasadę działania komputera - wszystko co komputer robi odnosi się do liczb. Komputer wyświetlił alfabet - i wszystkie inne znaki - używając liczb odpowiadających znakom. Kiedy naciskasz klawisz A, nie prosisz komputera, żeby napisał A na ekranie, ale każesz mu poszukać w części pamięci zawierającej informację numeryczną pozwalającą wyświetlić A na ekranie. Rzeczywiste miejsce tej informacji jest określone przez kod numeryczny wysyłany z klawiatury w chwili naciśnięcia klawisza. (Każdemu znakowi odpowiada liczba, wykaz znaków i liczb zamieszczono w 3 części rozdziału zatytułowanego "Nieco użytecznych informacji ...").

Również wyświetlanie znaku nie ma nic wspólnego z "pisaniem" litery na ekranie; znowu wszystko związane jest z liczbami.

Na przykład kod ASCII litery A jest 97. Komputer nie rozumie również tego (oporny gałgan, co?), liczba ta musi być przetłumaczona z "ludzkiego" kodu dziesiętnego na kod zrozumiały dla komputera - ogólnie nazywany "kodem maszynowym". Podstawowe pojęcia dotyczące tej strony komputera były omawiane wcześniej w tym rozdziale.

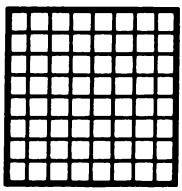
Tłumaczenie z zapisu dziesiętnego, do którego przyzwyczailiśmy się w życiu codziennym, na zapis heksadecymalny wydaje się z początku bardzo trudne. Myślenie o liczbach jako liczbach dziesiętnych jest tak naturalne, że przyjęcie innej podstawy liczenia jest jak jedzenie z nożem w lewej ręce i widelcem w prawej.

Dla zrozumienia zapisu heksadecymalnego trzeba dokonać pewnego wysiłku umysłowego, ale kiedy to zrobisz, wiele spraw się uporządkuje, a elegancja struktury systemu numerycznego stanie się oczywista.

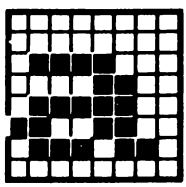
Jeżeli czujesz się niepewnie w zakresie binarnego heksadecymalnego systemu liczbowego, polecamy przeczytać 1 część tego rozdziału (jeżeli jeszcze tego nie zrobiłeś).

Kiedy komputer przetłumaczył już naciśnięcie klawisza A na liczbę, którą rozumie, zwraca się do wskazanej części pamięci, a rezultatem tego jest inny ciąg liczb, które określają znak.

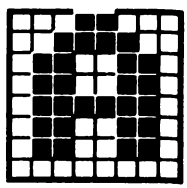
Znak, który jest wyświetlany na ekranie, tworzony jest na podstawie bloku danych przechowywanych w pamięci jako "matryca" liczbowa.



Znak pusty (siatka)



Małe a



Duże A

Matryca składa się z odpowiadających punktom na ekranie elementów ułożonych w rzędach i kolumnach. Znak jest wyświetlany poprzez zaświecanie lub wygaszanie kolejnych punktów na ekranie - każdy punkt jest określony przez dane przechowywane w pamięci komputera. Każda matryca znaku (lub "komórka" na ekranie 6128) ma 8 rzędów i 8 kolumn i, jeżeli w zestawie znaków nie ma znaku jaki chciałbyś mieć, możesz zdefiniować swój własny znak za pomocą rozkazu SYMBOL opisanego w tej części nieco później.

Takie "znaki definiowane przez użytkownika" mogą być tworzone z dowolnej kombinacji 64 punktów - tak więc "kompletny" zestaw znaków, w którym byłyby wszystkie kombinacje matryc, zawierałby o wiele więcej różnych znaków. Dodaj do tego fakt, że możesz grupować bloki znaków, żeby utworzyć większe znaki, a okaże się, że możliwości grafiki "definiowanej przez użytkownika" są ograniczone tylko twoim czasem i pomysłowością.

Funkcje logiczne ...

Główną różnicą pomiędzy kalkulatorem i komputerem jest umiejętność wykonywania przez komputer operacji logicznych na przykład tak jak w sekwencji warunkowej IF THEN. Operatory logiczne traktują wartości, do których się odnoszą jako struktury bitowe i działają na poszczególne bity. Ich opis i użycie

jest całkowicie,... hm, logiczne - ale powszechnie wiadomo jak trudno jest opisać funkcje logiczne w prostych słowach, bez precyzji ścisłych definicji.

Dwie części wyrażenia logicznego znane są jako argumenty. Wyrażenie logiczne zawiera:

<argument> [<operator logiczny> <argument>]

gdzie

<argument> może być: negacją <argumentu>
lub: <wyrażeniem liczbowym>
lub: <wyrażeniem relacji>
lub:(<wyrażeniem logicznym>)

Obydwa argumenty są dla operacji logicznych przekształcone do postaci liczb całkowitych, a w przypadku przekroczenia zakresu liczb całkowitych sygnalizowany jest błąd nr 6.

Operatory logiczne, w kolejności ich priorytetów oraz ich działanie na bity są następujące:

AND Wynik jest równy 0, chyba, że bity obydwu argumentów są 1
OR Wynik jest równy 1, chyba, że bity obydwu argumentów są 0
XOR Wynik jest równy 1, chyba, że bity obydwu argumentów są takie same

AND (i) jest najczęściej eksploatowanym operatorem logicznym i NIE oznacza "dodaj".

```
PRINT 10 AND 10
```

Daje wynik 10

```
PRINT 10 AND 12
```

Daje wynik 8

```
PRINT 10 AND 1000
```

Daje również wynik 8
Jest tak, ponieważ liczby 10 i 1000 są przekształcane do postaci binarnej:

```
        1010
  1111101000
```

Operator AND sprawdza bity na tych samych pozycjach i w wyniku stawia 1 tam, gdzie bity w górnym i dolnym rzędzie są 1:

0000001000

'... co odpowiada liczbie 8. Operator logiczny AND jest stosowany w celu wykrycia jednoczesności dwóch warunków. Oto przykładowy program:

```
10 INPUT "Numer dnia"; dzień
20 INPUT "Numer miesiąca"; miesiąc
30 IF dzień = 25 AND miesiąc = 12 THEN 50
40 CLS: GOTO 10
50 PRINT "Wesołych Świąt"
```

OR (lub) działa również na bitach i wynikiem tego działania jest 1 na pozycji, na której bity obydwu argumentów nie są równe 0. Biorąc te same liczby jak w przypadku AND:

```
PRINT 10 OR 1000
 1002
```

Postać binarna:

```
      1010
  1111101000
```

daje wynik

```
  1111101010
```

Przykładowy program:

```
10 CLS
20 INPUT "Numer miesiąca"; miesiąc
30 IF miesiąc = 12 OR miesiąc = 1 OR miesiąc = 2 THEN 50
40 GOTO 10
50 PRINT "To chyba zima!"
```

Operator NOT (negacja - nie) odwraca każdy bit argumentu (0 przekształca w 1 i odwrotnie)

```
10 CLS
20 INPUT "Numer miesiąca"; miesiąc
30 IF NOT (miesiąc = 6 OR miesiąc = 7 OR miesiąc = 8)
   THEN 50
40 GOTO 10
50 PRINT "To nie może być lato!"
```

Inną ważną i godną uwagi właściwością jest fakt, że można grupować dowolną liczbę warunków logicznych (aż do maksymalnej długości linii) po to, żeby wydestylować daną cechę.

```
10 INPUT "Numer dnia"; dzień
20 INPUT "Numer miesiąca"; miesiąc
30 IF NOT (miesiąc = 12 OR miesiąc = 1) AND dzień = 29
   THEN 50
40 CLS  GOTO 10
50 PRINT "To nie jest ani grudzień, ani styczeń, ale to
   może być rok przestępny"
```

Wynikiem wyrażenia relacji jest -1 lub 0. Postacią binarną -1 jest liczba, której wszystkie bity są 1, dla 0 wszystkie bity są 0. Wynikiem operacji logicznej na takich dwóch argumentach jest również -1 (Prawda) lub 0 (Fałsz).
Sprawdź to przez dodanie do powyższego programu:

```
60 PRINT NOT (miesiąc = 12 OR miesiąc = 1)
70 PRINT (miesiąc = 12 OR miesiąc = 1)
```

... i, w trakcie wykonywania programu wpisz dzień 29 i, powiedzmy miesiąc 2. Otrzymasz odpowiedź z linii 50 i rzeczywiste wartości wyrażeń logicznych z linii 60 i 70.

Wreszcie XOR (eXclusive OR - wyłącznie lub) daje na danej pozycji 1, jeżeli bity obydwu argumentów są różne.

Wszystkie właściwości operacji logicznych podsumowuje poniższa "tablica prawdy". Jest to wygodny sposób ilustracji działania operatorów logicznych.

```
Argument  A   1010
Argument  B   0110
Wynik AND     0010
Wynik OR      1110
Wynik XOR     1100
```

Znaki definiowane przez użytkownika

Jednym z pierwszych zastosowań liczb binarnych, przez które przejdziesz, będzie prawdopodobnie definiowanie nowych znaków za pomocą rozkazu SYMBOL. Skoro znak rysowany jest

w siatce 8 na 8, każdy rząd może być odwzorowany 8-bitową liczbą binarną przez postawienie 1 dla punktu w kolorze atramentu pióra i 0 dla punktu niewidocznego, to jest w kolorze papieru. Osiem takich liczb określa osiem rzędów znaku i liczby te są parametrami rozkazu SYMBOL. Na przykład dla znaku w kształcie domu:

```
     +      = 00001000=&08 =                8        = 8
   ++++     = 00111100=&3C =         32+16+8+4       = 60
  +    +    = 01000010=&42 =      64           +2    = 66
 + +  + +   = 10100101=&A5 = 128   +32      +4  +1 = 165
 +       +  = 10000001=&81 = 128                +1 = 129
 + ++ + +   = 10110101=&B5 = 128   +32+16  +4  +1 = 181
 + ++   +   = 10110001=&B1 = 128   +32+16       +1 = 177
 ++++++++   = 11111111=&FF = 128+64+32+16+8+4+2+1 = 255
```

... rozkaz jest następujący:

```
SYMBOL 240,8,60,66,165,129,181,177,255
```

... lub

```
SYMBOL 240,&08,&3C,&42,&A5,&81,&B5,&B1,&FF
```

... lub

```
SYMBOL 240,&X00001000,&X00111100,&X01000010,&X10100101,
          &X10000001,&X10110101,&X10110001,&X11111111
```

Jeżeli chcesz wydrukować zdefiniowany znak, wpisz:

```
PRINT CHR$ (240)
```

W końcu, jeżeli chcesz stworzyć bloki znaków drukowanych razem, możesz na przykład określić:

```
domy$ = CHR$ (240) + CHR$ (240)
PRINT domy$
```

... lub

```
ulica$ = STRING$ (15,240)
PRINT ulica$
```

Drukowanie ...

PRINT jest jednym z pierwszych rozkazów, które poznaje się w nauce programowania. Jest to jeden z rozkazów BASIC'a, który robi to co jest w nazwie (PRINT - drukować) - czy na

pewno? W rzeczywistości PRINT jest bardziej skomplikowany, niż to się z początku wydaje, na przykład w przypadku określania GDZIE i JAK drukować? ...

Formatowanie wydruku

Rozkaz PRINT może być używany na kilka sposobów. Najprostszym jest umieszczenie za słowem PRINT pozycji, która ma być drukowana. Taka pozycja może być liczbą, tekstem lub nazwą zmiennej.

```
PRINT 3
 3

PRINT "część"
 część

a = 5
PRINT a
 5

a$ = "próba"
PRINT a$
próba
```

W jednym rozkazie PRINT można umieścić kilka pozycji oddzielonych separatorem lub TAB lub SPC. Separatorem może być średnik lub przecinek. Średnik powoduje wydruk pozycji jedna za drugą, przecinek powoduje drukowanie pozycji z przesunięciem do następnej strefy. Początkowa szerokość strefy jest równa 13 znaków, ale może być zmieniona rozkazem ZONE.

```
PRINT 3; -4; 5
 3  -4  5

PRINT "dzień"; "dobry"
dzień dobry

PRINT "dzień", "dobry"
dzień          dobry

PRINT 3, -4, 5
 3             -4             5

ZONE 4
PRINT 3, -4,5
3  -4   5
```

Zwróćmy uwagę, że liczby dodatnie są drukowane ze spacją z przodu, liczby ujemne mają w tym miejscu znak -. Wszystkie liczby drukowane są ze spacją na końcu. Teksty są drukowane dokładnie tak, jak napisane są między cudzysłowami.

Parametrem funkcji SPC jest wyrażenie liczbowe. Zastosowanie tej funkcji powoduje wydrukowanie tylu spacji, ile wynosi wartość tego wyrażenia. Jeżeli wartość jest ujemna, zakłada się 0, a jeżeli wartość jest większa od szerokości strumienia (okna), zakłada się szerokość strumienia.

```
PRINT SPC (5) "cześć"
   cześć

x = 3
PRINT SPC (x*3) "cześć"
       cześć
```

TAB działa podobnie z tym, że powoduje wydrukowanie tylu spacji, ile potrzeba, aby pozycja, która ma być drukowana, pojawiła się w określonej kolumnie.

Jeżeli przed listą pozycji do druku nie występuje wskaźnik kierunkowy strumienia # , wydruk kierowany jest do okna 0. Podanie # strumienia powoduje przesłanie wydruku do innego okna. Strumienie 8 i 9 mają specjalne znaczenie. Skierowanie wydruku do strumienia 8 powoduje drukowanie na zewnętrznej drukarce (jeżeli jest dołączona). Strumień 9 kieruje dane do zbioru na dysku (lub taśmie magnetofonowej). W przypadku taśmy należy raczej używać rozkazu WRITE.

```
PRINT "cześć"
cześć                    - okno 0

PRINT #0, "cześć"
cześć                    - również okno 0



PRINT #4, "cześć"
cześć                    - okno 4
                         (u góry ekranu)
PRINT #8, "cześć"
cześć                    - na drukarce (jeżeli dołączona)
```

TAB i SPC są dobre w przypadku prostego formatowania, w przypadku bardziej szczegółowego formatowania można użyć rozkazu PRINT USING wraz z szablonem formatu. Szablon formatu jest wyrażeniem tekstowym zawierającym specjalne znaki, z których każdy określa poszczególne typy formatu. Znaki te, nazywane "specyfikatorami pola formatu" objaśnione są w opisie słowa kluczowego PRINT USING w rozdziale 3. Tym niemniej następujące przykłady powinny pomóc zrozumieć i stosowanie.

Zacznijmy od formatowania drukowanych tekstów:

```
PRINT USING "\      \"; "tekst próbny"
tekst o
```

"!" może być używane do drukowania pierwszego znaku tekstu.

```
PRINT USING "!"; "tekst próbny"
t
```

... Ale chyba najbardziej użytecznym formatem jest "&" Może on być używany do omijania cechy BASIC'a polegającej na przewidywaniu czy dany tekst zmieści się w linii. Z założenia BASIC drukuje tekst, który nie mieści się w linii, od początku następnej linii. "&"; pozwala to zmienić.
(Ustaw BORDER 0,żeby krańce papieru były widoczne).

```
MODE 1:LOCATE 39,1:PRINT "za długie"

                                        - linia 1
za długie                               - linia 2

MODE 1:LOCATE 39,1:PRINT USING "&"; "za długie"

                                  za    - linia 1
długie                                  - linia 2
```

Wielka liczba szablonów przewidziana jest dla drukowania liczb. Chyba najprostszym jest PRINT USING "#####", gdzie każdy znak "#" odpowiada jednej cyfrze wydruku

```
PRINT USING "#####"; 123
  123
```

Pozycję kropki dziesiętnej określa się znakiem "."

```
PRINT USING "####.#####"; 12.45
  12.45000
```

Cyfry przed kropką dziesiętną mogą być grupowane po 3 z oddzieleniem przecinkiem. Aby to uzyskać należy wpisać w szablon znak "," przed kropką dziesiętną.

```
PRINT USING "########,.###"; 123456.78
  123,456.7800
```

Format może również zawierać znak dolara lub funta (symbol waluty) drukowanego zawsze bezpośrednio przed pierwszą cyfrą liczby, nawet jeżeli nie wypełnia ona całego formatu. Można to uzyskać przez użycie w szablonie znaków "$$" lub "££"

```
PRINT USING "$$##";7
 $7
PRINT USING "$$##"; 351
$351
PRINT USING "££####,.##"; 1234.567
£1,234.57
```

Zwróć uwagę na zaokrąglenie
Spacje przed liczbą mogą być zapełnione gwiazdkami '**', jeżeli w szablonie występuje "**"

```
PRINT USING "**####.#"; 12.22
    12.2
```

Format ten może być łączony ze znakiem waluty (wówczas używa się tylko jednego znaku waluty) - np. "**$..." lub "**£..."
Znak "+" na początku szablonu powoduje drukowanie znaku (+ lub -) przed pierwszą cyfrą. Znak "+" na końcu szablonu powoduje drukowanie znaku (+ lub -) za liczbą.
Znak "-" może być umieszczony tylko na końcu szablonu i powoduje drukowanie znaku minus za liczbami ujemnymi.

```
PRINT USING "+##"; 12
+12
PRINT USING "+##"; -12
-12
PRINT USING "##+"; 12
12+
```

```
PRINT USING "##-"; -12
12-

PRINT USING "## -"; 12
12
```

Znak "↑↑↑↑ " na końcu szablonu powoduje, że liczba jest drukowana w postaci wykładniczej.

```
PRINT USING "###.##↑↑↑↑ "; 123.45
12.35E + 01
```

Jeżeli liczba jest za długa dla danego formatu, przed liczbą drukowany jest symbol %, a rezultat NIE jest skracany do długości szablonu.

```
PRINT USING "####"; 123456
%123456
```

Chcesz mieć swoje okno? ...

BASIC 6128 pozwala na ustanowienie do ośmiu okien tekstowych. Każdy rozkaz kierujący tekst na ekran może następnie korzystać z każdego z tych okien.

Okno ustanawia się rozkazem WINDOW, za którym występuje pięć wartości. Pierwsza wartość, opcjonalna, określa numer okna (od 0 do 7) - jeżeli jest pominięta, przyjmowane jest zero. Wszystkie sygnały i komunikaty BASIC'a (na przykład "Ready") są podawane w oknie zero. Numer okna musi być poprzedzony znakiem #, co identyfikuje liczbę jako odnoszącą się do strumienia. Następne cztery liczby określają kolejno lewą, prawą, górną i dolną krawędź okna. Wartości te są numerami kolumn i rzędów ekranu więc zakres tych wartości wynosi 1 do 80 dla kolumn (lewa/prawa) i 1 do 25 dla rzędów (góra/dół).

Poniższy przykład definiuje okno (strumień) numer 4 leżące między kolumnami 7 i 31 oraz rzędami 6 i 18. Wyzeruj komputer i wpisz:

```
WINDOW #4,7,31,6,18
```

Nic się nie zdarzy po tym rozkazie, ale spróbuj teraz wpisać:

```
INK 2,9
PAPER #4,3
CLS #4
```

Spowoduje to, że na ekranie ukaże się wielki zielony prostokąt i to jest okno numer 4. Przykład powyższy pokazuje również, że rozkazy PAPER i CLS mogą dotyczyć któregokolwiek z ośmiu okien, według wskazania numeru kierunkowego strumienia; jego pominięcie powoduje, że rozkaz działa na okno 0 - okno przyjmowane domyślnie.

Każdy z następujących rozkazów może zawierać numer kierunkowy strumienia identyfikujący okno, którego rozkaz dotyczy.

CLS
COPYCHR$
INPUT
LINE INPUT
LIST
LOCATE
PAPER
PEN
POS
PRINT
TAG
TAGOFF
VPOC
WINDOW
WRITE

Nowe, zielone okno,które założyłeś na ekranie, zaciemni część wcześniej napisanego tekstu (w oknie numer 0).

Tekst może być kierowany do dowolnego okna przez umieszczenie w rozkazie PRINT numer kierunkowego strumienia

```
PRINT #4, "jak leci"
```

Słowa te pojawią się raczej u góry zielonego prostokąta niż w kolejnej linii, jak byłoby gdybyś wpisał:

```
PRINT "jak leci"
```

Podczas wpisywania poprzedniego rozkazu zauważyłeś, że część zielonego okna została zapisana tekstem.
Jeżeli chcesz, żeby wszystkie komunikaty BASICA pojawiały się w oknie 4, możesz okno 4 zamienić z oknem 0 za pomocą rozkazu WINDOW SWAP

```
WINDOW SWAP 0,4
```

"Ready", które pojawi się po tym rozkazie, zostanie wydrukowane w zielonym oknie. Kursor będzie umieszczony bezpośrednio niżej. Teraz wpisz następujące:

```
PRINT #4, "jak leci"
```

..., a słowa "jak leci" pojawią się bezpośrednio poniżej rozkazu WINDOW SWAP, w starym oknie 0, które teraz jest oknem numer 4. Z przykładu tego wynika również, że bieżąca pozycja drukowania w każdym oknie jest pamiętana, tak, że nawet po zamianie okien tekst drukowany jest w następnej linii, a nie od góry okna. Spróbuj jak działa następujący ciąg rozkazów:

```
LOCATE #4,20,1
PRINT "to jest okno 0"
PRINT #4"to jest okno 4"
```

Komunikat "okno 0" pojawi się w linii następnej po PRINT, podczas gdy komunikat "okno 4" pojawi się w środku górnej linii całego ekranu.

Przed wydaniem rozkazu WINDOW wszystkie 8 okien pokrywa się z całym ekranem. Tak samo jest po wydaniu rozkazu MODE - więc, jeżeli po założeniu okna, okaże się, że jest ono za małe, po prostu wpisz MODF 1 jak poniżej:

```
MODE 1
WINDOW 20,21,7,18
MO
DE
 1
```

Nie martw się, że słowo "MODE" rozpadło się - będzie działać, nie zapomnij tylko o spacji pomiędzy MODE a 1

Teraz, skoro już wiesz co nieco o funkcjonowaniu okien, spróbuj wpisać następujący krótki program:

```
10 MODE 0
20 FOR n=0 TO 7
30 WINDOW #n,n+1,n+6,n+1,n+6
40 PAPER #n, n+4
50 CLS #n
60 FOR c=1 TO 200: NEXT c
70 NEXT n
```

Program ten ustanawia 8 zachodzących na siebie okien i czyści każde innym kolorem papieru. Kiedy program się skończy i pojawi się "Ready", naciśnij kilkakrotnie ENTER , żeby zobaczyć jak skrolowanie okna 0 oddziaływuje na kolorowe bloki na ekranie. Tym niemniej, chociaż te kolorowe bloki mogą być skrolowane, rozmieszczenie pozostałych okien w rzeczywistości się nie zmieni. Spróbuj teraz:

```
CLS #4
```

..., a zobaczysz, że okno 4 jest nadal w tym samym miejscu - - na nowo zakolorowany blok, zacierający znajdujące się poniżej, jak można się było spodziewać. Dla ciekawości obserwuj różnice wpisując rozkazy:

```
LIST
LIST #4
LIST #3
```

Dalszą właściwością rozkazu WINDOW, demonstrowaną w końcowym programie tego podrozdziału, jest to, że nieważne jest czy podajesz w rozkazie najpierw lewą skrajną kolumnę, a potem prawą, czy odwrotnie. Jeżeli pierwszy parametr jest większy od drugiego, BASIC automatycznie dokonuje przestawienia (w sposób niewidoczny). To samo dotyczy parametrów określających górny i dolny skraj okna.

```
10 MODE 0
20 a=1+RND*19:b=1+RND*19
30 c=1+RND*24:d=1+RND*24
40 e=RND*15
50 WINDOW a,b,c,d
60 PAPER e:CLS
70 GOTO 20
```

Jeżeli mogę przerwać ...

Jeżeli jeszcze tego nie zauważyłeś, należy powiedzieć, że główną innowacją programową komputerów AMSTRAD jest ich zdolność obsługi przerwań z BASIC'a - co oznacza, że AMSTRAD BASIC jest zdolny do wykonywania kilku różnych operacji jednocześnie w zakresie danego programu (BASIC "wielozadaniowy").

Właściwość tę uzyskano wprowadzając rozkazy AFTER i EVERY. Właściwość ta przejawia się również w sposobie sterowania dźwiękiem poprzez ułatwienia takie jak kolejki dźwiękowe i rendez-vous.

Wszystkie właściwości związane z odmierzaniem czasu wynikają z działania głównego zegara systemu, którym jest układ czasowy synchronizowany generatorem kwarcowym znajdujący się wewnątrz komputera. Zegar ten daje taktowanie i synchronizację wszystkich układów komputera - na przykład układu procesora czy układu przesuwania plamki świetlnej ekranu. Każde działanie sprzętu związane z czasem ma swój początek w zegarze głównym.

Wykorzystanie programowe znalazł zegar w rozkazach AFTER (po) i EVERY (co), które, zgodnie z założeniem przystępności BASIC'a dla użytkownika, robią dokładnie to co mają w nazwie; to jest PO czasie (lub CO jakiś czas) zadanym w rozkazie,BASIC przechodzi do określonego podprogramu i wykonuje zadanie tam zaprogramowane.

6128 ma zegar czasu rzeczywistego. Rozkaz AFTER pozwala na zorganizowanie programu napisanego w języku BASIC w postaci podprogramów, które mają być wywołane za jakiś czas, w przyszłości. Istnieją cztery liczniki czasu dostępne dla programisty, z każdym z nich może być skojarzony inny podprogram.

Kiedy mija podany czas, automatycznie wywoływany jest właściwy podprogram, tak, jakby w aktualnie wykonywanym miejscu programu był rozkaz GOSUB. Po zakończeniu podprogramu (rozkazem RFTURN) BASIC wraca do programu głównego, do miejsca, w którym nastąpiło przerwanie.

Rozkaz EVERY pozwala na zorganizowanie programu w postaci podprogramów, które mają być wywoływane wielokrotnie w regularnych odstępach części. Również w tym wypadku dostępne są (te same) cztery liczniki i każdy z nich może wywoływać swój podprogram.

Liczniki mają różne priorytety przerwań. Najwyższy priorytet ma licznik 3, najniższy - licznik 0 (patrz rozdział zatytułowały "Nieco użytecznych informacji ...").

```
 10 MODE 1:n=14:x=RND#400
 20 AFTER x,3 GOSUB 80
 30 EVERY 25,2 GOSUB 160
 40 EVERY 10,1 GOSUB 170
 50 PRINT "wypróbuj swój refleks"
 60 PRINT "naciśnij spację.";
 70 IF ślad = 1 THEN END ELSE 70
 80 t = REMAIN (2)
 90 IF INKEY (47) = -1 THEN 110
100 SOUND 1,900:PRINT "oszust!": GOTO 150
110 SOUND 129,20: PRINT "TERAZ": t=TIME
120 IF INKEY (47) = -1 THEN 120
130 PRINT  "potrzebowałeś na to";
140 PRINT (TIME - t)/300; "sekundy"
150 CLEAR INPUT: ślad = 1:RETURN
160 SOUND 1,0,50: PRINT".";:RETURN
170 n=n+1:IF n > 26 THEN n=14
180 INK 1, n:RETURN
```

Rozkazy AFTER i EVERY mogą być wydawane w każdej chwili, kasując czas i podprogram skojarzone z danym licznikiem, wcześniejszym rozkazem AFTER lub EVERY. Liczniki dla AFTER i dla EVERY są te same, tak więc AFTER unieważnia wcześniejsze EVERY dla danego licznika i odwrotnie.

Rozkazy DI i EI odpowiednio uniemożliwiają i umożliwiają przerwania przez liczniki, w czasie wykonywania programu pomiędzy nimi. Może to dawać efekt opóźnienia przerwania o wyższym priorytecie, jeżeli przerwanie to nastąpi w czasie wykonywania podprogramu obsługi przerwania o niższym priorytecie. Funkcja REMAIN blokuje dany licznik i podaje stan licznika w chwili zablokowania. (Liczniki działają rewersyjnie, przerwanie następuje w chwili osiągnięcia stanu 0).

Korzystanie z danych...

W programie, który zawsze korzysta z tego samego zestawu informacji, sensowniej jest wprowadzić wszystkie wartości na raz, niż zmuszać użytkownika do wpisywania ich sukcesywnie. Udogod-

nienie to zapewniają rozkazy READ i DATA. Rozkaz READ jest bardzo podobny do INPUT w tym, że obydwa są używane do przypisywania zmiennym wartości. Różnica polega na tym, że READ czerpie wartości z linii programu oznaczonych słowem DATA, a INPUT z klawiatury. Pokazują to dwa następujące przykłady:

```
10 INPUT "wpisz 3 liczby oddzielone przecinkiem"; a,b,c
20 PRINT "Te liczby to: ";a; "i"; b; "i"; c
run
```

```
10 READ a,b,c
20 PRINT "Te liczby to ";a; "i"; b; "i"; c
30 DATA 12,14,21
run
```

Tak samo jak w rozkazie INPUT tak i w rozkazie DATA, poszczególne pozycje są oddzielone przecinkami.
W liniach DATA mogą występować również stałe tekstowe.

```
10 DIM a$ (8)
20 FOR i=0 TO 8
30 READ a$ (i)
40 NEXT
50 FOR i=0 TO 8
60 PRINT a$ (i); " ";
70 NEXT
80 DATA Mały, szybki, brązowy, lis, przeskoczy, przez,
        dużego, leniwego, psa
```

Zauważ, że chociaż w linii DATA są teksty, nie są one ujęte w podwójne cudzysłowy. Użycie cudzysłowów jest w rozkazie DATA opcjonalne (tak samo jak w odpowiedzi na rozkaz INPUT). Jest jedna sytuacja, w której podwójny cudzysłów jest użyteczny: wtedy, kiedy tekst sam zawiera przecinki. Gdyby teksty w tej sytuacji nie były oddzielone cudzysłowami, rozkaz READ traktowałby przecinki jako separatory.

```
10 READ a$
20 WHILE a$
30 PRINT a$
40 READ a$
50 WEND
```

```
60 DATA Stary, zdezelowany, spróchniały dom zwalił się
       na wietrze
70 DATA "Wysoki, szczupły, ciemnowłosy mężczyzna
       zakasłał głośno"
80 DATA "*"
```

Tekst w linii 60 zawiera przecinki, więc każda część będzie czytana i drukowana oddzielnie. Natomiast tekst w linii 70, podany w cudzysłowie, wydrukowany zostanie w całości w jednej linii.

Powyższy przykład ilustruje również fakt, że dane mogą być pisane w kilku liniach. Rozkaz READ powoduje wczytywanie danych w kolejności zgodnej z kolejnością linii (60,70,80). Innym faktem, mniej oczywistym być może, jest to, że linie DATA mogą być rozmieszczone w programie gdziekolwiek, nawet przed rozkazem READ, który czyta z tych linii.

Jeżeli w programie jest więcej rozkazów READ, każdy następny kontynuuje czytanie od miejsca, do którego dotarł poprzedni.

```
10 DATA 123,456,789,521,654,2343
20 FOR i=1 TO 5
30 READ liczba
40 suma = suma + liczba
50 NEXT
60 READ suma 2
70 IF suma = suma 2 THEN PRINT "dane są w porządku"
   ELSE PRINT "w danych jest błąd"
```

Spróbuj zmienić jedną z 5 pierwszych liczb, a następnie uruchomić program powtórnie. Technika dodawania, na końcu ciągu DATA, liczby będącej sumą wszystkich liczb poprzednich jest dobrym sposobem detekcji błędów w danych, szczególnie jeżeli w programie jest dużo linii DATA. Liczba taka nazywana jest sumą kontrolną (check sum).

Jeżeli w programie czytane są na przemian liczby i teksty, dozwolone jest mieszanie danych liczbowych i tekstowych w rozkazach READ i DATA, o ile dane te są prawidłowo czytane. Na przykład, jeżeli DATA zawiera sekwencję dwóch liczb, za którymi jest tekst, wówczas w rozkazie READ muszą kolejno wystąpić dwie zmienne liczbowe i zmienna tekstowa:

```
10 DIM a(6), b(5), s$(5)
20 FOR i=1 TO 5
30 READ a(i), b(i), s$(i)
40 NEXT
50 DATA 1,7,fredek,3,9,jurek,2,2,eryk,4,6,piotr,9,1,alfons
60 FOR i=1 TO 5
70 PRINT s$(i),":";a(i)*b(i)
80 NEXT
```

Można też rozdzielić zmienne różnych typów:

```
10 DIM a(5), b(5), s$(5)
20 FOR i=1 TO 5
30 READ a(i), b(i)
40 NEXT
50 FOR i=1 TO 5
60 READ s$(i)
70 NEXT
80 DATA 1,7,3,9,2,2,4,6,9,1
90 DATA fredek, jurek, eryk, piotr, alfons
100 FOR i=1 TO 5
110 PRINT s$(i),":";a(i)*b(i)
120 NEXT
```

Jeżeli linię 20 zmienić na:

```
20 FOR i=1 TO 4
```

... wówczas dwa pierwsze odczyty w linii 60 będą "0" i "1". Są to oczywiście dobre wartości dla zmiennych tekstowych, ale rezultat nie będzie taki, jak zaplanowano. Jedną z metod przywrócenia właściwego działania programu jest dopisanie:

```
15 RESTORE 80
45 RESTORE 90
```

Rozkaz RESTORE przesuwa "wskaźnik pozycji czytanej danej na początek linii o podanym numerze i dlatego może być używany w instrukcjach warunkowych do wyboru bloku danych w zależności od pewnych kryteriów. Na przykład w grze o różnych stopniach trudności, o różnych planszach, dane dla każdej planszy mogą być pobrane w zależności od pewnej zmiennej - na przykład "poziom". Ilustruje to następujący program.

```
1000 REM fragment rysowania ekranu
1010 IF poziom = 1 THEN RESTORE 2010
1020 IF poziom = 2 THEN RESTORE 2510
1030 IF poziom = 3 THEN RESTORE 3010
1040 FOR y=1 TO 25
1050 FOR x=1 TO 40
1060 READ znak
1070 LOCATE x,y: PRINT CHR$ (znak)
1080 NEXT x,y
.
.
2000 REM dane dla planszy 1
2010 DATA 200, 190, 244, 244, 210 ... itd.
.
.
2500 REM dane dla planszy 2
2510 DATA 100, 103, 245, 243, 251 ... itd.
.
.
3000 REM dane dla planszy 3
3010 DATA 190, 191, 192, 193, 194 ... itd.
```

Innym przykładem użycia DATA, READ i RESTORE jest program grający melodię. Wartości okresów są czytane z instrukcji DATA, a RESTORE jest używane wtedy, kiedy należy powtórzyć fragment melodii.

```
 10 FOR i=1 TO 3
 20 RESTORE 100
 30 READ nuta
 40 WHILE nuta <> -1
 50 SOUND 1, nuta, 35
 60 READ nuta
 70 WEND
 80 NEXT
 90 SOUND 1,142,100
100 DATA 95,95,142,127,119,106
110 DATA 95,95,119,95,95,119,95
120 DATA 95,142,119,142,179,119
130 DATA 142,142,106,119,127,-1
```

Niech brzmi muzyka ...

Ze wszystkich właściwości 6128, rozkazy dotyczące dźwięku i obwiedni wydają się na początku najbardziej zniechęcające - a przecież nie musi tak być. Wystarczy odrobina wprawy, żeby wytwarzać cały szereg różnego rodzaju hałasów, a nawet programować wielogłosowe melodie.

Zacznijmy od 4 pierwszych parametrów rozkazu SOUND. Są to: numer kanału, okres tonu, czas trwania nuty i głośność.

Pomińmy na razie numer kanału, który to parametr omówimy trochę później. Okres tonu musi być liczbą całkowitą od 0 do 4095, ale tylko niektóre z tych wartości odpowiadają rzeczywiście nutom ze skali muzycznej - wartości te podano w wykazie zamieszczonym w rozdziale zatytułowanym "Nieco użytecznych informacji ..." Na przykład 239 odpowiada środkowemu C, a 253 - nucie B poniżej środkowego C natomiast liczby od 240 do 252 dają tony pośrednie, które nie istnieją w skali fortepianowej. Jeżeli okres tonu jest równy 0, wówczas nie jest wydawany żaden ton - przydaje się to przy używaniu parametru "szum" (omawianego później).

Czas trwania nuty podawany jest w jednostkach 0,01 sekundowych. W zasadzie wartość tego parametru może być w zakresie 1 do 32767. Tym niemniej, wartość 0 mówi o tym, że czas trwania jest określony przez użytą "obwiednię" (o tym później), a wartość ujemna mówi o tym, że obwiednia powinna być powtórzona daną liczbę razy, np. -3 oznacza: "powtórz obwiednię głośności trzy razy" (to również później).

Wartość głośności może być od 0 do 15 (jeżeli parametr ten jest pominięty, przyjmowane jest 12). Dla prostych dźwięków, omawianych do tej pory, głośność pozostaje stała przez cały czas trwania dźwięku. W przypadku używania "obwiedni głośności" do modulacji głośności, parametr ten jest początkową wartością głośności danej nuty.

Wróćmy teraz do numeru kanału. Jest to wartość, której działanie tależy od poszczególnych bitów - dlatego powinieneś wiedzieć coś o liczbach binarnych, żeby w pełni zrozumieć ten parametr (patrz część 1. tego rozdziału).

Dźwięki są wytwarzane w trzech kanałach. Jeżeli komputer jest dołączony do wzmacniacza stereofonicznego, dźwięki z kanału pierwszego (A) brzmią z lewej strony, dźwięki z kanału drugiego (B) brzmią z prawej strony, a dźwięki z kanału trzeciego (C) brzmią ze środka. Przypisane kanałom numery są następujące:

1 kanał A
2 kanał B
4 kanał C

W celu wytworzenia dźwięku w kilku kanałach jednocześnie należy dodać do siebie ich numery. Na przykład dla dźwięku z kanału A i C należy napisać 1+4 = 5.

```
SOUND 5,284
```

Być może dziwisz się dlaczego kanał C ma numer 4, a nie 3. Wynika to z tego, że numery te są kolejnymi potęgami 2 (1=2↑0, 2=2↑1, 4=2↑2). Oznacza to, że dany kanał włączany jest określonym bitem reprezentacji binarnej tego parametru. Najmniej znaczący bit steruje (1 włącza, 0 wyłącza) kanałem A. Wracając do przykładu:
5 odpowiada 1*4+0*2+1*1, w zapisie binarnym 101. Każda pozycja (bit) liczby binarnej określa jeden kanał:

C B A
1 0 1

Innymi słowy kanał C jest włączony, kanał B wyłączony, a kanał A włączony. Gdyby nuta miała być grana w kanale A i B, wyglądałoby to tak:

C B A
0 1 1

Liczba binarna 011 odpowiada w zapisie dziesiętnym 0*4+1*2+1*1=3. Rozkaz SOUND byłby wtedy na przykład:

```
SOUND 3,142
```

Oczywiście jest to ta sama wartość, jaką otrzymuje się z dodania numerów kanałów (pamiętaj A=1, B=2, C=4). Tak więc, aby grały kanały A i B, numer kanału powinien być 1+2=3.

Jeżeli nie wszystko zrozumiałeś - nie martw się. Dopóki rozumiesz, że dana kombinacja kanałów może być wybrana przez dodanie numerów czynnych kanałów, dopóty wszystko jest w po-

Niestety w numerze kanału zawarte są również inne informacje. Liczby 8 (bit 2⁄3), 16 (bit 2⁄4) i 32 (bit 2⁄5) wskazują, że dźwięk powinien współbrzmieć (rendez vous) z innym dźwiękiem wytworzonym odpowiednio w kanale A, B lub C. Prawdopodobnie dziwisz się co kryje się za terminem "rendez vous". No tak, jak dotychczas wszystkie nasze dźwięki odnosiły się bezpośrednio do danego kanału. Spróbuj:

```
SOUND 1,142,2000
SOUND 1,90,200
```

Na pewno udało ci się wpisać drugi rozkaz, zanim działanie pierwszego ustało. Mimo to drugi dźwięk również zabrzmiał. Jest tak dlatego, że system dźwiękowy może przyjąć do 5 rozkazów dźwiękowych w każdym kanale. Dźwięki te czekają w "kolejkach dźwiękowych" na swoją kolej. Jeżeli chcemy wytworzyć dźwięk w kanale A, a potem dwa jednocześnie brzmiące dźwięki w kanale A i B, musimy w jakiś sposób powiedzieć komputerowi, że dźwięk w kanale B nie powinien zacząć się przed zakończeniem pierwszego dźwięku w kanale A - jeden kanał powinien czekać na drugi. To właśnie jest nazywane "rendez vous" i może być osiągnięte dwoma sposobami:

```
SOUND 1,200,1000
SOUND 3,90,200
```

Druga nuta jest tutaj skierowana do A i B, nie może się więc ona zacząć zanim skończy się pierwsza nuta. Ograniczeniem tej metody (polegającej na podaniu wszystkich kanałów, w których nuta ma być grana) jest to, że każdy kanał wytwarza ten sam dźwięk. Drugi sposób jest następujący:

```
SOUND 1,200,2000
SOUND 1+16,90,200
SOUND 2+8,140,400
```

Tutaj druga nuta kanału A ma "rendezvous" z nutą kanału B (a nuta kanału B ma "rendezvous" z drugą nutą kanału A). Zaletą tego sposobu jest jasna - chociaż druga nuta kanału A jest inna niż nuta kanału B, zostały one połączone, tak, że żadna z nich nie może się zacząć, zanim obydwa kanały nie będą wolne. To właśnie jest "rendezvous". Efektem jest współbrzmienie. I znowu wartości są kodowane bitami:

8=2↑3, 16=2↑4, 32=2↑5

... jak dotąd numer kanału można rozpatrywać jako liczbę binarną, w której poszczególne bity oznaczają:

Rendezvous C	Rendezvous B	Rendezvous A	Gra kanał C	Gra kanał B	Gra kanał A
dodaj 32	dodaj 16	dodaj 8	dodaj 4	dodaj 2	dodaj 1

Dla nuty, której źródłem ma być kanał C i która ma "rendezvous" z kanałem A, powinno być:

0	0	1	1	0	0

Jest to liczba binarna 001100 równa 8+4=12
W ten sposób numer kanału równy 12 powie komputerowi, że ma zagrać nutę w kanale C czekając jednak na nutę, która grana w kanale A ma "rendezvous" z kanałem C.

Dodanie 64 (2↑6) do numeru kanału powoduje zatrzymanie nuty w kanale dźwiękowym. W efekcie nuta nie będzie grana aż do chwili wysłania rozkazu RELEASE.

Wreszcie dodanie 128 (2↑7) do numeru kanału powoduje wyczyszczenie kanału (usunięcie kolejki). Dlatego szybkim sposobem przerwania dźwięku, który trwa zbyt długo jest wysłanie takiego rozkazu:

```
SOUND 1,248,30000     (to będzie grane przez 5 minut)
SOUND 1+128,0         (to spowoduje przerwanie dźwięku)
```

W trybie rozkazów bezpośrednich szybszym sposobem przerwania jakiegokolwiek dźwięku jest naciśnięcie klawisza DEL na początku linii: krótki, ostrzegający sygnał dźwiękowy usuwa kolejki ze wszystkich kanałów.

Teraz, kiedy już potrafimy wysyłać dźwięki do dowolnego z trzech kanałów (z"rendezvous" jeżeli trzeba), miło byłoby umieć grać nieco więcej niż raczej niemelodyjne "piip", jakie wytwarza prosty rozkaz SOUND. Sposobem na to jest modulacja dźwięku. Robi się to przez zaprojektowanie obwiedni - wzoru, według którego zmienia się głośność w tym krótkim czasie, kiedy nuta jest grana. Nuta grana na instrumencie tradycyjnym ma na początku szybki wzrost głośności, potem brzmi trochę ciszej przez jakiś czas, wreszcie cichnie zupełnie. Możliwe jest wytworzenie takiej obwiedni głośności nuty granej rozkazem SOUND. Służy do tego rozkaz ENV. Rozważmy najpierw prosty przykład:

```
ENV 1,5,3,4,5,-3,8
SOUND 1,142,0,0,1
```

Rozkaz ENV musi być wydany przed rozkazem SOUND, w którym obwiednia jest wykorzystana. W celu zastosowania w rozkazie SOUND obwiedni o danym numerze, numer ten musi występować w rozkazie SOUND jako piąty parametr - w tym przykładzie numer obwiedni jest równy 1. Numer obwiedni definiowany jest pierwszym parametrem rozkazu ENV. Kolejne parametry tego rozkazu określają czas trwania obwiedni i głośność, dlatego odpowiednie parametry w SOUND są równe 0. Zdefiniowana w przykładzie obwiednia powoduje, że głośność zwiększa się w 5 krokach, przy czym każdy krok to zwiększenie głośności o 3, a czas trwania kroku jest równy 4 setne sekundy. Następnie głośność zmniejsza się również w 5 krokach, o 3 w każdym kroku, przy czym każdy krok trwa 8 setnych sekundy. Innymi słowy pierwsza liczba rozkazu ENV określa numer obwiedni, a następne występują w trzyliczbowych sekcjach, w których pierwsza liczba określa przez ile kroków ma być zmieniana głośność, druga liczba określa o ile ma się zmienić głośność w każdym kroku, a trzecia liczba określa jak długo ma trwać jeden krok.

Całkowity czas trwania wykonywania jednej sekcji jest równy iloczynowi pierwszej liczby (liczba kroków) przez trzecią (czas przerwy). Całkowity przyrost głośności jest równy iloczynowi liczby kroków przez wartość kroku. Całkowity czas trwania obwiedni, która ma więcej niż jedną sekcję, jest równy sumie czasów trwania poszczególnych sekcji.

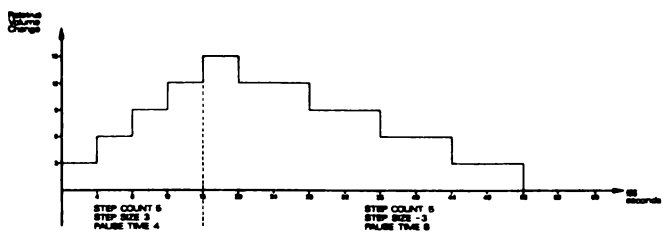
Oczywiście wartość początkowa głośności nie zawsze musi być równa 0 (jak w naszym przykładzie). W powyższym przykładzie głośność była zwiększona, a potem zmniejszana. W następnym przykładzie jest odwrotnie:

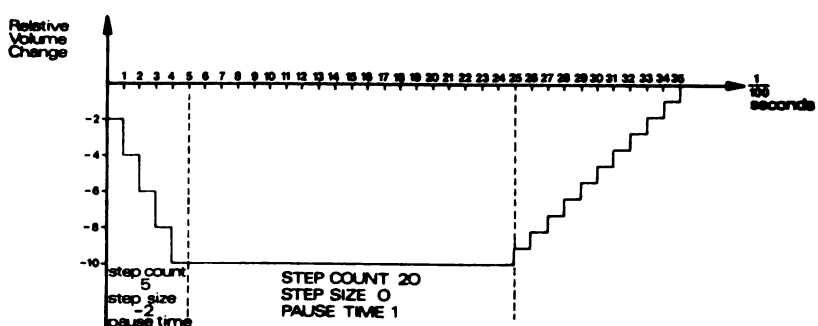
```
ENV 2,5,-2,1,20,0,1,10,1,1
```

Ta obwiednia ma numer 2 i zawiera 3 sekcje. W pierwszej głośność jest zmniejszana w 5 krokach co 2. Długość każdego kroku wynosi jedną setną sekundy. Druga sekcja składa się z 20 kroków o zerowym przyroście (głośność jest stała). Również tu długość każdego kroku jest 0,01 sekundy. Wreszcie trzecia sekcja składa się z 10 kroków, z których każdy zwiększa głośność o 1 i znowu długość kroku jest 0,01 sekundy.

Rozkaz SOUND podaje wartość początkową głośności równy 15. Pierwsza sekcja zmniejsza głośność do poziomu 5, druga utrzymuje ją przez 0,20 sekundy, trzecia zwiększa z powrotem do 15.

Może trochę trudno jest wyobrazić sobie kształt obwiedni. Wygodnie jest rysować wpierw kształt obwiedni na papierze kratkowanym, a następnie przepisać w rozkazie ENV parametry z rysunku. Następujące rysunki pokazują kształty obwiedni zaprojektowanych wcześniej.

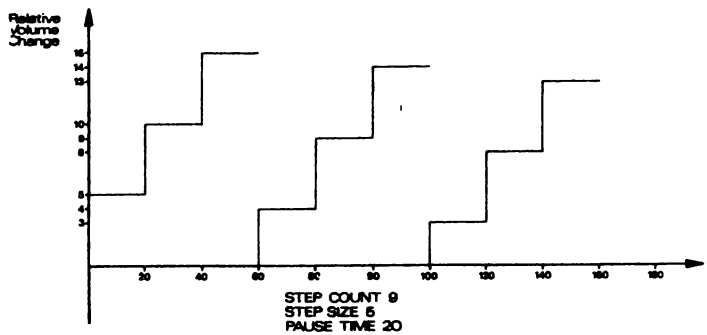




Maksymalna liczba sekcji w obwiedni jest równa 5, w każdej sekcji 3 wartości, co daje razem maksymalną liczbę parametrów w rozkazie ENV równą 16 (wliczając numer obwiedni, który musi być w zakresie 1 do 15). Jeżeli kolejne kroki powodują przekroczenie wartości głośności 15 lub 0, odejmowane lub dodawane jest 16; krok powyżej 15 daje 0, krok poniżej 0 daje 15:

```
FNV 3,9,5,20
SOUND 1,142,0,0,3
```

Ta prosta obwiednia daje 9 kroków, każdy krok zwiększa głośność o 5, każdy krok trwa 0,20 sekundy. Po pierwszych trzech krokach głośność jest 15, następny krok powoduje ustawienie 4 (20-16), potem 9 itd. Poniższy diagram pokazuje działanie:



Liczba kroków może być równa od 0 do 127. Wartość kroku może być od -128 do +127 (wartości ujemne zmniejszają głośność) a czas przerwy (to znaczy czas pomiędzy krokami) może być w zakresie od 0 do 255.

Kiedy już potrafimy tworzyć obwiednię głośności dla uzyskania charakterystycznego kształtu, możemy również chcieć określić charakterystyczny wzór tonu; na przykład wibrato - gdzie ton wibruje dokoła głównego tonu nuty.

Robi się to w sposób bardzo podobny do tworzenia obwiedni głośności. Obwiednie tonu są wytwarzane rozkazami ENT. Na przykład:

```
ENT 1,5,1,1,5,-1,1
SOUND 1,142,10,15,,1
```

Obwiednia tonu wywoływana jest z rozkazu SOUND, jeżeli szóstym parametrem rozkazu jest numer obwiedni. Również w przypadku obwiedni tonu rozkaz ENT musi być wydany przed wywołującym rozkazem SOUND.

W tym pierwszym przykładzie obwiednia tonu numer 1 powoduje wykonanie 5 kroków, z których każdy zwiększa okres tonu o 1 i trwa 1 setną sekundy. Następna sekcja obwiedni znowu powoduje wykonanie 5 kroków o tej samej długości, ale zmniejszających okres tonu o 1. Całkowity czas trwania obwiedni wynosi 5+5=10 setnych sekundy. Zwróć uwagę, że ta sama wartość została

podana jako 3 parametr rozkazu SOUND, ponieważ obwiednia tonu NIE determinuje czasu grania nuty (tak jak to robiła obwiednia głośności). Jeżeli czas trwania podany w SOUND jest krótszy od długości obwiedni tonu, wówczas końcowa część obwiedni jest nie wykorzystana. Jeżeli jest odwrotnie, wówczas końcowa część nuty jest grana stałym tonem. Dotyczy to również przypadku, gdy czas trwania nuty określa obwiednia głośności. (Zauważ, że w powyższym przykładzie w rozkazie SOUND nie występuje numer obwiedni głośności - głośność tej nuty pozostaje stała).

Większość obwiedni tonu trwa krócej niż oczekiwana długość nuty. Dlatego obwiednia może być zdefiniowana jako powtarzalna i jest wtedy powtarzana przez cały czas trwania nuty. W tym celu należy przypisać danej obwiedni tonu numer ujemny, w wywołującym rozkazie SOUND znak minus pomijamy:

```
ENT -5,4,1,1,4,-1,1
SOUND 1,142,100,12,,5
```

Daje to nucie efekt wibrato. Przy definiowaniu obwiedni tonu zazwyczaj jest najlepiej, jeżeli daje ona zmiany tonu symetryczne względem tonu podstawowego. Zapobiega to, przy powtarzaniu obwiedni, przesuwaniu częstotliwości tonu coraz dalej w jedną stronę. Spróbuj tego:

```
ENT -6,3,1,1
SOUND 1,142,90,12,,6
```

Wysokość tonu tego dźwięku spada w dół dość drastycznie, ponieważ przy każdym powtórzeniu obwiedni okres jest zwiększany o 3 i dzieje się tak 30 razy (90/3). Taki efekt jest użyteczny przy imitacji syren itp.

```
ENT -7,20,1,1,20,-1,1
SOUND 1,100,400,12,,7

ENT -8,60,-1,1,60,1,1
SOUND 1,100,480,12,,8
```

Liczba możliwych obwiedni jest 15 (zakres 1 do 15), przy czym znak minus oznacza powtarzalność obwiedni. Liczba kroków (pierwsza liczba w każdej trzyliczbowej sekcji) ma zakres 0 do 239. Podobnie jak w obwiedni głośności wartość kroku może być

w zakresie od -128 do 127, a czas przerwy (między krokami) w zakresie 0 do 255. Rozkaz ENT, tak jak ENV, może mieć do 5 sekcji (każda sekcja - trzy liczby).

Ostatnim parametrem, który może być dodany do rozkazu SOUND jest liczba, która wskazuje poziom szumu towarzyszącemu dźwiękowi. O jednej rzeczy należy pamiętać przy stosowaniu szumu, mianowicie o tym, że jest tylko jeden kanał szumu i każdy kolejny okres szumu wpisywany w ten kanał zastępuje poprzedni.

Szum może być dodany do tonu tworząc nowy dźwięk, lub może być użyty oddzielnie przez przyjęcie okresu tonu (drugi parametr) w rozkazie SOUND równego 0, tak, że tylko szum tworzy dźwięk. Używane jest to do tworzenia dźwięków typu perkusyjnego. Spróbuj:

```
ENT -3,2,1,1,2,-1,1
ENV 9,15,1,1,15,-1,1
FOR a=1 TO 10: SOUND 1,4000,0,0,9,3,15:NEXT
```

Sekwencja ta formuje podstawę odgłosów pociągu. Zauważ, że użyto tu obydwu rodzajów obwiedni oraz szumu.

Części rozkazu określające czas trwania i głośność są równe w tym przykładzie 0, ponieważ są one określone w obwiedni głośności.

Teraz, kiedy już potrafimy w pełni posługiwać się rozkazami SOUND, ENV i ENT, przyjrzyjmy się kilku innym rozkazom i funkcjom pokrewnym.

Przy opisie numeru kanału powiedzieliśmy, że dodanie 64 powoduje zatrzymanie dźwięku w kolejce dźwiękowej do chwili jego zwolnienia. Zwolnienie następuje rozkazem RELEASE. Po słowie RELEASE występuje liczba, której bity wskazują, który kanał ma być zwolniony. Powtarzamy jeszcze raz: niekoniecznie musisz w pełni rozumieć zapis binarny, ale musisz wiedzieć, że:

4 oznacza kanał C
2 oznacza kanał B
1 oznacza kanał A

..., a w celu zwolnienia dźwięków w kilku nakałach należy dodać numery tych kanałów. Tak więc, żeby zwolnić dźwięki zatrzymane we wszystkich kanałach należy zastosować:

```
RELEASE 7
```

... gdzie 7=1+2+4. Jeżeli w żadnym kanale nie ma zatrzymanych dźwięków, rozkaz RELEASE jest ignorowany. Wypróbuj poniższe:

```
SOUND 1+64,90
SOUND 2+64,140
SOUND 4+64,215
RELEASE 3:FOR t=1 TO 1000:NEXT:RELEASE 4
```

Dźwięki rozkazów SOUND nie są wytwarzane, dopóki rozkaz RELEASE nie zwolni kanałów A i B. Po opóźnieniu zwolniony zostaje kanał C.

Jest jeszcze jeden sposób, za pomocą którego można osiągnąć "rendezvous". Kiedy do kolejki dźwiękowej wysłany zostaje dźwięk z ustawionym bitem zatrzymania (dodane 64), wówczas nie tylko ten dźwięk zostaje zatrzymany, ale i wszystkie następne wysłane do tej samej kolejki. Jeżeli do takiej zatrzymanej kolejki zostanie wysłanych więcej niż 4 kolejne dźwięki, następuje zatrzymanie wykonywania programu do czasu zwolnienia kolejki, być może przez podprogram wywoływany przerwaniem po określonym czasie (instrukcje AFTER i EVERY). Jednakże nie jest to szczególnie dobra metoda użycia systemu dźwiękowego, ze względu na zawieszanie wykonywania programu, kiedy kolejka dźwiękowa zostaje przepełniona. Zatrzymywanie programu jest również widoczne kiedy długo trwające dźwięki są wysyłane w szybkiej sekwencji, na przykład:

```
10 FOR a=1 TO 8
20 SOUND 1,100  a,200
30 NEXT
40 PRINT "halo"
```

Wyraz "halo" nie pojawia się natychmiast, lecz dopiero po trzech pierwszych dźwiękach (kolejka nie jest wówczas przepełniona 8-3=5). Jest tak, ponieważ program nie może być kontynuowany, kiedy kolejka jest przepełniona.

BASIC jest wyposażony w mechanizm przerwania w przypadku zwolnienia się miejsca w kolejce, mechanizm podobny do działającego w rozkazach AFTER lub EVERY oraz w rozkazie ON BREAK GOSUB. Pozwala to wydzielić w programie podprogram wysyłania dźwięków wywoływany tylko wówczas, gdy jest miejsce w określonej kolejce. Zobacz jak działa:

```
10 a=0
20 ON SQ (1) GOSUB 1000
30 PRINT a;
40 GOTO 30
1000 a=a+10
1010 SOUND 1,a,200
1020 IF a  200 THEN ON SQ (1) GOSUB 1000
1030 RETURN
```

Zauważysz, że program ten działa bez przerwy. Rozkaz SOUND jest wysyłany tylko wtedy, kiedy w kolejce kanału A (numer 1) jest wolne miejsce. Warunek ten jest wprowadzony rozkazem ON SQ (1) GOSUB w linii 20. Rozkaz ten inicjalizuje mechanizm przerwania wywołującego podprogram dźwiękowy wtedy, kiedy w podanej kolejce jest miejsce. Mechanizm ten musi być reinicjalizowany i to zostało zrobione w linii 1020 w podprogramie wysyłania dźwięków. W przykładzie tym reinicjalizacja przerwań ma miejsce tylko wtedy, gdy a jest mniejsze od 200.

Korzystanie z przerwań pozwala na wykonywanie podkładu muzycznego i jednoczesne wykonywanie innych działań jak przesuwanie obiektów na ekranie, dodawanie punktów itd. Dźwięki są wysyłane do kolejki tylko wówczas, gdy w kolejce jest wolne miejsce. Dzięki temu wykonywanie programu nie jest wstrzymywane czekaniem na wolne miejsce w kolejce. Jeżeli wartości nut są czytane z instrukcji DATA, wówczas podprogram wysyłania dźwięków powinien przestać reinicjalizować się przy ostatniej wartości.

Liczba w nawiasach w ON SQ ( ) GOSUB powinna być równa 1, 2 lub 4 w zależności od tego, który kanał ma być sprawdzony.

W zestawie słów kluczowych języka BASIC jest również funkcja SQ ( ), która może być używana w programie do czytania bieżącego statusu dowolnego kanału dźwiękowego. Liczba w nawiasach i tutaj może być równa 1, 2 lub 4 i określa numer kanału. Funkcja ta daje wynik w postaci liczby, której poszczególne bity mają swoje znaczenie, tak więc i tym razem potrzebna jest znajomość liczb binarnych. Bity wyniku tej funkcji mają następujące znaczenie:

Bit	Dziesiętnie	Znaczenie
0,1,2	1,2,4	Liczba wolnych miejsc w kolejce
3	8	Pierwsza nuta w kolejce ma "rendezvous" z kanałem A
4	16	Pierwsza nuta w kolejce ma "rendezvous" z kanałem B
5	32	Pierwsza nuta w kolejce ma "rendezvous" z kanałem C
6	64	Górna nuta w kolejce ma ustawiony bit zatrzymania (kolejka jest zatrzymana)
7	128	Nuta jest właśnie grana

Zobacz jak działa prosty przykład:

```
10 SOUND 2,200
20 x=SQ (2)
30 PRINT BIN$(x)
```

Wydrukowana zostanie liczba binarna 10000100, w której bit 7 wskazuje, że kanał wytwarzał dźwięk w chwili sprawdzania. Ostatnie 3 bity 100, po przejściu na zapis dziesiętny dają liczbę 4, co wskazuje, że w kanale były 4 wolne miejsca. Funkcja ta może być użyta do sprawdzenia statusu kanału w danym miejscu programu - przeciwnie ON SQ( ) GOSUB sprawdza i reaguje na status kanału w miejscu trudnym z góry do określenia.

Jak dotąd, wszystkie przykłady zawierały jedną lub dwie nuty. Przetworzenie całej grupy nie związanych ze sobą nut, na przykład w utworze muzycznym, można osiągnąć przez zestawienie nut w instrukcji DATA, z której następnie są one czytane i wysyłane do kanału w rozkazie SOUND:

```
10 FOR oktawa = -1 TO 2
20 FOR x=1 TO 7: REM liczba nut w oktawie
30 READ nuta
40 SOUND 1, nuta/2 oktawa
50 NEXT
60 RESTORE
70 NEXT
80 DATA 426, 379, 358, 319, 284, 253, 239
```

Ostatni program przykładowy bazuje głównie na tej zasadzie. Melodia i rytm wykonywane są w kanałach A i B z wykorzystaniem "rendezvous" dla współbrzmienia. Przykład demonstruje jeden ze sposobów formowania danych zawierających informacje o wysokości i długości nuty, o wyborze oktawy i "rendezvous"

```
 10 REM linia 190 daje melodię w kluczu wiolinowym
 20 REM linia 200 daje melodię w kluczu basowym
 30 DIM scale%(12):FOR x%=1 TO 12:READ scale%(x%):NEXT
 40 ch1%=1:READ ch1$=ch2%=1:READ ch2$
 50 CLS
 60 Spd%=12
 70 scale$=" a-b b c+c d-e e f+f g+g"
 80 ENV 1,2,5,2,8,-1,10,10,0,15
 90 ENV 2,2,7,2,12,-1,10,10,0,15
100 ENT -1,1,1,1,2,-1,1,1,1,1
110 DEF FNM$(s$,s)=MID$(s$,s,1)
120 ch1%=1:GOSUB 200
130 ch2%=1:GOSUB 380
140 IF ch1%+ch2%  0 THEN 140
150 END
160 DATA  &777,&70c,&6a7,&647,&5ed,&598
170 DATA  &547,&4fc,&4b4,&470,&431,&3f4
180 DATA 4cr4f4f1f1g1A1-B2C2f4g2g1A1-B6A2Cr1f1g1f1g1a1-
    b1A1-b2C2g2A2g2f1g1a2g2f6e2c2e2c2g2e2c1-B1A2g2f4e4d
    8c4f3f1c2d4-b2fr2-B2A2g2f6e2gr4C4-B1a1f1-b1g2c2-b4a
    4g4fr6A2A2-B4-B2Ar2-B2A2g2f6e2g4C4-B1A1f1-B1g2C2-B1
    A4g8f
190 DATA r4f4f8f4e4c4fr8f4c2f2e4d2e2d8c8c6e2f4g1c8e4f2f
    1c4dr8g4cr4c4c6f2d4c4c8fr8-e4dr8g8c4c4c6f2d4c1c8f,
200 REM wysłanie dźwięku do kanału A
210 P1$=FNM$(ch1$,ch1%)
220 IF p1$<>"r" THEN r1%=0:GOTO 240
230 r1%=16:ch1%=1:p1$=FNM$(ch1$,ch1%)
240 IF p1$="." THEN ch1%=0:RETURN ELSE q1=VAL(p1$)
250 ch1%=ch1%+1
260 n1$=FNM$(ch1$,ch1%)
270 ch1%=ch1%+1
280 IF n1$="+" OR n1$="-" THEN 350
```

```
290 n1$=" "+n1$
300 nd1%=(1+INSTR(scale$,LOWER$(n1$)))/2
310 IF ASC(RIGHT$(n1$,1))  96 THEN 11%=8 ELSE o1%=16
320 SOUND 1+r1%,scale%(nd1%)/o1%,Spd%*l1%,0,1,1
330 ON SQ(1) GOSUB 200
340 RETURN
350 n1$=n1$+FNM$(ch1$,ch1%)
360 ch1%=ch1%+1
370 GOTO 300
380 REM wysłanie dźwięku do kanału B
390 p2$=FNM$(ch2$,ch2%)
400 IF p2$<>"r" THEN r2%=0:GOTO 420
410 r2%=8:ch2%=ch2%+1:p2S=FNm$(ch2$,ch2%)
420 IF p2$="." THEN ch2%=0:RETURN ELSE l2%=VAL(p2$)
430 ch2%=ch2%+1
440 n2$=FNM$(ch2$,ch2%)
450 ch2%=ch2%+1
460 IF n2$="+" OR n2$="-" THEN 530
470 n2$=" "+n2$
480 nd2%=(1+INSTR(scale$,LOWER$(n2$)))/2
490 IF ASC(RIGHT$(n2$,1))> 96 THEN o2%=4 ELSE o2%=8
500 SOUND 2+r2%,scale%(nd2%)/o2%,Spd%*l2%,0,2
510 ON SQ(2) GOSUB 380
520 RETURN
530 n2$=n2$+FNM$(ch2$,ch2%)
540 ch2%=ch2%+1
550 GOTO 480
run
```

Wypowiadając się graficznie ...

W punkcie tym opisano właściwości graficzne komputera CPC6128. Pierwszy przykład, rozbudowywany stopniowo, demonstruje kolejne cechy.

Na początek podzielmy ekran na okno tekstowe (linia 40) i okno graficzne (linia 30), ustanawiając po drodze MODE i pary przełączanych kolorów (linia 20):

```
10 REM definiowanie okien
20 MODE 1:INK 2,10,4:INK 3,4,10
30 ORIGIN 440,100,440,640,100,300
40 WINDOW 1,26,1,25
50 CLG 2
```

Po uruchomieniu tego programu zobaczysz kwadrat z przełączonymi kolorami w prawej części ekranu. Kwadrat ten został zapełniony atramentem numer 2 (przełączane magenta/cyjan) przez linię 50, a początek układu współrzędnych przesunięto do lewego, dolnego rogu kwadratu. Rozkaz MODE ustawił kursor graficzny w początku układu współrzędnych (X=0, Y=0), tak, że można narysować przekątną kwadratu za pomocą:

```
60 DRAW 200,200,3
```

Uruchom program, żeby zobaczyć efekt. Dodaj teraz:

```
80 MOVE 0,2:FILL 3
```

Linia 80 ustawia kursor graficzny wewnątrz jednej z dwu połówek kwadratu i wypełnia ją atramentem 3. Granicą wypełniania jest kraniec okna graficznego (w tym przypadku również kraniec kwadratu) i jakakolwiek linia narysowana aktualnym piórem graficznym (3) lub atramentem użytym do wypełniania (również 3). Uruchom program.

W celu udowodnienia tego, co powiedziano o krańcach wypełniania dopisz linię 70 jak niżej. Zauważ, że jedyną przyczyną, która ogranicza wypełnianie do połowy kwadratu jest to, że wypełnianie jest tym samym atramentem, którym była narysowana przekątna.

```
70 GRAPHICS PEN 1
```

Przerób linię 80, aby było FILL 1 i uruchom program, żeby sprawdzić ostatnią uwagę. Potem przywróć pierwotną postać (FILL 3).

Teraz dodaj linie 100 do 140, które rysują prostokąt:

```
100 MOVE 20,20
110 DRAW 180,20
120 DRAW 20,180
140 DRAW 20,20
```

Prostokąt jest narysowany atramentem 1 z powodu linii 70. Gdyby nie było linii 70, trzeba by dodać ",1" jako trzeci parametr w rozkazie MOVE w linii 100 lub jako trzeci parametr w rozkazie DRAW w linii 110 w celu poinformowania komputera o zmianie atramentu pióra graficznego.

Dołącz kropki ...

Linie nie muszą być ciągłe, mogą być kreskowane. Rozkaz MASK pozwala określać wymiary kropek. Wzór powtarzany jest co 8 punktów ekranu i każda następna linia kontynuuje sposób kropkowania od tego miejsca, do którego dotarła poprzednia linia. Wysłanie nowego rozkazu MASK (nawet z tym samym parametrem) powoduje odliczanie punktów (do 8) od początku.

Wzór rozłożenia kropek jest liczbą jednobajtową, w której bity równe 1 wskazują miejsce użycia atramentu pióra. W naszym przykładzie użyjemy stałej binarnej (oznaczonej X) wskazującej, że cztery środkowe punkty każdej ósemki mają być rysowane, a po dwa skrajne punkty każdej ósemki mają nie być rysowane. Dzięki temu otrzymamy linię przerywaną z czterema punktami narysowanymi i czterema nienarysowanymi. Dodaj:

```
90 MASK &X00111100
```

Poczekaj chwilę! Program ten nie daje nam ciągłej kontynuacji kropkowania w rogach figury, jak tego oczekiwaliśmy. Jest tak dlatego, że każdy punkt narożny jest w rzeczywistości rysowany dwukrotnie: raz na końcu jednej linii, drugi raz na początku drugiej linii. Niezręcznym sposobem ominięcia tego efektu jest wpisać:

```
115 MOVE 180,22
125 MOVE 178,180
135 MOVE 20,178
```

Tym niemniej, istnieje prostszy sposób, którym jest dodanie drugiego parametru ",0" w rozkazie MASK. Mówi to komputerowi, że pierwszy punkt każdej linii ma nie być rysowany. Przerób linię 90:

```
90 MASK &X00111100,0
```

... i usuń niepotrzebne linie wpisując:

115
125
135

Teraz uruchom program - kreskowany prostokąt jest znowu symetryczny. Zwróć uwagę na to, że jeżeli drugi parametr rozkazu MASK jest ",1", przywrócone zostaje rysowanie całych linii wraz z pierwszym punktem.

Teraz przyjrzyj się przerwom między kreskami linii. Coś tam jest w dolnym, prawym trójkącie, czego nie ma w górnym lewym trójkącie. Jest to papier graficzny, któremu rozkaz CLG2 przypisał atrament numer 2 (linia 50). Papier jest niewidoczny w górnym, lewym trójkącie, ponieważ jest koloru tła. Zmień linię 50:

50 CLG 2:GRAPHICS PAPER 0

... i powtórnie uruchom program. Papier jest teraz widoczny dokoła całego prostokąta.

Papier graficzny może być niewidzialny lub, jak mówimy, "przezroczysty". Oznacza to, że linia przerywana poprowadzona przez istniejący rysunek zachowa tło między kreskami. Rysowanie graficzne może być "przezroczyste" wtedy, kiedy zostanie do rozkazu GRAPHICS PEN dodany parametr ",1" (powrót do "nieprzezroczystego" trybu przez parametr ",0". Zmień linię 70:

70 GRAPHICS PEN 1,1

... i zaobserwuj wynik.

Z pozycją kursora graficznego związane mogą być nie tylko linie (i punkty) lecz również znaki tekstowe. Ma to wielkie znaczenie ze względu na ustawianie kursora graficznego z dużą dokładnością (do jednego punktu, a nie ośmiu jak w przypadku kursora tekstowego), jak również ze względu na to, że znaki w tym przypadku mogą być rysowane z dodaniem efektów wynikających z rodzajów rysowania graficznego (patrz dalej).

Jeżeli chcesz wpisać znak na pozycji kursora graficznego, powinieneś umieścić kursor w lewym, górnym rogu przewidywanego znaku, następnie wydać rozkaz TAG (lub TAG#1 itp. dla innych strumieni tekstowych), a potem rozkaz PRINT. Kursor graficzny

jest automatycznie przesuwany o 8 punktów po każdym wpisaniu znaku. Dopisz:

```
160 MOVE 64,108
170 TAG
180 PRINT "SALLY"
190 TAGOFF
```

(Komunikaty BASICA kierowane są na ekran tekstowy bez względu na stan przełączenia TAG/TAGOFF, ale kasowanie TAG przez TAGOFF natychmiast po wpisaniu tekstu na ekran graficzny jest dobrym przyzwyczajeniem).

Ale skąd się wzięły strzałki za imieniem? No cóż, są to znaki powrotu do początku wiersza (CR) CHR$(13) i przesunięcia do nowej linii (LF) CHR$ (10). Program rysowania graficznego tłumaczy wszystkie znaki kontrolne ASCII (pierwsze 32 znaki) na ich wersję drukowaną (tak jakby były one poprzedzone znakiem CHR$ (1); przy skierowaniu ich do okna tekstowego). Przyczyną tego jest to, że większość znaków kontrolnych ma właściwe znaczenie tylko w oknie tekstowym. Z tego samego powodu na przykład, przekroczenie prawego krańca okna graficznego nie powoduje przeniesienia do nowej linii.

Symbole CHR$ (13) i CHR$ (10) mogą być usunięte normalną techniką kończenia rozkazu PRINT średnikiem:

```
180 PRINT "SALLY";
```

Tekst pisany na ekranie graficznym przy użyciu TAG, jest pisany tym samym piórem graficznym, co rysowane linie. W naszym przypadku imię SALLY jest pisane piórem określonym przez GRAPHICS PEN 1 i jest "przezroczyste". Rozkaz:

```
150 GRAPHICS PEN 1,0
```

... przywróci "nieprzezroczystość" papieru, podczas gdy:

```
150 GRAPHICS PEN 0,1
```

... spowoduje użycie atramentu 0 i "przezroczystego" papieru. Usuń teraz linię 150 i ponownie uruchom program. Atrament numer 1 + "przezroczystość" (wcześniej ustawione w linii 70) zostaną znowu przypisane pióru graficznemu.

Przezroczyste znaki

Istnieje również możliwość wpisywania w sposób "przezroczysty", znaków w okno tekstowe. Należy w tym celu użyć odpowiednich znaków kontrolnych. Dopisz:

```
200 PRINT #2,CHR$(22);CHR$(1)
210 LOCATE #2,32,14:PRINT #2,"*#####*"
220 LOCATE #2,32,14:PRINT #2,"------"
230 PRINT #2,CHR$(22);CHR$(0)
```

Linia 200 ustawia tryb "przezroczystości" w strumieniu #2. Zwróć uwagę, że podkreślenie przecina gwiazdki. Pokazuje to, że można tworzyć nowe znaki z kombinacji znaków podstawowych, nawet przy użyciu różnych kolorów. Linia 230 wyłącza tryb "przezroczystości".

Tryby rysowania

Przy rysowaniu można korzystać z różnych trybów rysowania, które określają sposób interakcji atramentu używanego z atramentem już występującym na ekranie. Ostateczny numer atramentu dla każdego punktu jest obliczany jako kombinacja logiczna starego atramentu danego punktu z atramentem graficznym (pióra lub papieru), którym punkt ma być narysowany. Możliwe są następujące kombinacje logiczne: XOR (eXclusive OR), AND i OR. Tryb rysowania może być określony czwartym parametrem rozkazów DRAW, DRAWR, PLOT, PLOTR, MOVE, MOVER lub przez wydrukowanie rozkazem PRINT znaków: CHR$ (23); CHR$ (<tryb rysowania>). W każdym przypadku 1 ustala tryb XOR, 2 ustala tryb AND, 3 ustala tryb OR. Wartość 0 przywraca tryb "siłowy", w którym atrament stary nie jest brany pod uwagę.

Następny przykład przedstawia użycie trybu XOR. Tryb ten jest często używany w tak zwanej "Grafice żółwia", ponieważ ma właściwość przywracania pierwotnego rysunku po dwukrotnym narysowaniu tego samego wzoru. Dlatego podprogram rysowania kwadratu jest wykonywany dwukrotnie (w liniach 110 i 130), podobnie jak drukowanie według kursora graficznego (poprzedzone rozkazem TAG) (w liniach 170 i 190). Rozkazy FRAME wprowadzają

opóźnienie wystarczające dla zaobserwowania efektu. Zwróć uwagę na brak pierwszego parametru w rozkazach linii 90. Jest to całkowicie poprawne w tych rozkazach i oznacza, że pierwszy parametr ma pozostać niezmieniony.

Trzeci parametr (,1) rozkazu MOVE w linii 220 ustala GRAPHICS PAN 1, unieważniając ustalenie GRAPHICS PEN 3 w linii 60. Tryb XOR jest ustanowiony czwartym parametrem rozkazu DRAWR a linii 230. Znowu zwróć uwagę na brak parametru.

Dodatkowo efekt rysowania pierwszego punktu linii można zobaczyć przez usunięcie rozkazu MASK z linii 90. Wierzchołki kwadratu znikają, ponieważ są rysowane dwukrotnie (na końcu linii i na początku następnej linii) i w związku z tym są eliminowane przez działanie XOR.

```
10 REM tryb rysowania XOR
20 MODE 1:INK 2,10:INK 3,4
30 ORIGIN 440,100,440,640,100,300
40 WINDOW 1,26,1,25
50 CLG 2:GRAPHICS PAPER 0
60 DRAW 200,200,3
70 MOVE 2,0:FILL 3
80 ORIGIN 440,0,440,640,0,400
90 GRAPHICS PEN ,1:MASK ,0
100 FOR y=60 TO 318 STEP 2
110 GOSUB 220
120 FRAME:FRAME
130 GOSUB 220
140 NEXT
150 TAG
160 FOR y=60 TO 318 STEP 2
170 MOVE 96,y:PRINT CHR$ (224);
180 FRAME:FRAME
190 MOVE 96,y:PRINT CHR$ (224);
200 NEXT
210 END
220 MOVE 90,y,1
230 DRAWR 20,0,,1
240 DRAWR 0,20
```

```
250 DRAWR -20,0
260 DRAWR 0,-20
270 RETURN
run
```

Animacja

Efekt animacji można uzyskać również przez zmianę kolorów przypisanych do atramentów. Chociaż zawartość pamięci ekranu pozostaje wówczas niezmieniona, na ekranie widać ruch. Przykład takiego efektu podano w programie "Welcome" na 4. stronie pakietu dysków systemowych (wpisz RUN "disc), żeby zobaczyć ten pokaz). Proste przełączanie kolorów tego przykładu, nie jest jednak wystarczające w przypadku, gdy animowane wzory zachodzą na siebie. W następnym przykładzie tryb rysowania OR używamy jest przy pisaniu na ekranie liczb od 1 do 4. (Kształt określany jest przez badanie znaku wydrukowanego w lewym dolnym rogu ekranu i reprodukowanie go w powiększeniu). Liczby wpisywane są po kolei przy użyciu atramentów 1, 2, 4 i 8 z trybem OR włączonym w tym przypadku przez wysłanie znaków kontrolnych w linii 50.

Linie 160 i następne wykonują rotację kolorów zgodnie z określonym wyrażeniem matematycznym, co powoduje, że w danej chwili na ekranie widoczna jest tylko jedna powiększona liczba. Przypisywanie kolorów do atramentów odbywa się przez przeglądanie kolejno wszystkich atramentów i sprawdzanie, czy numer danego atramentu zawiera składową binarną, której poszukujemy. Na przykład, liczba 3 została narysowana atramentem numer 4 i dlatego, w celu pokazania liczby 3 musimy przypisać kolor widzialny wszystkim atramentom, których numery zawierają 4 w zapisie binarnym. Są to:
4(0100),5(0101),6(0110),7(0111),12(1100),13(1101),14(1110), 15(1111)

W praktycznych zastosowaniach numery atramentów, które należy zmienić na każdym stopniu animacji mogą być obliczone, a linie 180 do 200 wymienione na szybciej działający fragment programu.

```
10 REM animacja zatrzaskowa
20 ON BREAK GOSUB 220
30 FOR i=1 TO 15:INK 1,26:NEXT
40 m(1)=1:m(2)=2:m(3)=4:m(4)=8
50 MODE 0:PRINT CHR$(23);CHR$(3);:TAG
60 FOR P=1 TO 4
70 GRAPHICS PEN m(p),1
80 LOCATE #1,1,25:PRINT#1,CHR$(48+p);
90 FOR x=0 TO 7
100 FOR y=0 TO 14 STEP 2
110 IF TEST (x*4,y)=0 THEN 140
120 MOVE (x+6)*32,(y+6)*16:PRINT CHR$(143);
130 MOVE (x+6)*32,(y+7)*16:PRINT CHR$(143);
140 NEXT y,x,p
150 LOCATE #1,1,25:PRINT#1," ";
160 FOR p=1 TO 4
170 FOR i=1 TO 25:FRAME:NEXT
180 FOR i=0 TO 15
190 IF (i AND m(p))=0 THEN INK 1,0 ELSE INK 1,26
200 NEXT i,p
210 GOTO 160
220 INK 1,26
run
```

Ożywiają również kolory wielopoziomowe ...

W poprzednim przykładzie widzieliśmy, jak, po narysowaniu elementów graficznych atramentami 1, 2, 4 i 8, efekt animacji został osiągnięty poprzez zmianę kolorów atramentów. Jeżeli używane są atramenty z tymi samymi kolorami, lecz przypisanie kolorów do atramentów następuje w inny sposób, efekt ostateczny jest zupełnie inny. Efekt ten znany jest jako efekt "kolorów wielopoziomowych" i został zademonstrowany w poniższym przykładzie.

```
10 REM góry
20 DEFINT a-z
30 INK 0,1:INK 1,26
40 INK 2,6:INK 3,6
50 FOR i=4 TO 7:INK 1,9:NEXT
60 FOR i=8 TO 15:INK 1,20:NEXT
70 MODE 0:DEG:ORIGIN 0,150:CLG:MOVE 0,150
80 FOR x=16 TO 640 STEP 16
90 DRAW x,COS(x)*150+RND*100,4
100 NEXT
110 MOVE 0,0:FILL 4
120 cx=175:GOSUB 320
130 cx=525:GOSUB 320
140 SYMBOL 252,0,0,&7C,&1F,&30,&7F,&FF
150 SYMBOL 253,0,6,&E,&F2,2,&F2,&FE
160 SYMBOL 254,0,&60,&70,&7F,&7F,&7F,&7F
170 SYMBOL 255,0,0,0,&F8,&FC,&FE,&FF
180 pr$=CHR$(254)+CHR$(255)
190 pl$=CHR$(252)+CHR$(253)
200 TAG:t!=TIME
210 FOR x=-32 TO 640 STEP 4
220 x2=((608-x)*2)MOD 640:hl=RND*10:hr=50*SIN(x)
230 GRAPHICS PEN 8,1:MOVE x,100+hr,,3:PRINT pr$;
240 GRAPHICS PEN 2,1:MOVE x2,115+hl,,3:PRINT pl$;
250 IF (TEST(x2-2,115+hl-12) AND 8)=8 THEN 380
260 IF TIME-t!<30 THEN 260
270 FRAME:t!=TIME
280 GRAPHICS PEN 7,1:MOVE x,100+hr,,2:PRINT pr$;
290 GRAPHICS PEN 13,1:MOVE x2,115+hl,,2:PRINT pl$;
300 NEXT
310 GOTO 210
320 MOVE cx,100
330 FOR x=0 TO 360 STEP 10
340 DRAW cx+SIN(x)*50+10*RND,100+COS(x)*25+10*RND,1
350 NEXT
360 DRAW cx,100:MOVE cx,90:FILL 1
370 RETURN
380 ENT -1,1,1,1
```

```
390 SOUND 1,25,400,15,,1,15
400 FOR y=100+hr TO -132 STEP -2
410 GRAPHICS PEN 7,1:MOVE x,y,,2:PRINT pr$;
420 GRAPHICS PEN 8,1:MOVE x,y-2,,3:PRINT pr$;
430 NEXT
440 GOTO 70
run
```

W celu wyjaśnienia działania tego programu, raz jeszcze musimy wrócić do numeru binarnego atramentu. Zaczynając od najwyższego numeru atramentu (15), wszystkie atramenty, które mają bit o wadze 8 równy 1 (od 15 do 8) mają przypisany kolor cyjan. Następnie atramenty, których bit o wadze 4 jest równy 1 (od 7 do 4) mają przypisany kolor zielony. Atramenty 2 i 3, których bit o wadze 2 jest równy 1 mają przypisany kolor czerwony, wreszcie atrament numer 1 jest jaskrawo-biały, a atrament 0 pozostaje błękitny.

Elementy graficzne są rysowane na ekranie w trybie OR - patrz linie 230 i 240. Kolor widziany na ekranie w każdym punkcie jest określony przez najbardziej znaczący bit wyniku funkcji OR, równy 1. Dlatego obraz na "bardziej znaczącym" poziomie zawsze zaciemnia obraz znajdujący się na "mniej znaczącym" poziomie, ale z zachowaniem tego obrazu, który może być znowu widzialny po usunięciu obrazu "bardziej znaczącego". Sposób usunięcia polega na narysowaniu powtórnym w trybie AND atramentami 7, 11, 13 lub 14 usuwającymi atramenty pierwotnie użyte (8,4,2 i 1 odpowiednio) - zobacz linie 280 i 290.

Grafika przy użyciu dodatkowej pamięci

Na zakończenie rozdziału podajemy obszerny program "Projektant ekranu graficznego" ("Graphics Screen Designer"), który używa drugiego bloku 64 K pamięci RAM komputera 6128.

Dwa rozkazy RSX |SCRFENCOPY i |SCREFNSWAP, zastosowane w tym programie są otrzymywane z programu użytkowego "Bank Manager" znajdującego się na 1.stronie pakietu dysków.systemo-

wych. Rozkazy te ułatwiają kopiowanie i wymianę obrazów pamięci ekranu pomiędzy odpowiednimi blokami pamięci i między dwoma bankami pamięci 64k.

Ze względu na to, należy uruchomić program "Bank Manager" przed uruchomieniem programu "Screen Designer". Aby to uczynić, włóż w napęd dysków strony 1. pakietu dysków systemowych i wpisz:

```
run "bankman"
```

Teraz możesz uruchomić program "Screen Designer".

```
10 'SCREEN DESIGNER by DAVID RADISIC
20 ' copyright (c) AMSOFT 1985
30 '
40 'Remember to RUN "BANKMAN" before running program!
50 '************************************************
60 '
70 ON ERROR GOTO 2740
80 DEFINT a-x
90 MODE 1:ch=127:cmnd=1:pn(0)=0:pn(1)=26:pn(2)=15:pn(3)
   =6:pn(4)=0:pn=1:norx=1:menu=1:zzz=HIMEM
100 DIM command$(22)
110 norx$(0)="Normal":norx$(1)="XOR    ":norx$(2)="Trans
    p":norx$(3)="XOR    "
120 RESTORE:READ cmnds$(1),cmnds$(2):cmnd$=CHR$(16)+CHR
    $(&7F)+cmnds$(1)+cmnds$(2)
130 READ cmno:FOR i=1 TO cmno:READ command$(i):NEXT
140 READ st$:IF st$<>"**" THEN cmnd$(cmnd)=st$:cmnd=cmn
    d+1:GOTO 140
150 WINDOW #0,1,40,1,3:PAPER #0,0:PEN #0,1:CLS #0
160 WINDOW #1,1,40,4,4:PAPER #1,3:PEN #1,1:CLS #1
170 ORIGIN 0,0,0,640,0,334
180 x=320:y=200:MOVE x,y
190 BORDER pn(4):FOR i=0 TO 3:INK i,pn(i):NEXT
200 MASK 255,0:PAPER 0:PEN 1:PAPER #1,3:PEN #1,1:GRAPHI
    CS PEN pn,norx
210 IF flag<>5 THEN 280
```

```
220 IF pn<2 THEN pnt$=CHR$(240):px=(pn+1)*13 ELSE IF pn
    <4 THEN pnt$=CHR$(241):px=(pn-1)*13 ELSE pnt$=CHR$(
    243):px=37
230 LOCATE px,2:PRINT pnt$;
240 LOCATE 1,1:PRINT USING"   PEN 0 : ##   PEN 1 : ##";
    pn(0);pn(1);
250 LOCATE 29,2:PRINT USING"Border : ##";pn(4)
260 LOCATE 1,3:PRINT USING"   PEN 2 : ##   PEN 3 : ##";
    pn(2);pn(3);
270 LOCATE px,2:PRINT " ";
280 LOCATE #1,1,1:PRINT#1,USING"X :####  Y :####     ";
    x;y;:PRINT #1,"Plot mode : ";norx$(norx+(undraw*2))
    ;" ";
290 IF flag=0 THEN GOSUB 2260
300 '
310 GOSUB 970
320 '
330 IF flag>0 THEN 390
340 IF i$="" THEN 390
350 cmnd=INSTR(cmnd$,i$):IF cmnd=0 THEN 390
360 IF cmnd=1 THEN CLG:x=320:y=200:GOTO 390
370 IF cmnd=2 THEN RUN 70
380 ON cmnd-2 GOSUB 1240,1410,1520,1640,1840,1860,1950,
    2020,2090,2120,2170,2200,2660,2660,2660,2660,2390,2
    330,2200
390 IF tx=0 AND ty=0 THEN 200
400 IF flag>0 THEN 440
410 GOSUB 630
420 GOSUB 680:FRAME:GOSUB 680
430 GOTO 200
440 MOVE tempx,tempy,pn,1
450 ON flag GOSUB 470,490,550,640
460 GOTO 200
470 PLOT x,y:GOSUB 630:PLOT x,y
480 RETURN
490 DRAW tempx+x,tempy:DRAW tempx+x,tempy+y
500 DRAW tempx,tempy+y:DRAW tempx,tempy
510 GOSUB 630
520 DRAW tempx+x,tempy:DRAW tempx+x,tempy+y
530 DRAW tempx,tempy+y:DRAW tempx,tempy
540 RETURN
550 MOVE tempx,tempy:DRAWR x,y
560 IF triside=0 THEN 580
570 DRAW tempxx,tempyy:DRAW tempx,tempy
```

```
580 GOSUB 630
590 MOVE tempx,tempy:DRAW tempx+x,tempy+y
600 IF triside=0 THEN RETURN
610 DRAW tempxx,tempyy:DRAW tempx,tempy
620 RETURN
630 x=x+tx:y=y+ty:RETURN
640 MOVE tempx,tempy:DRAW x,y
650 GOSUB 630
660 MOVE tempx,tempy:DRAW x,y
670 RETURN
680 ' draw and undraw cursor
690 IF flag=5 THEN RETURN
700 MASK 255,1
710 IF flag>1 THEN xx=tempx+x:yy=tempy+y ELSE xx=x:yy=y
720 IF flag=4 THEN xx=x:yy=y
730 IF flag=1 THEN xx=x:yy=y
740 IF undraw=1 THEN 820
750 GOSUB 790
760 MASK 255,0
770 IF i$=" " THEN GOSUB 2150:i$=""
780 RETURN
790 MOVE xx-4,yy,pn,1:DRAW xx+4,yy
800 MOVE xx,yy-4:DRAW xx,yy+4
810 MOVE xx,yy,,xorn:RETURN
820 nx=1:GOSUB 1220
830 FRAME:GOSUB 1220
840 IF i$=" " THEN nx=norx:GRAPHICS PEN pn,1:GOSUB 1220
850 i$=""
860 IF flag<>6 THEN 760
870 IF moved=0 AND j$<>"" AND (j$<CHR$(240) OR j$>CHR$(
    247)) THEN ch=ASC(j$):moved=1
880 IF moved=0 THEN RETURN
890 LOCATE 5,2
900 FOR i=ch-5 TO ch+5
910 PEN ABS(i<>ch)+1
920 ch$=CHR$(1)+CHR$(ABS(i+256)MOD 256)
930 IF ch=i THEN PRINT" "ch$" "; ELSE PRINT ch$;
940 NEXT
950 PEN 1:PRINT"   = "ch"  ";
960 GOTO 760
970 ty=0:tx=0:GOSUB 680:FRAME:GOSUB 680
980 IF INKEY(0)<>-1 OR INKEY(72)<>-1 THEN ty=16
990 IF INKEY(2)<>-1 OR INKEY(73)<>-1 THEN ty=-16
1000 IF INKEY(8)<>-1 OR INKEY(74)<>-1 THEN tx=-16
```

```
1010 IF INKEY(1)<>-1 OR INKEY(75)<>-1 THEN tx=16
1020 IF INKEY(21)<>-1 OR INKEY(76)<>-1 THEN tx=tx/8:ty=
     ty/8
1030 IF tx=0 AND ty=0 THEN moved=0 ELSE moved=1
1040 j$=INKEY$:i$=UPPER$(j$)
1050 IF (i$=" " OR i$=CHR$(13)) AND flag>0 THEN 1090
1060 IF flag=5 THEN 1120
1070 IF flag=6 THEN 1170
1080 RETURN
1090 ON flag GOSUB 1240,1410,1640,1860,1950,2020
1100 i$=""
1110 RETURN
1120 IF moved=0 THEN RETURN
1130 IF tx>2 THEN pn=(pn+1) MOD 5 ELSE IF tx<-2 THEN pn
     =ABS((pn<1))*5-1+pn
1140 IF ty>2 THEN pn(pn)=(pn(pn)+1) MOD 27 ELSE IF ty<-
     2 THEN pn(pn)=ABS((pn(pn)<1))*27-1+pn(pn)
1150 GRAPHICS PEN pn:PEN #1,pn
1160 tx=0:ty=0:BORDER pn(pn):RETURN
1170 IF tx<0 THEN ch=ABS(ch+255) MOD 256
1180 IF ty<0 THEN ch=ABS(ch+246) MOD 256
1190 IF tx>0 THEN ch=(ch+1) MOD 256
1200 IF ty>0 THEN ch=(ch+10) MOD 256
1210 tx=0:ty=0:RETURN
1220 TAG:MOVE xx-8,yy+6,pn,nx:PRINT CHR$(ch);:TAGOFF
1230 RETURN
1240 ' C
1250 IF flag=1 THEN 1290
1260 ro=1:GOSUB 2240
1270 tempx=x:tempy=y:flag=1
1280 RETURN
1290 IF tempx=x AND tempy=y THEN 1390
1300 PLOT x,y,,1
1310 tix=MAX(x,tempx)-MIN(tempx,x):tiy=MAX(y,tempy)-MIN
     (tempy,y)
1320 ti=SQR((tix↑2)+(tiy↑2))
1330 ORIGIN tempx,tempy
1340 PLOT 0,0,pn,0:MOVE 0,-ti
1350 FOR z=0 TO PI*2+0.01 STEP PI/(ti/2)
1360 DRAW SIN(z+PI)*ti,COS(z+PI)*ti,pn,norx
1370 NEXT z
1380 ORIGIN 0,0
1390 x=tempx:y=tempy:tempx=0:tempy=0:flag=0
1400 RETURN
```

```
1410 ' B
1420 IF flag=2 THEN 1470
1430 ro=2:GOSUB 2240
1440 tempx=x:tempy=y:flag=2
1450 x=0:y=0
1460 RETURN
1470 IF norx=1 THEN 1500
1480 MOVE tempx,tempy:DRAW tempx+x,tempy,,norx
1490 DRAW tempx+x,tempy+y:DRAW tempx,tempy+y:DRAW tempx
     ,tempy
1500 x=tempx:y=tempy:flag=0
1510 RETURN
1520 ' F
1530 ro=3:GOSUB 2240
1540 GOSUB 1620:IF i$=" " THEN 1600
1550 edgecol=VAL(i$)
1560 ro=4:GOSUB 2240
1570 GOSUB 1620:IF i$=" " THEN 1600
1580 filler=VAL(i$)
1590 MOVE x,y,edgecol:FILL filler
1600 flag=0:i$=""
1610 RETURN
1620 i$=INKEY$:IF (i$<"0" OR i$>"3") AND i$<>" " THEN 1
     620
1630 RETURN
1640 ' T
1650 IF flag=3 THEN 1700
1660 flag=3:ro=5:GOSUB 2240
1670 tempx=x:tempy=y
1680 x=0:y=0
1690 RETURN
1700 IF triside<>0 THEN 1770
1710 ro=6:GOSUB 2240
1720 MOVE 0,0,pn,1:GOSUB 590
1730 tempxx=tempx+x:tempyy=tempy+y:x=x/2:y=20
1740 triside=1
1750 GOSUB 550:GOSUB 590
1760 RETURN
1770 IF norx=1 THEN 1800
1780 MOVE tempxx,tempyy,,norx:DRAW tempx,tempy
1790 DRAW tempx+x,tempy+y:DRAW tempxx,tempyy
1800 tempxx=0:tempyy=0
1810 x=tempx:y=tempy:triside=0
1820 tempx=0:tempy=0:flag=0
```

```
1830 RETURN
1840 ' @
1850 norx=1:undraw=undraw XOR 1:RETURN
1860 ' L
1870 IF flag=4 THEN 1910
1880 ro=7:GOSUB 2240
1890 tempx=x:tempy=y:flag=4
1900 RETURN
1910 IF norx=1 THEN 1930
1920 MOVE tempx,tempy,,norx:DRAW x,y
1930 x=tempx:y=tempy:flag=0
1940 RETURN
1950 ' I
1960 IF flag=5 THEN flag=0:CLS:INK 3,tmpcol:INK pn,col:
     GOTO 1990
1970 CLS:flag=5:BORDER pn(pn)
1980 RETURN
1990 FOR i=0 TO 3:INK i,pn(i):NEXT:BORDER pn(4)
2000 IF pn=4 THEN pn=1
2010 CLS:RETURN
2020 ' A
2030 IF flag=6 THEN 2070
2040 tempx=0:tempy=0:CLS
2050 undraw=1:flag=6:norx=1:moved=1
2060 RETURN
2070 flag=0
2080 RETURN
2090 ' N
2100 norx=0
2110 RETURN
2120 ' E
2130 GRAPHICS PEN pn,0:TAG:MOVE xx-8,yy+6,,0:PRINT " ";
     :TAGOFF
2140 RETURN
2150 '<SPACE>
2160 PLOT x,y,pn,norx:RETURN
2170 ' X
2180 norx=1
2190 RETURN
2200 ' M
2210 menu=menu MOD 2+1
2220 GOSUB 2260:RETURN
2230 i$=UPPER$(INKEY$):IF i$="" OR INSTR(ser$,i$)=0 THE
     N 2230 ELSE RETURN
```

```
2240 CLS:undraw=0:PRINT cmnd$(ro);:LOCATE 1,3:PRINT"<SP
     ACE> ";:IF ro=3 OR ro=4 THEN PRINT"To exit"
2250 RETURN
2260 CLS:flag=-1
2270 FOR i=1 TO LEN(cmnds$(menu))
2280 ps=i+ABS(menu=2)*LEN(cmnds$(1))
2290 PEN 1:PRINT"<"MID$(cmnds$(menu),i,1)">"MID$(comman
     d$(ps),2,4)" ";
2300 NEXT
2310 PRINT"<CLR>   <DEL>   <SPACE>";
2320 RETURN
2330 ' S
2340 GOSUB 2460:IF filename$="" THEN 2370
2350 GOSUB 2550
2360 SAVE filename$,b,&C000,&4000
2370 GOSUB 2260
2380 RETURN
2390 ' R
2400 GOSUB 2460:IF filename$="" THEN 2440
2410 GOSUB 2730
2420 LOAD filename$,&C000
2430 GOSUB 2570
2440 GOSUB 2260
2450 RETURN
2460 CLS:LOCATE 10,3:PRINT"<RETURN> to Abort!";
2470 LOCATE 1,1:PRINT"Enter Filename :";
2480 INPUT "",filename$:IF filename$="" THEN RETURN
2490 n=INSTR(filename$,"."):IF n=0 THEN 2520
2500 IF n=1 THEN 2460
2510 filename$=LEFT$(filename$,n-1)
2520 filename$=LEFT$(filename$,8)+".scn"
2530 CLS
2540 RETURN
2550 FOR i=0 TO 4:POKE &C000+i,pn(i):NEXT
2560 RETURN
2570 FOR i=0 TO 4:pn(i)=PEEK(&C000+i) MOD 27:NEXT
2580 cn=0:FOR i=0 TO 2:IF pn(i)=pn(i+1) THEN cn=cn+1
2590 NEXT:IF cn=3 THEN 2630
2600 FOR i=0 TO 3:INK i,pn(i):NEXT
2610 BORDER pn(4):pn=1:GRAPHICS PEN pn
2620 RETURN
2630 pn(0)=0:pn(1)=26:pn(2)=15:pn(3)=6:pn(4)=0
2640 GOTO 2600
2650 ' 1, 2, 3, & 4
```

```
2660 CLS:PRINT"Do you wish to <S>tore":PRINT TAB(16)"<R
     >etrieve":PRINT TAB(13)"or <E>xchange  the screen
     ?"
2670 ser$="SRE"+CHR$(13):GOSUB 2230:IF i$=CHR$(13) THEN
      2260
2680 bnk2=(cmnd-13):bnk1=1
2690 IF i$="S" THEN CLS:GOSUB 2550:|SCREENCOPY,bnk2,bnk
     1
2700 IF i$="R" THEN GOSUB 2730:|SCREENCOPY,bnk1,bnk2:GO
     SUB 2570
2710 IF i$="E" THEN CLS:GOSUB 2730:GOSUB 2550:|SCREENSW
     AP,bnk2,bnk1:GOSUB 2570
2720 GOSUB 2260:RETURN
2730 FOR i=0 TO 3:INK i,0:NEXT:BORDER 0:RETURN
2740 CLS:GOSUB 2600:RESUME 2260
2750 DATA "CBFT@LIANEXM","1234RSM"
2760 DATA 19,Circle,"Box  ","Fill ",Triangle,Alternate,
     "Line ","Inks ",ASCII,Normal,Erase,"Xor  ","Menu "
     ,"1st  ","2nd  ","3rd  ","4th  ",Restore,"Save ","
     Menu "
2770 DATA Circle,Box,Edge colour,Filler colour,Triangle
      1,Triangle 2,Line,**
```

Co teraz?

Po uruchomieniu programów "Bank Manager" i "Screen Designer" na ekranie wyświetlane jest menu dostępnych opcji, jak również migający kursor graficzny w środku ekranu.

W tym miejscu należy nacisnąć odpowiedni klawisz /np. C dla okręgu /Circle// w celu wybrania żądanej opcji.

Dla przykładu wpisz:

C

... potem naciśnij klawisz "kursor w górę" aż do przesunięcia kursora około 1 cal od środka ekranu.

W końcu naciśnij klawisz SPACJA, co spowoduje narysowanie okręgu i powrót do menu.

Wpisując M /Menu/ uzyskasz przełączenie do dodatkowego menu, w którym możesz dokonać zapisu na dysk lub odtworzenia z dysku pamięci ekranu oraz wykonywać manipulacje zawartości ekranów o numerach 1,2,3 i 4 /umieszczonych w drugich 64 K pamięci komputera/.

W celu użycia tych funkcji ekranu, wpisz "numer pamięci" /1,2,3 lub 4/, po czym wyświetlone zostanie kolejne menu.

Stąd można wybrać:

S	zapis /stare/ ekranu na dysku
R	odtworzenie /Retrieve/ ekranu
E	wymiana /Exchange/ ekranów
RETURN	wycofanie się z operacji

Jeżeli, na przykład, chcesz zapisać, aktualny ekran w bloku pamięci o numerze 2, wpisz 2, a następnie S.

Po powrocie do dodatkowego menu (po operacji R, S lub E) ponowne wpisanie M spowoduje przełączenie do menu podstawowego.

Screen Designer daje pewną liczbę ułatwień graficznych takich jak: rysowanie prostokątów, okręgów, trójkątów, linii, punktów, wypełnianie i rysowanie znaków.

Po utworzeniu obrazu na ekranie, może być on zapisany na dysku, w celu późniejszego wykorzystania, za pomocą opcji S z dodatkowego menu i wprowadzonej z powrotem, w każdej chwili, za pomocą opcji R.

No cóż, w ten sposób zakończyliśmy ostatni rozdział podręcznika. Przez stosowanie, analizowanie i eksperymentowanie z programami zamieszczonymi w niniejszym przewodniku możesz uzyskać dobre zrozumienie języka AMSTRAD BASIC i samego komputera 6128.

Dalsze szczegóły ...

Wiele publikacji AMSOFT i innych dostarczy ci dalszych szczegółów o 6128 oraz o języku BASIC, oprogramowaniu firmowym, systemie CP/M i języku Dr.LOGO.

Mamy nadzieję, że podręcznik ten był dla ciebie ciekawy i dostarczył ci potrzebnych informacji. Dziękujemy za zakup CPC6128.

DODATEK

SŁOWNICZEK TERMINÓW

Wyjaśniamy użytkownikom 6128 niektóre powszechnie używane terminy z dziedziny techniki komputerowej. Ze względu na możliwość natrafienia na te terminy w brzmieniu angielskim, tłumacze pozostawili angielskie nazwy poszczególnych haseł.

Accumulator - Akumulator
Element pamiętający (rejestr) wewnątrz struktury mikroprocesora, główny rejestr roboczy mikroprocesora w którym są przejściowo przechowywane dane w czasie ich przetwarzania. Intensywnie używany w programach w języku maszynowym - programujący w języku BASIC mogą nie wiedzieć o istnieniu takiego rejestru.

Acoustic coupler - akustyczne urządzenie wejścia/wyjścia
Także określane jako modem akustyczny. Elektroniczna przystawka łącząca komputer ze słuchawką telefoniczną i umożliwiająca komputerowi przesyłanie informacji za pomocą zwykłej akustycznej sieci telefonicznej. W ten sposób komputer może komunikować się z systemami publicznej informacji jak np. system PRESTEL lub z innymi użytkownikami komputerów domowych w celu wymiany oprogramowania, pobrania danych lub informacji itp.

Address - Adres
Liczba w instrukcji identyfikująca lokację komórki pamięci komputera. Za pomocą adresu wyodrębnić można określoną komórkę pamięci tak, że jej zawartość staje się możliwa do odczytania a w przypadku pamięci RAM możliwy jest i odczyt i zapis do komórki.

Adventure game - gra przygodowa
Dla niektórych przedmiot kultu dla innych nudziarstwo. Gra komputerowa z dużą ilością tekstu w której gracz jest zapraszany do udziału w serii pseudolosowych zdarzeń przy próbie np. znalezienia drogi wyjściowej z labiryntu.

Algorithm - algorytm
Pompatyczna nazwa skomplikowanej formuły lub sumy. Sekwencja dokładnie zdefiniowanych czynności logicznych i matematycznych prowadzących do wykonania określonego zadania w technice komputerowej.

Alphanumeric - alfanumeryczny
Atrybut wyrażający różnicę między literami i cyframi a innymi znakami graficznymi.

ALU
Arthimetic Logic Unit, jednostka arytmetyczno-logiczna. Część mikroprocesora, w której wykonywane są operacje arytmetyczne i logiczne. Bloku tego dotyczą bezpośrednio tylko rozkazy programu w języku maszynowym.

Ambiguous File Name - niejednoznaczna nazwa zbioru
Nazwa zbioru zawierająca jeden lub więcej znaków zastępczych. Nazwa taka obejmuje więcej niż jedną określoną nazwę zbioru i jest używana do jednoczesnego wywołania jednego lub kilku zbiorów.

AMSDOS
AMStrad Disc Operating System. Dyskowy system operacyjny firmy AMSTRAD. Program umożliwiający pracę Locomotive BASIC ze zbiorami dyskowymi.

AMSOFT
Specjalistyczny oddział firmy AMSTRAD opracowujący oprogramowanie, urządzenia peryferyjne i publikacje, przede wszystkim w celu rozszerzenia możliwości użytkowych 6128.

Analoque - A/C, Analogowy
Analogowy - to sygnał, którego wielkość może zmieniać się w sposób ciągły w odróżnieniu od sygnału cyfrowego, którego wartość zmienia się stopniowo, określonymi przyrostami. Komputer jest urządzeniem cyfrowym - w świecie rzeczywistym występują z reguły zjawiska analogowe, aby zatem komputer mógł przetwarzać sygnały rzeczywiste, konieczne jest wstępne przetworzenie sygnału analogowego na cyfrowy (A/C) - Analog to Digital (A/D).

Animation - animacja
Najbardziej znaną formą animacji jest film rysunkowy - animacja komputerowa wykorzystuje koncepcję ruchomej grafiki do symulowania "żywego" ruchu.

Applications program - program aplikacyjny
Program realizujący określone zadanie a nie "narzędzie" programowe o zastosowaniu uniwersalnym jak np. asembler czy edytor.

Arcade Game - Gra rozrywkowa
Typ komputerowej gry wizyjnej z akcją pełną ruchu w której np. zwycięża się atakujących kosmitów lub nienasycone monstra gonią dookoła labiryntu i pożerają nieostrożnych. Postacie sterowane przez grającego muszą unikać różnego rodzaju okropnych "śmierci". Ogólnie wyrabiające refleks ale o małym znaczeniu dydaktycznym dla uczących się posługiwania komputerem.

Architecture - architektura (systemu komputerowego)
Podstawowa koncepcyjnie struktura systemu mikrokomputerowego, określająca ukształtowanie systemu i powiązania między CPU i urządzeniami peryferyjnymi poprzez linie przesyłania danych.

Argument
Zmienna występująca w operacji arytmetycznej lub logicznej, np. w wyrażeniu x+y=z argumentami są x, y i z.

Array - tablica
2-wymiarowa macierz (siatka) w której dane są przechowywane w pozycjach, określonych przez adresowanie we współrzędnych "poziomych" i "pionowych".

Artificial intelligence AI - sztuczna inteligencja
Ukształtowanie programu, które umożliwia programowi uczenie się na podstawie poprzednich doświadczeń.

ASCII
American Standard Code for Information Interchange, amerykański kod standardowy do wymiany informacji. Ogólnie przyjęty sposób (kod) zapisu cyfr, liter i innych symboli wprowadzanych z klawiatury komputera lub za pomocą różnych rozkazów.

Assembler - asembler
Praktyczna metoda programowania w kodzie maszynowym, w której kody maszynowe instrukcji wywoływane są przez nazwy mnemoniczne (tj. takie, których litery sugerują funkcję wykonywaną przez odpowiednią instrukcję w kodzie maszynowym).

Backup - żelazna rezerwa
Kopia (duplikat) używanej informacji przechowywana jako zabezpieczenie na wypadek straty lub przypadkowego zniszczenia oryginału. Szczególnie zalecane w przypadku cennych zbiorów dyskowych.

Bar code - kod paskowy
Możliwy do przeczytania przez komputer kod drukowany, czytany za pomocą urządzeń optycznych (np. lasera małej mocy). Umieszczany na opakowaniach większości zachodnich towarów powszechnego użytku.

Base - podstawa liczenia
Podstawa każdego systemu przedstawiania liczb. System dwójkowy ma jako podstawę liczenia cyfrę 2; system dziesiętny ma za podstawę liczbę 10, a system szesnastkowy to system o podstawie 16.

BASIC

Beginners' All-purpose Symbolic Instruction Code", język instrukcji symbolicznych ogólnego zastosowania dla początkujących". Interpretacyjny język programowania używany w prawie wszystkich komputerach domowych. BASIC został specjalnie zaprojektowany tak, aby był łatwy do nauczenia i prosty w użyciu; umożliwia "sklejanie" programów i wypróbowywanie ich działania w każdym momencie ich opracowywania w przeciwieństwie do języków typu kompilacyjnego, w których uruchomić można tylko kompletny program bez właściwego sprawdzenia poszczególnych jego fragmentów

Baud - bod
Bit na sekundę. Jednostka miary szybkości transmisji danych cyfrowych w systemach transmisji szeregowej.

BCD
Binary Coded Decimal. System kodowania liczb dziesiętnych w zapisie dwójkowym (binarnym) w którym każda cyfra dziesiętna jest przedstawiana za pomocą grupy czterech cyfr binarnych.

BDOS
Basic Disc Operating System, podstawowy dyskowy system operacyjny. Część systemu operacyjnego CP/M umożliwiająca programom użytkownika korzystanie z funkcji CP/M.

Benchmark - "urząd sędziowski"
Standardowe zadanie, które może być rozwiązywane przez różne komputery w celu porównania ich szybkości, sprawności i dokładności, np. obliczenie pierwiastka kwadratowego z 99.999 podniesionego do kwadratu.

Binary - system dwójkowy (binarny)
System liczenia o podstawie 2, w którym wszystkie liczby są przedstawiane za pomocą dwu cyfr dwójkowych 0 i 1 (patrz część 1 rozdziału zatytułowanego "W wolnej chwili...")

Binary number - liczba dwójkowa (binarna)
Liczba przedstawiona w zapisie dwójkowym. Oznaczana w programach 6128 przedrostkiem &X, np. &X0101 = (dziesiętnie) 5.

BIOS
Basic Input/Output System, podstawowy system wejścia/wyjścia. Moduł programu CP/M, uzależniony od sprzętu, pisany specjalnie dla określonego typu komputera. Wszystkie oporacje wejścia/wyjścia związane z ekranem, klawiaturą, dyskiem itd. są przeprowadzane za pośrednictwem modułu BIOS.

Bit
Skrót od BInary digiT, cyfra binarna. Pojedyncza cyfra (pozycja) liczby binarnej. Cyfra binarna jest jedną z dwu cyfr, reprezentowanych przez 0 i 1, jakie są używane w dwójkowym (binarnym) systemie liczenia.

Bit Significant - znacząca bitowo
Gdy informacja zawarta w liczbie jest wykorzystywana przez rozważenie stanu każdego z ośmiu bitów, tworzących kompletny bajt. Całkowita dziesiętna wartość liczby nie ma żadnego znaczenia.

Boolan algebra - algebra Boola
Wyrażenie zależności logicznych w których mogą być tylko dwie odpowiedzi: prawda lub fałsz, zwykle oznaczane symbolami 1 lub 0.

Boot
Proces ładowania systemu operacyjnego do pamięci. Gdy CP/M jest wprowadzany z języka BASIC mały program ładujący (boot program) jest wprowadzany do pamięci z dysku i program ten wprowadza następnie resztę systemu operacyjnego do pamięci.

Booting lub Bootstraping - wciąganie, wprowadzanie
Programy i systemy operacyjne nie ładują się same, lecz są wciągane (bootstrapped) przez krótki program umieszczony (zwykle) w ROM, który inicjalizuje proces ładowania do określonego obszaru pamięci.

Buffer - bufor
Obszar pamięci zarezerwowany do przejściowego przechowywania czyli buforowania informacji w czasie przenoszenia informacji.

Bug - pluskwa , błąd w programie
Wywołują kłopoty od drobnych "niespodziewanych właściwości" wynikających z pewnych niejasnych aspektów działania programu (np. gdy naciśniesz jednocześnie cztery klawisze, ekran zmienia kolor) do następstw które kompletnie i nieodwracalnie niszczą program i wszystkie zgromadzone w pamięci dane.

Build-in commands - polecenia (rozkazy) wbudowane (rezydentne)
Rozkazy które są integralną częścią systemu operacyjnego. Rozkazy takie są zawsze szybsze niż rozkazy nierezydentne (przejściowe) ponieważ wykonujące je programy nie muszą być wciągane z dysku.

Bus - magistrala przesyłowa
Zespół połączeń albo wewnątrz komputera albo łączących urządzenia zewnętrzne, przenoszący informacje o stanie CPU, RAM i innych urządzeń sprzętowych. Magistrala 6128 jest doprowadzona do złącza rozszerzającego, oznaczonego symbolem EXPANSION, znajdującego się na płycie drukowanej komputera z tyłu komputera.

Byte - bajt
Grupa 8 bitów, odpowiadających najmniejszej komórce pamięci jaką 8-bitowa jednostka centralna (CPU) może wczytać lub zapisać.

CAD
Computer Aided Design, Projektowanie wspomagane komputerem. Zwykle wykorzystanie mocy obliczeniowych i graficznych komputera do wytworzenia elektronicznej deski kreślarskiej, chociaż każde obliczenia wykonywane za pomocą komputera na użytek prac projektowych prowadzone mogą być pod szyldem CAD.

CAE
Computer Aided Education, Nauczanie wspomagane komputerem. Kolejny wymysł frazeologii komputerowej. Użycie komputera jako pomocy w nauczaniu. Dwiema odmianami CAE są: CAI (Computer Aided Instruction) i CAL (Computer Aided Learning) co również oznacza nauczanie wspomagane komputerem.

Cartridge - ładunek
Odpowiednio obudowany scalony układ pamięciowy zawierający oprogramowanie, wtykany bezpośrednio w gniazdo specjalnie przygotowane w tym celu w komputerze. Oprogramowanie zamieszczone w takim ładunku można łatwo i szybko wykorzystać, lecz koszty są znacznie wyższe niż w przypadku oprogramowania dostarczanego na dysku.

Cassette - kaseta
Oprócz oczywistego określenia kasety magnetofonowej pospolite określenie różnorodnych "pakietów" - włączając w to "pakiety" z oprogramowaniem w ROM itp.

CCP
Console Command Processor. Jest to moduł programu CP/M który interpretuje i wykonuje polecenia wprowadzane przez użytkownika za pomocą klawiatury ("z konsoli"). Zwykle wprowadzane są rozkazy (commands), które CCP wyszukuje i wykonuje.

Character - znak
Jakikolwiek symbol który może być zapisany i wyświetlony przez komputer, włącznie z literami, cyframi i symbolami graficznymi.

Character set - zestaw znaków
Wszystkie litery, cyfry i symbole jakie mogą być używane przez komputer lub drukarkę. Występowanie znaku w komputerze nie oznacza, że występuje on także w każdej drukarce.

Character string - ciąg znaków
Kawałek danych zmiennej , zawierający sekwencję znaków, który może być przechowywany i manipulowany jako niepodzielna jednostka np. słowo lub zbiór słów.

Chip
Błędne lecz popularne żargonowe określenie elektronicznego monolitycznego układu scalonego. "Chip" oznacza dokładnie mały kawałek specjalnie przetworzonego materiału krzemowego, na którym wytworzony jest układ scalony (chip = odłamek, skrawek).

Clock - zegar
Układ odmierzający i taktujący czas w komputerze używany do synchronizacji i wyznaczenia czasu wszystkich operacji komputera. Zegar czasu rzeczywistego to z kolei taki zegar który odmierza minuty, godziny, dni itp.

Code - kod
Wyrócz zwykłego znaczenia, często używany przez programistów jako skrótowe określenie "Kodu maszynowego".

Cold start - "zimny start"
Proces ładowania i inicjalizacji systemu operacyjnego. "Zimny start" systemu CP/M następuje po rozkazie ICPM.

Command - rozkaz
Instrukcja programu

Compiler - kompilator
Złożony program który zamienia kompletny program napisany w języku interpretacyjnym wysokiego poziomu jak np. BASIC w bezpośredni kod instrukcji mikroprocesora co umożliwia wykonanie programu ze znacznie większą prędkością.

Computer generations - generacje komputerów
Technologiczne "kamienie milowe" wyznaczyły kilka odrębnych kroków w rozwoju technologii komputerów i grupowanie według wynikających z tego warstw określa "generacje" komputerów.

Computer literacy - znajomość komputerowego abecadła
Pompatyczne wyrażenie oznaczające podstawową znajomość techniki komputerowej.

Console mode - bezpośredni tryb pracy "z konsoli"
Bezpośredni tryb pracy systemu CP/M; na ekranie wyświetlany jest znak A> i system oczekuje na wprowadzenie polecenia rezydentnego CP/M lub nazwy programu użytkowego.

Corruption - zepsucie
Zniszczenie lub zmiana zawartości zbioru dyskowego lub pamięci w niepożądany i niewyjaśniony sposób.

CP/M
Control Program for Microcomputers, Program sterujący dla mikrokomputerów. Dyskowy system operacyjny opracowany przez Digital Research, zapewniający standardową obsługę systemową oprogramowania napisanego dla szerokiej grupy systemów mikroprocesorowych (z procesorami 8080 lub Z80).

CPU
Central Processing Unit, Centralna jednostka przetwarzająca. Zasadnicza część składowa, "serce" każdego systemu komputerowego która interpretuje instrukcje wprowadzane do komputera i nakazuje postępowanie zgodne z tymi instrukcjami. W mikrokomputerach CPU stanowi najczęściej sam mikroprocesor.

Cursor - kursor
Ruchomy znacznik wskazujący miejsce w którym pojawi się na ekranie następny znak.

Cursor control keys - klawisze sterujące kursorem
Klawisze które powodują przesuwanie kursora po ekranie i są często używane do kierowania akcją w grach rozrywkowych; klawisze te oznaczone są strzałkami.

Daisy-wheel printer - drukarka rozetkowa
Drukarka która może wytwarzać dokumenty o wysokiej jakości, podobne do pisanych na dobrej maszynie do pisania. Drukowane znaki powstają przez uderzanie czcionek kompletnych liter barwionych tuszem lub poprzez taśmę maszynową.

Database - baza danych
Tablica różnego typu danych o różnych formatach atresowania przez komputer.

Data capture - porcja (zdobycz, łup) danych
Termin określający zbiór danych z jakiegoś źródła zewnętrznego dołączany w pewien sposób do centralnego komputera.

Debugging - "odpluskwianie"
Proces usuwania błędów z programu.

Decimal notation - zapis dziesiętny
Także określany jako system dziesiętny tj. system liczenia o podstawie 10, w którym używane są cyfry od 0 do 9 dla określenia liczby jednostek, dziesiątek, setek, tysięcy itd.

Defaut - zaniedbanie
Wartość przyjmowana domyślnie w przypadku braku jakiegokolwiek określenia jej przez użytkownika. Na przykład, po inicjalizacji CP/M dysk A jest przyjmowany domyślnie.

Delimiter
Patrz Separator

Diagnostic - diagnostyczny
Komunikat samoczynnie wysyłany przez komputer w celu zasygnalizowania i identyfikacji błędu w programie.

Digital - cyfrowy
Opisuje przebieg zmieniającej się wielkości za pomocą dyskretnych kroków a nie jako proces ciągły. Przeciwieństwo do analogowego.

Digitiser
Przetwornik analogowo-cyfrowy, urządzenie umożliwiające wprowadzanie do komputera informacji analogowej. Pojęcie kojarzone czasem także z cyfrowymi stołami graficznymi (koordynatografami).

Directory - skorowidz dysku
Wyodrębniona sekcja dysku zawierająca opisy wszystkich zbiorów na dysku. Lista zawartości dysku.

Disc (lub disk) - dysk
Płaski, cienki krążek wykonany z tworzywa sztucznego, pokryty po jednej lub obu stronach warstwą tlenku o właściwościach magnetycznych stosowany jako środek do przechowywania informacji. Dysk jest umieszczony w osłaniającej kopercie z okienkiem, umożliwiającym dostęp do dysku głowicy czytająco-zapisującej. W dysku 3-calowym okienko zasłaniane jest przez metalową przesłonę, zasuwaną automatycznie po wyjęciu dysku z napędu dyskowego.

Disc drive - napęd dyskowy
Mechanizm używany do napędzania dysku i odczytywania lub zapisywania danych na dysku.

Documentation - dokumentacja
Podręczniki dostarczane z komputerami lub programami, wyjaśniające ich działanie.

DOS
Disc Operating System, dyskowy system operacyjny. Program kontrolujący wszystkie operacje napędu dyskowego.

Dot matrix - matryca punktów
Prostokątna siatka punktów w której mogą być wyświetlane znaki; wymagany kształt znaku uzyskuje się przez wybór odpowiednich punktów siatki.

Double sided - dwustronny
Dysk który może przechowywać informacje po obu stronach. Dwustronny napęd dyskowy ma dostęp do obydwu stron dysku bez konieczności przekręcania dysku.

Download - "ładowanie w dół"
Przesyłanie informacji z jednego komputera do innego - komputer przyjmujący dane jest zwykle określany jako "ładowany w dół"; komputer przekazujący dane jest natomiast "ładującym z góry" (uploading)

Dr. LOGO
Wersja LOGO, języka programowania z grafiką żółwia, opracowana przez Digital Research

Dumb terminal - "niemy terminal"
Terminal komputerowy działający wyłącznie jako urządzenie wprowadzania i wyprowadzania danych bez żadnego przetwarzania przechodzących przez niego informacji. Taki "nieinteligentny" terminal nie zawiera nawet układów elektronicznych sterujących wyświetlaniem a informacje do wyświetlania na ekranie są do niego wprowadzane w postaci sygnału wizyjnego.

Edit - edytować, redagować
Wprowadzać poprawki lub zmiany w danych, programie lub tekście.

Editor - edytor, program redagujący
Program, zwykle umieszczony w pamięci ROM komputera, za pomocą którego prowadzić można proces redagowania (edycji).

EPROM
Erasable Programmable Read Only Memory, pamięć stała, "tylko do czytania", którą można programować i kasować. Podobna do pamięci PROM, lecz dane zaprogramowane w układzie pamięci mogą być kasowane za pomocą promieniowania ultrafioletowego po czym pamięć może być ponownie zaprogramowana. Pamięci typu EEPROM mogą być kasowane elektronicznie.

Expression - wyrażenie
Prosta lub skomplikowana formuła używana w programie do przeprowadzenia obliczeń; wyrażenie zwykle określa rodzaj danych, jakimi może operować. W Dr. Logo wyrażenie zawiera nazwę procedury z niezbędnymi dla niej danymi.

Fifth generation computers - komputery piątej generacji
Głównie duże komputery, które mają się pojawić niebawem, o zdolnościach do samoprogramowania w oparciu o rozwiniętą sztuczną inteligencję.

File - zbiór
Zbiór danych, zwykle przechowywany na dysku lub taśmie magnetofonowej.

File name - nazwa zbioru
Nazwa zbioru. W Dr. Logo nazwa zbioru może zawierać do 8 liter lub cyfr. W CP/M i AMSDOS do nazwy dodaje się po kropce . dalsze trzy znaki określające typ zbioru.

Firmware - oprogramowanie firmowe
Oprogramowanie zawarte w pamięciach ROM, zapewniające połączenie między czystym oprogramowaniem uniwersalnym a urządzeniami sprzętowymi komputera.

Fixed-point number - liczba stałoprzecinkowa
Liczba przedstawiana, przekształcana i przechowywana z kropką dziesiętną w określonej, stałej pozycji.

Floating-point number - liczba zmiennoprzecinkowa
Liczba rzeczywista, przekształcana i przechowywana z kropką dziesiętną ustawianą w żądanej pozycji. Metoda szczególnie użyteczna przy operowaniu dużymi liczbami.

Floppy disc- dysk elastyczny
Patrz Disc - dysk

Flowchart - diagram (schemat) postępowania
Graficzne przedstawienie kroków programowych i procesów logicznych, odpowiadających sekwencji zdarzeń w czasie wykonywania programu.

Forth
Język programowania o dużej szybkości wykonywania programu; o szybkości i złożoności lokującej go między językiem wysokiego poziomu a programem w kodzie maszynowym. Język nie dla początkujących.

Function key - klawisz funkcyjny
Klawisz klawiatury któremu może być przypisana specjalna funkcja; funkcję taką może wykonywać w dodatku lub zamiast głównego zadania, przypisanego do tego klawisza.

Gate - bramka
Bramka logiczna; bramki takie przenoszą sygnały przy spełnieniu określonych warunków. Istnieją bramki różnych typów, np. OR, AND, XOR itd.

Graphics - grafika
Sposób wyświetlania na ekranie komputera różniący się od wyświetlania "znaków". Grafika umożliwia rysowanie i wykreślanie linii, kół i różnych innych wzorów.

Graphics character - znak graficzny
Postać, kształt lub wzór specjalnie zaprojektowany do użytku przy tworzeniu obrazów.

Graphic cursor - kursor graficzny
Podobny do kursora tekstowego, lecz adresuje ekran graficzny. W 6128 niewidoczny, występuje tylko jako pojęcie, lecz mimo to stanowi niezbędne ułatwienie przy lokalizowaniu rysowanych linii graficznych. Nie należy używać tego kursora do lokalizowania znaków graficznych, które wchodzą w skład "zestawu znaków" i są lokalizowane przez kursor tekstowy.

Graphics mode - tryb graficzny
Starsze mikrokomputery wymagały specjalnego przestawienia na pracę w trybie znakowym lub graficznym. Nowoczesne komputery osobiste umożliwiają jednoczesne mieszanie tekstu i grafiki.

Graphics tablet - Koordynatograf
Urządzenie określające współrzędne punktów zadanego obrazka lub wykresu dla przetwarzania ich przez komputer. Rodzaj przetwornika analogowo-cyfrowego (A/D).

Handshaking - transmisja z potwierdzeniem
Sekwencja sygnałów elektronicznych które inicjują potwierdzają i synchronizują wymianę danych między komputerem a urządzeniem peryferyjnym lub między dwoma komputerami.

Hard Copy - wydruk
Wydruk na papierze programu lub innego tekstu albo zawartości ekranu graficznego. Przejściowe przedstawienie tego samego na ekranie jest określane mianem soft copy .

Hardware - sprzęt
Elektroniczne i mechaniczne urządzenia systemu komputerowego - wszystko, co nie jest oprogramowaniem (soft ware) (ang. ware - towar; hard - twardy, soft - miękki).

Hexadecimal (lub HEX) - szesnastkowy
System liczb o podstawie 16. Przy programowaniu 6128 liczby szesnastkowe oznacza się przedrostkiem &lub &H np. &FF =(dziesiętnie) 255. (Patrz część 1 rozdziału zatytułowanego "W wolnej chwili").

Hex file - zbiór typu hex
Zbiór z zapisanym za pomocą znaków ASCII programem w kodzie maszynowym.

High-level - language - język wysokiego poziomu
Język jak BASIC, który jest napisany w"niemal dosłownej" formie, gdzie najwięcej pracy wymaga interpretowanie. Wolniejszy niż programy w kodzie maszynowym lecz łatwiejszy do zrozumienia.

IEEE-488
Jeden ze standardowych systemów sprzęgających do łączenia różnych urządzeń z mikrokomputerem. Podobny do łącza równoległego "Centronics" choć nie całkiem z nim kompatybilny.

Information technology - technika informatyczna
Wszystko co dotyczy użycia elektroniki w przetwarzaniu i przesyłaniu danych: przetwarzanie tekstów, wymiana danych, PRESTEL itp.

Initialise - inicjalizacja
Włączenie systemu lub zadeklarowanie specyficznych wartości zmiennych przed rozpoczęciem wykonywania głównej części programu - np. zwymiarowanie tablic, zadeklarowanie zmiennych jako typu liczby całkowitej itp.

Input - wejście
Wszystko co wchodzi do pamięci komputera z klawiatury, jednostki dyskowej, łącza transmisji szeregowej lub innych źródeł wejściowych.

Instruction - instrukcja
Polecenie lub rozkaz dla komputera nakazujący wykonanie określonej operacji. Zbiór lub sekwencja instrukcji tworzą program.

Instruction set - zbiór instrukcji
Podstawowe logiczne i matematyczne operacje wykonywane przez mikroprocesor. Wszystkie instrukcje wysokiego poziomu (włącznie z mnemonikami asemblera) muszą dać się rozłożyć na instrukcje rozpoznawane przez jednostkę centralną (CPU) komputera. Pojedyncza instrukcja wysokiego poziomu może wywoływać dużą liczbę elementarnych instrukcji ze zbioru instrukcji mikroprocesora.

Integer - część całkowita
Część całkowita liczby

Integer number - liczba całkowita
Liczba bez części ułamkowej w przeciwieństwie do liczby rzeczywistej, która zawiera część całkowitą i część ułamkową.

Integrated circuit - układ scalony
Zbiór elementów elektronicznych tworzących funkcjonalny układ elektroniczny, zminiaturyzowany i wykonany we wspólnym cyklu produkcyjnym na kawałku płytki krzemowej (patrz także Chip).

Intelligent terminal - terminal inteligentny
Terminal, który oprócz wykonywania zleceń komputera z zakresu wprowadzania i wyprowadzania danych przeprowadza także lokalne przetwarzanie danych w czasie, gdy nie jest bezpośrednio wykorzystywany przez komputer.

Interactive - interakcyjny
Zwykle odnosi się do programów, których działanie polega na zachęcaniu użytkownika do wprowadzania różnych danych - od zaproszenia do sterowania statkiem kosmicznym w grze rozrywkowej do odpowiedzi na pytania w programach edukacyjnych. Akcja użytkownika oddziaływuje w "czasie rzeczywistym" na działanie programu.

Interface - interfejs, łącze
Sposób wprowadzania i wyprowadzania informacji do lub z komputera, zarówno w sensie sygnałów elektrycznych jak i bezpośredniego komunikowania się z użytkownikiem. Interfejs 6128 stanowi klawiatura (wejście) i ekran (wyjście) - jak również środki umożliwiające dołączenie urządzeń peryferyjnych do różnych gniazd komputera.

Interpreter
Dalsze rozszerzenie analogii między zbiorem instrukcji komputera a językiem. Moduł oprogramowania systemowego który interpretuje język wysokiego poziomu tak aby uzyskać interpretację (tłumaczenie) na poziomie zrozumiałym przez jednostkę centralną (CPU), np. zamienia instrukcje języka BASIC, wprowadzone za pomocą klawiatury na instrukcje w języku maszynowym komputera.

I/O - we/wy
Input/Output - wejście/wyjście

Iteration - iteracja
Jeden z elementów obliczania. Komputer wykonuje wszystkie zadania przez rozłożenie ich na proste procedury, jakie mają być wykonywane przez jednostkę centralną (CPU). Aby wykonać zadanie wykonuje wielokrotnie wiele prostych procedur aż do spełnienia żądanych warunków.

Joystick - drążek sterowy
Urządzenie wejściowe które zastępuje klawisze sterowania kursorem i ułatwia prowadzenie gier komputerowych.

K
Skrótowa forma przedrostka jednostki miary 1000 razy większej od jednostki podstawowej, "kilo" - w technice komputerowej szeroko używana w odniesieniu do "kilobajta", w skrócie także KB: kilobajt jest dokładnie równy 1024 bajtom co w stosowanym dwójkowym systemie liczenia odpowiada 2 do potęgi 10.

Keyboard - klawiatura
Matryca klawiszy alfanumerycznych przeznaczona do wprowadzania poleceń tekstowych i innych informacji do komputera.

Keyword - słowo kluczowe
Słowo, którego użycie w programie komputerowym lub języku jest zarezerwowane dla określonych funkcji lub rozkazów.

Least significant bit - bit najmniej znaczący
W liczbie dwójkowej (binarnej) najmniej znaczący bit (LSB) to skrajny bit na prawym końcu wyrażenia.

Light Pen - pióro świetlne
Inna alternatywna metoda wprowadzania danych przy użyciu "pióra" lub "laseczki".

Line number - numer linii
BASIC i niektóre inne języki używają programów których linie ułożone są w kolejności określonej przez ich numery porządkowe.

Lisp
Skrót pochodzący od nazwy LISt Processor language. Inny komputerowy język wysokiego poziomu.

Logic- logika
Układy elektroniczne realizujące elementarne operacje i funkcje logiczne z których składa się każda operacja komputerowa.

Logical device - urządzenie logiczne
Reprezentacja urządzenia która może być różna od jego formy fizycznej. Na przykład w systemie CP/M urządzeniu logicznemu LST może być przydzielone łącze Centronics lub monitor ekranowy.

LOGO
Nazwa języka programowania wywodząca się od greckiego słowa "logos", oznaczającego "słowo". Logo jest przeznaczone do nauczania podstawowych zasad programowania komputerowego.

Loop - pętla
Część programu wykonywana przez komputer wielokrotnie aż do spełnienia określonego warunku.

Low-level language - język niskiego poziomu
Taki jak "język asemblera". Język programowania w którym każda instrukcja odpowiada instrukcji w kodzie maszynowym komputera.

LSI
Large Scale Integration, wielka skala integracji. Rozwinięte układy scalone, umożliwiające upakowanie większej liczby funkcji na mniejszym kawałku krzemu.

Machine Code - kod maszynowy
Język programowania bezpośrednio rozumiany przez mikroprocesor, gdyż każda jego instrukcja przedstawiana jest za pomocą liczby dwójkowej (binarnej).

Machine readable - możliwe do czytania przez maszynę
Rodzaj danych lub innych informacji które mogą być bezpośrednio wprowadzone do komputera bez dodatkowej pracy z klawiaturą lub t.p.

Man - machine interface - interfejs człowiek - komputer
Urządzenia umożliwiające współdziałanie operatora z komputerem: klawiatura, ekran, dźwięk itd.

Matrix - matryca; macierz
Układ punktów które formują komórkę znaku na ekranie lub w głowicy drukującej drukarki znakowo-mozaikowej. Także pojęcie używane w matematyce i technice komputerowej na określenie tablicy.

Memory - pamięć
"Przestrzeń parkingowa" komputera do przechowywania informacji i danych, zorganizowana w logiczne komórki, indywidualnie osiągane przez komputer. Pamięć jest określana mianem RAM (Random Access Memory, pamięć o dostępie swobodnym), jeżeli informacja może być zarówno zapisywana do pamięci jak i z pamięci odczytywana lub ROM (Read-Only Memory, pamięć stałe, tylko do odczytu) jeżeli informacja może być tylko odczytywana z pamięci a nie może być do niej zapisana. Dyski i taśmy magnetyczne to przykłady "pamięci masowej", chociaż pojęcie pamięci dotyczy głównie pamięci, bezpośrednio adresowanej przez CPU.

Memory map - mapa pamięci
Plan pamięci, przedstawiający przypisanie obszarom pamięci o sprecyzowanych adresach określonych funkcji jak np. pamięć ekranu, pamięć dyskowego systemu operacyjnego itp.

Menu
"Lista dań", lista różnych opcji realizowanych przez program przedstawionych do wyboru przez użytkownika.

Microprocessor - mikroprocesor
Układ scalony, stanowiący serce mikrokomputera; układ ten wykonuje instrukcje wysyłane przez interpreter języka BASIC lub inny program i steruje zgodnie z tymi instrukcjami urządzeniami wejścia/wyjścia komputera.

Modem
MOdulator-DEModulator łączący kanał wejścia/wyjście komputera z linią telefoniczną lub innym środkiem szeregowej transmisji danych łącznie ze światłowodami. (Patrz także "Acoustic coupler - akustyczne urządzenie wejścia/wyjścia).

Monitor
Część ekranowa terminala komputerowego; także nazwa określająca program w języku maszynowym umożliwiający realizację podstawowych funkcji maszynowych komputera.

Mouse - myszka
Manipulator kulowy oparty kulką o blat stołu. Przesuwany ręką po blacie stołu przesuwa kursor po ekranie. Zaprojektowany przede wszystkim po to, aby ominąć strach przed klawiaturą i ułatwić posługiwanie się programami przez użytkownika.

MSB
Most Significant Bit, najbardziej znaczący bit liczby dwójkowej (binarnej) tj. bit na lewym krańcu wyrażenia dwójkowego.

Network - sieć
Kilka komputerów połączonych razem przewodami lub poprzez modemy w celu wymiany danych i informacji.

Nibble (kąsek)
Pół bajtu, wyrażenie czterobitowe. Każda cyfra liczby szesnastkowej (heksadecymalnej) np. &F6 zajmuje pół bajtu.

Node - węzeł
Jednostka przechowywania w przestrzeni roboczej LOGO. Najczęściej jeden węzeł zajmuje 4 bajty obszaru pamięci.

Noise - szum
Urządzenia dźwiękowe 6128 mają możliwość dołączania różnych ilości czumu w celu wytwarzania takich efektów jak np. eksplozje

Numeric keypad - blok klawiszy cyfrowych
Obszar klawiatury ze zgrupowanymi dodatkowymi klawiszami cyfrowymi dla ułatwienia wprowadzania danych liczbowych. W 6128 klawisze takie mogą być dodatkowo programowane przez użytkownika jako klawisze funkcyjne.

OCR
Optical Character Recognition, optyczne rozpoznawanie znaków. Oznacza czytanie za pomocą czytnika optycznego znaków drukowanych i pisanych i przetwarzanie ich do postaci możliwej do wprowadzenia do komputera.

Octal - ósemkowy
System liczenia o podstawie 8, w którym każda cyfra (0-7) jest zapisywana za pomocą trzech bitów

Off line
Urządzenie peryferyjne komputera - zwykle terminal lub drukarka - które nie jest czynnie dołączone lub jest niedostępne dla głównej jednostki przetwarzającej.

On line
Przeciwieństwo Off line

Operating system - system operacyjny
Program, rezydujący w pamięci komputera poniżej lub powyżej obszaru pamięci, pozostawionego do dyspozycji użytkownika. Program taki organizuje pracę komputera, wyznaczając m.in. pierwszeństwo i synchronizując wykonanie poszczególnych operacji.

Operator
Znak w wyrażeniu arytmetycznym, określający rodzaj operacji przeprowadzanej na liczbach (argumentach) wyrażenie, np.
+ - * / itd.

Output - wyjście
Wszystko co wychodzi z komputera jako wynik wykonywanych operacji.

Overwrite - pisanie na czymś już zapisanym
Wykasowanie obszaru pamięci w wyniku zapisania w miejsce dotychczasowej treści nowych danych.

Paddle - wiosło
Inna nazwa joystick'a

Page zero - strona zerowa
Nazwa używana w systemie CP/M na określenie obszaru pamięci od adresu &0000 do adresu &00FF, w którym zapisywane są parametry o znaczeniu decydującym o działaniu systemu.

Paperware - zapis na papierze
Inne określenie drukowanej kopii (hardcopy) wyników operacji komputerowej.

Parallel interface - interfejs równoległy, łącze równoległe
Łącze drukarki w 6128 jest łączem równoległym, co oznacza, że poszczególne linie danych magistrali systemowej są połączone z odpowiednimi liniami wejściowymi drukarki. Dane są dzięki temu przesyłane znacznie szybciej niż za pomocą łącza szeregowego, w którym przed wysłaniem każdy bajt musi być odpowiednio przetworzony do postaci szeregowej i obramowany impulsami synchronizującymi przebieg transmisji.

Pascal
Strukturalny język programowania wysokiego poziomu wymagający kompilowania programu przed jego wykonaniem lecz za to dostarczający programy bardzo szybko działające. Zalecany jako następny do opanowania po języku BASIC.

PEEK - (zerknięcie)
Funkcja języka BASIC która "zagląda" bezpośrednio do pamięci komputera i podaje zawartość określonej lokacji pamięci.

Peripheral - urządzenie peryferyjne
Drukarki, modemy, joysticki, magnetofony kasetowe - wszystko co dołącza się do komputera w celu zwiększenia jego możliwości.

Physical device - urządzenie fizyczne
Aktualnie istniejące urządzenie sprzętowe. Urządzenie fizyczne może być reprezentowane przez urządzenie logiczne.

Pixel
Najmniejszy obszar powierzchni ekranu jaki może być kontrolowany przez sprzęt.

Plotter - pisak x-y
Urządzenie do rysowania lub wykreślania, rejestrator cyfrowy x-y lub specyficzny rodzaj drukarki

POKE - (wpychać)
Instrukcja języka BASIC umożliwiająca wpisanie danych w określone komórki pamięci.

Port - brama
Niezależnie adresowane elementarne urządzenie interfejsowe do wprowadzania lub wyprowadzania danych.

Portability - przenośność
Znaczenie nieco inne niż potoczne, określa możliwość stosowania programów w różnych komputerach - zwykle w wyniku kompatybilności systemu operacyjnego, takiego jak np. CP/M firmy Digital Research.

Primitives - prymitywy
Procedury, operacje lub rozkazy wchodzące w skład Dr. Logo; procedury rezydentne (wbudowane).

Printer - drukarka
Urządzenie umożliwiające drukowanie tekstu w jakikolwiek sposób.

Procedure - procedura
Ciąg wyrażeń lub instrukcji programowych realizujących określone zadanie

Program
Zestaw instrukcji powodujących wykonanie zadania przez komputer. Może to być prosty program w kodzie maszynowym lub kompletny, złożony program aplikacyjny jak np. procesor tekstu.

Programming language - język programowania
Zbiór reguł formalnych wg których pisany jest program; zasady używania słów i liczb oraz kolejności ich wprowadzania.

PROM
Programmable Read Only Memory, programowana pamięć stała. Scalony układ pamięciowy do którego można jeden raz wpisać dane; dane te nie mogą już być później zmienione. (Patrz także FPROM)

Prompt - zachęta
Krótka wiadomość lub sekwencja znaków przypominająca użytkownikowi o oczekiwaniu na wprowadzenie określonego typu informacji. Na przykład, CP/M zachęca użytkownika znakiem > , a w Dr. Logo znakiem zachęty jest znak zapytania ?

PSU
Power Supply Unit, jednostka zasilająca. Zasilana z sieci elektrycznej powszechnego użytku wytwarza napięcia wymagane do zasilania komputera (i urządzeń peryferyjnych).

QWERTY Keyboard - klawiatura QWERTY
Potoczna nazwa określająca klawiaturę ze standardowym w USA i W.Brytanii układem klawiszy.

RAM
Random Access Memory, pamięć o swobodnym dostępie. Pamięć z której można czytać informacje i do której można zapisać informacje za pomocą wewnętrznych struktur komputera w trakcie normalnej procedury wykonywania programu.

Random access - swobodny dostęp
Możliwość odczytu i zapisu informacji w pamięci lub na dysku w dowolnie żądanym porządku.

Random number - liczba losowa
Liczba generowana przez program komputerowy w sposób przypadkowy tak, że nie może być przewidziana lub powtórzona. 6128 może generować sekwencje liczb pseudolosowych.

Raster
Podstawa tworzenia obrazu na ekranie z określonej liczby poziomych, kolejno wyświetlanych linii.

Read only R/O - "tylko do czytania"
Atrybut przypisywany dyskowi, zbiorowi dyskowemu lub napędowi dyskowemu w celu uniemożliwienia zapisu lub zmiany danych.

Read write R/W - "do zapisu i odczytu"
Atrybut przypisywany dyskowi, zbiorowi dyskowemu lub napędowi dyskowemu, umożliwiający zarówno odczyt jak i zapis danych.

Real number - liczba rzeczywista
Liczba zawierająca zarówno część całkowitą jak i część ułamkową tj. liczba w której występują cyfry po obu stronach punktu

Real time - czas rzeczywisty
Zdarzenia które występują "na Twoich oczach" w odróżnieniu od takich, które stają się widoczne tylko po zakończeniu wywołujących je procesów.

Record - rekord
Grupa bajtów w zbiorze. CP/M używa rekordów o długości 128 bajtów

Recursion - rekursja
Seria powtarzanych kroków w programie lub procedurze, w której wynik każdego powtarzanego cyklu jest dołączany do poprzedniego wyniku.

Refresh - odświeżanie
Proces podtrzymywania informacji na ekranie monitora lub w pamięci. Proces musi być nieniszczący lecz jedynie utwierdzający wszystko co już się znajduje w pamięci lub na ekranie.

Register - rejestr
Element pamiętający wewnątrz CPU, używany do tymczasowego przechowywania informacji.

Remark - uwaga
Nie wykonywana instrukcja w programie, zamieszczana w celu przypomnienia programiście co to za część programu lub umieszczenia daty bądź numeru kolejnej "edycji".

Reserved word - słowo zastrzeżone
Słowo które ma określone znaczenie w programie komputerowym i nie może być użyte inaczej niż w kontekście w jakim zostało wstępnie zdefiniowane. Na przykład BASIC nie dopuszcza stosowania słowa NEW jako oznaczenia zmiennej - słowo to jest "zarezerwowane" do innych celów.

Resolution - rozdzielczość
Możliwość określenia, gdzie kończy się jeden element wyświetlanego obrazu a gdzie zaczyna się drugi. Także używane do określenia dokładności operacji arytmetycznych prowadzonych przez komputer z dużymi liczbami.

Reverse Polish notation - odwrotna notacja polska
(RPN) Metoda zapisu operacji arytmetycznych, preferowana przez niektórych producentów kalkulatorów, w której operatory (+, -, *, /) są umieszczane za wartościami z którymi prowadzone są operacje.

RF Modulator - modulator wielkiej częstotliwości
Urządzenie za pomocą którego sygnał wizyjny z komputera jest przekształcany w sygnał wielkiej częstotliwości doprowadzany do wejścia antenowego standardowego odbiornika telewizyjnego.

ROM
Read Only Memory, pamięć stała, "tylko do czytania". Pamięć półprzewodnikowa, która raz zaprogramowana (w procesie wytwarzania) nie może być skasowana lub zapisana inną treścią.

Routine - program standardowy, procedura
Część programu, która wykonuje zadanie "rutynowe". Podprogram rezydujący wewnątrz głównego programu lub występujący jako oddzielny moduł do wykorzystania w wielu programach aplikacyjnych np. program zapewniający wyświetlanie czasu rzeczywistego (godziny, minuty itp.) w wyniku przetwarzania sygnału zegara systemowego.

RS232C
Określony standard łącza (interfejsu) szeregowego do transmisji danych. Urządzenia po obydwu stronach łącza wymagają specyficznych konfiguracji według szczegółowych warunków tego standardu. Porównaj z łączem (interfejsem) równoległym Centronics gdzie wzajemne połączenie jest zawsze z założenia standardowe.

Screen Editor - edytor ekranowy
Program redagujący - edytor tekstu lub programu w którym możliwe jest swobodne przesuwanie kursora w dowolne miejsce ekranu w celu zmiany występujących tam znaków.

Scrolling - rolowanie
Nazwa opisująca sposób w jaki treść wyświetlana na ekranie jest przesuwana w górę gdy po zapisaniu najniższej linii ekranu należy wytworzyć miejsce do wyświetlenia następnej linii.

Sector - sektor
Blok danych na dysku. W systemie dyskowym komputera AMSTRAD używany jest sektor o długości 512 bajtów.

Separator
Nazywany także "delimiter". Znak oddzielający, stanowiący rozgraniczenie między zastrzeżonymi słowami i innymi elementami programu lub danych.

Serial interface - interfejs szeregowy
Chociaż określenie to dotyczy prawie zawsze interfejsu RS232, istnieją także inne standardy łącz do szeregowej, sekwencyjnej transmisji danych do lub z komputera.

Simulation - symulacja
Technika symulacji wzajemnie oddziaływujących rzeczywistych procesów życiowych za pomocą komputera, np. symulacja lotu, symulacja kierowania samochodem itp.

Single side - jednostronny
Określenie dysku który umożliwia przechowywanie danych tylko po jednej stronie.

Soft key - klawisz definiowany
Patrz UDK - klawisz definiowany przez użytkownika

Software - oprogramowanie
Oprogramowanie wszelkiego rodzaju. Umieszczone na dysku, taśmie magnetofonowej, w pamięci ROM itp.

Software engineering - technika programowania
Wyrażenie określające programowanie komputera w sposób strukturalny i starannie przemyślany, w odróżnieniu od programowania "swobodnego".

Sound generator - generator dźwięku
Część komputera (sprzętowa lub programowa) wytwarzająca dźwięk i szum.

Speech synthesis - synteza mowy
Wytwarzanie symulowanej mowy przy użyciu odpowiedniego oprogramowania i sprzętu.

Spreadsheet - formularz
Program umożliwiający wprowadzanie cyfr do kolumn i wierszy tabel i wykonywanie na tych cyfrach operacji arytmetycznych. Zmiana jednej cyfry w tabeli powoduje jednoczesne wykonanie określonych operacji i aaktualnienie wszystkich wyników.

Sprite - chochlik
Znak na ekranie, poruszający się swobodnie po ekranie, wytwarzany przez specyficzny sprzęt i program w taki sposób, że jego pojawianie się i znikanie wygląda na przypadkowe.

Stack - stos
Obszar pamięci przeznaczony do przechowywania informacji "na stosie" tzn. w sposób umożliwiający odczytanie tylko ostatnio wprowadzonej informacji, znajdującej się "na wierzchu stosu".

Statement - dyrektywa
Instrukcja lub ciąg instrukcji w programie komputerowym

Stream - strumień (logiczny)
Droga użyta do wyprowadzania informacji z komputera np. na ekran, do drukarki lub do dysku.

String - ciąg znaków, tekst
Typ danych, które nie mogą być traktowane jako zmienna liczbowa. Ciąg znaków może zawierać nawet wyłącznie znaki cyfrowe, ale nie mogą one być bezpośrednio traktowane jako takie zanim nie zostaną zamienione w specyficzny sposób w wartość liczbową przez odpowiedni rozkaz.

Structured programming - programowanie strukturalne
Technika programowania w sposób logiczny i przemyślany, umożliwiająca uzyskanie programów przebiegających "od góry do dołu" w jasno opisanych krokach.

Variable - zmienna
Wielkość wprowadzona w programie komputerowym i oznaczona nazwą, której wartość może się zmieniać w czasie wykonywania programu.

Warm start - "ciepły start"
Reinicjalizacja systemu CP/M wykonywana po naciśnięciu klawiszy [CTRL] C. Reinicjalizuje system dyskowy i przekazuje sterowanie systemowi CP/M w celu przyjmowania dalszych poleceń.

Wildcard character - znak zastępczy
Gwiazdka * lub znak zapytania ?. W Dr. Logo tylko ?. Znak * oznacza dowolną potrzebną liczbę?. Przy wywoływaniu zbiorów znaki zastępcze są używane do utworzenia niejednoznacznej nazwy zbioru. Każdy znak ? w nazwie zbioru oznacza dowolną literę lub cyfrę. (Dosłownie "wildcard" to "karta dowolnej wartości").

Write protection - zabezpieczenie przed zapisem
Zabezpieczenie przed niepożądanym zapisem dysku lub zbioru dyskowego. Dysk lub zbiór dyskowy zabezpieczony przed zapisem jest "tylko do czytania".

XYZY
Magiczne słowo umożliwiające wyjście z tarapatów w grach przygodowych.

Subroutine - podprogram
Patrz hasło "Routine"

Syntax error - błąd składni
BASIC wysyła taki komunikat w przypadku napisania programu niezgodnie z regułami właściwego użycia słów kluczowych i zmiennych.

System tracks - ścieżki systemowe
Ścieżki dysku zarezerwowane dla systemu CP/M

Terminal
Klawiatura jako urządzenie wejściowe i monitor ekranowy lub dalekopis jako urządzenie wyjściowe

TPA
Transient Program Area, obszar programów przejściowych. W systemie CP/M obszar pamięci rozpoczynający się od adresu &0100, przeznaczony na programy użytkownika i przechowywanie danych.

Track - ścieżka
Koncentryczny pierścień na dysku po którym przesuwa się głowica. Każda ścieżka zawiera określoną, stałą liczbę sektorów. Sektory są wyznaczane na ścieżkach przez wpisanie specjalnych pól indeksowych w procesie formatowania dysku.

Transient program - program przejściowy
Program użytkowy systemu CP/M, np. PIP, ładowany do TPA i uruchamiany przez wprowadzenie jego nazwy za pomocą klawiatury.

Truncated - obcięty
Liczba lub ciąg tekstowy, skrócone przez odrzucenie pewnej liczby pierwszych lub ostatnich znaków. Gdy proces obcinania jest zamierzony, może powodować zaokrąglenie wartości liczby. Gdy proces taki jest niezamierzony, pewne znaki są po prostu pomijane aby umożliwić liczbie lub zmiennej tekstowej zmieszczenie się w przewidzianym dla niej miejscu.

Truth table - tablica prawdy
Wynikiem operacji logicznej może być "prawda" albo "fałsz". Komputer interpretuje te pojęcia jako 1 lub 0, a tablica prawdy przedstawia odpowiednio zbiór wszystkich możliwych wyników operacji logicznej jak np. IF A> B THEN C.

Turnkey - "gotowy po włączeniu"
Określenie używane do opisania programu wykonywanego samoczynnie po inicjalizacji systemu.

Turtle - żółw
Symbol graficzny w kształcie grotu strzały, stosowany jako kursor graficzny na ekranie graficznym w języku Lr. Logo.

Turtle graphics - grafika żółwia
Obraz graficzny pozostawiony na ekranie w wyniku ruchów żółwia. Przesuwający się żółw pozostawia na ekranie rysunek swojej drogi.

Turtle step - krok żółwia
Najmniejsza odległość jaką może przebyć żółw. Zwykle jeden elementarny punkt ekranu (pixel).

UDK
User Defined Keys, klawisze definiowane przez użytkownika. 6128 umożliwia definiowanie do 32 klawiszy w celu wykonywania różnych zadań.

Unsigned number - liczba bez znaku
Liczba bez żadnego znaku określającego czy jej wartość jest dodatnia czy ujemna.

Utility - program narzędziowy
Kompleksowy program umożliwiający wykonywanie określonych standardowych operacji jak np. sortowanie danych lub kopiowanie zbiorów.

Utility program - program użytkowy
Program na dysku, umożliwiający użytkownikowi wykonanie określonego zadania.

InterSoft

Spółka z o.o.

KATALOG INSTRUKCJI DO KOMPUTEROV AMSTRAD CPC6128 PCW8256 PCW8512

Tytuł instrukcji:	Jez:	Typ komputera:		Cena:
LocoScript t.1	POL		PCW	*5.000zł*
LocoScript t.2	POL		PCW	*20.000zł*
CP/M PCW8256	POL		PCW	*20.000zł*
Mallard BASIC	POL	CPC	PCW	*28.000zł*
Podręcznik użytkownika CPC464	POL	CPC		*10.000zł*
CPC BASIC	POL	CPC		*10.000zł*
Systemy operacyjne CPC	POL	CPC		*10.000zł*
CPC6128 Intern (opis układów komputera i oprogramowania firmowego)	POL	CPC		*20.000zł*
CPC464 Intern	FRA	CPC		*6.000zł*
CPC464 Firmware	ANG	CPC		*10.000zł*
DDI1 Firmware	ANG	CPC		*6.000zł*
AMSTRAD CP/M Plus	ANG	CPC	PCW	*20.000zł*

INSTRUKCJE DO PROGRAMOW:

Tytuł:	Jez:	Cena:	Tytuł:	Jez:	Cena:
ACE	ANG		CARDBOX	POL	
dBASE II PL	POL		Dr.CBASIC	POL	
Dr.Draw	POL		Dr.Graph	POL	
Fleet Street Ed.	ANG		Flexi File	ANG	
Flexi Writer	ANG		FORTRAN 80	POL	
GEM database	ANG		HardDump	POL	
Heavy (Morton)	ANG		HiSoft C	POL	
HiSoft DEVPAC	POL		HiSoft PASCAL	POL	
JTR PASCAL			Laser BASIC	POL	
MACRO 80	ANG		MASTERCALC	POL	
MASTERFILE	POL		MiniOffice II	ANG	
MultiPlan	POL		MBASIC	ANG	
Nevada FORTRAN	ANG		NewWord 2	POL	
ODDJOB	POL		PASCAL MT+	POL	
Perfect Calc			Perfect Writer		
Poly Plot	POL		Profi-Painter	POL	
PROTEXT	POL		Screen Designer	POL	
SuperCalc II	ANG		TAJFUN		
TASCOPY	POL		TASWORD	POL	
Turbo PASCAL	POL		Turbo GRAPHICS	POL	
WordStar	POL				

Zapraszamy również po programy pisane na zamówienie użytkownika: Bazy danych, programy kosztorysowe, magazynowe, personalne, kartoteki, programy statystyczne itp.

Do komputerów serii CPC i PCW oferujemy programy rozrywkowe i gry (około 300 pozycji do CPC, a do PCW takie programy jak szachy, brydż, gry symulacyjne, przygodowe, zręcznościowe oraz wiele innych).

Dysponujemy poza tym wieloma programami o charakterze edukacyjnym, czy raczej poglądowym - proste BASICowe bazy danych, edytory tekstów typu BASI-ANSWORD, programy graficzne itp. Nie umieściliśmy ich w katalogu cenowym ze względu na niską wartość użytkową. Jeśli jednak będą Ci do czegoś potrzebne - zapraszamy.

Tylko na wystawie !!! Przyjmujemy subskrybcje na podręcznik CPC6128 INTERN w cenie o 50% niższej - 10.000zł

InterSoft sp.z o.o. 00-478 WARSZAWA Al.Ujazdowskie 16 m 1. tel 28 01 76

Cena Tom I/II zł 2000.–

SOEM

„BIT"

6111