

ice  **felix**®
COMPUTER S.A.

HC 91



Str. G. Constantinescu 2
78009 București 2
Tel.: 688 22 95 , 688 23 60,
688 46 75 , 688 61 25
687 53 02, 688 23 60

Fax : 687 62 20 , 312 87 50
Tlx : 11626 felix r

MANUAL DE UTILIZARE

CUPRINS

Capitolul 1.	
INTRODUCERE	5
Prezentare generala, Caracteristici tehnice, Instalare, Tastatura, Limbaje de programare, Cite ceva despre HC-uri	
Capitolul 2.	
ELEMENTE DE PROGRAMARE SI EDITARE	10
Utilizarea tastaturii, Modul de afisare, Programe, linii de program si editare.	
Capitolul 3.	
LIMBAJUL BASIC	15
Variabile si expresii aritmetice, Siruri de caractere, Tablouri, Initializarea variabilelor, Operatii logice, Functii, Iteratii, Subrutine, Generarea numerelor aleatoare, Setul de caractere, Grafice, Instructiuni I/E, Culori, Miscare, Memoria, Producerea sunetelor, Utilizarea codului masina, Utilizarea porturilor I/E, Inregistrarea pe caseta, Imprimanta, Variabile de sistem, Canale I\ E si cai, Alte echipamente.	

INAINTE DE ORICE VERIFICATI CONFIGURATIA STANDARD:

- unitatea centrala cuprinzind si tastatura
- sursa de alimentare (alimentatorul)
 - cablu pentru televizor
 - cablu pentru casetofon
 - prezentul manual
 - caseta de demonstratii.

Capitolul 1. INTRODUCERE

1.1. Prezentare generala

Stimate cumparator, acest manual este facut cu intentia de a ghida primii pasi in utilizarea calculatoarelor din familia Home Computers - microcalculatoare cu destinatie educationala, divertisment, calcule stiintifice si ingineresti.

Un calculator personal este folosit de o singura persoana, spre deosebire de alte tipuri de calculatoare (micro sau mini sisteme) la care pot lucra simultan mai multe persoane. Calculatoarele personale sint si ele de doua feluri:

- calculatoare personale profesionale **PERSONAL COMPUTER**;
- calculatoare personale familiale **HOME COMPUTER**.

Acestea din urma au un pret accesibil pentru a putea fi cumparate pentru acasa. Calculatoarele tip HC fac parte din aceasta grupa.

Manualul se adreseaza tuturor, fara a cere o pregatire in electronica sau informatica. El nu va arata cum se construiesc un calculator ci din ce este format, cum se foloseste si ce se poate atasa la el pentru a-i putea imbunatati performantele.

1.2. Caracteristici tehnice

Un calculator HC este construit cu microprocesorul Z 80 A CPU, procesor pe 8 biti pe magistrala sa de date si 16 biti pentru magistrala de adrese. Astfel Z 80 poate adresa 64K memorie si 64K spatiu aditional dedicat dispozitivelor de intrare-iesire. Prin urmare HC-ul este alcatuit din:

- CPU/MEMORIE:-Z 80 A - microprocesor pe 8 biti cu ceas de 3,5 MHz;
- 16K - memorie ROM constituind interpretorul BASIC;
- 64K - memorie RAM din care 48 disponibila BASIC.

TASTATURA: - 40 taste similare cu masina de scris exceptind literele Q,Z,M.

DISPLAY: - afisare pe televizor alb/negru sau color PAL pe canalul 8, monitor RGB sau monitor PAL.

- rezolutie: 192*256 pixeli (24*32 caractere).
- realizeaza punct, linie, cerc, arc de cerc de inalta rezolutie grafica.
- 16 caractere grafice predefinite, 21 de posibilitati de definire grafica.
- textul scris pe SCREEN are 32 caractere pe 24 linii.

SUNET: - sunetul auzit in difuzorul calculatorului cuprinde circa 10 octave realizate prin comanda BASIC: BEEP.

CULORI: -detaliile in plan apropiat cit si in plan indepartat se realizeaza prin culoare, stralucire si flash cu setul de instructiuni: INK, PAPER, BORDER, BRIGHT si FLASH.

- codul culorilor este controlabil de la tastatura.
- comanda INVERSE 1 inverseaza fundalul cu cerneala, iar OVER 1 realizeaza supraimprimarea.

INTERFETE: - interfata casetofon, 1500 bauds.

- extensie porturi.

OPTIUNI: interfata de disc flexibil, imprimanta, retea.

SOFTWARE: - interpretor BASIC 16K in scris in memorie eprom.
- LOGO, FORTH, PASCAL, BETABASIC si altele pe caseta.
- JOCURI pe caseta.

1.3. Instalare

Calculatorul se alimenteaza prin intermediul alimentatorului de +9v de la retea de curent alternativ de 220v.

Pentru punerea in functiune si utilizarea calculatorului urmariti secventa de mai jos:

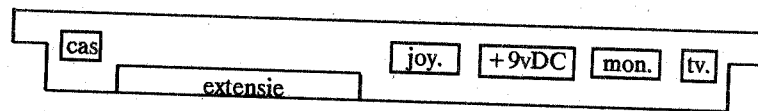


Fig.1.1

1. Introduceți fișa alimentatorului în mufa notată +9vDC din spatele calculatorului.
2. Introduceți stecherul alimentatorului într-o priză de curent alternativ 220v/50Hz. Din acest moment HC-ul funcționează. Dacă apăsați tastele auziți bipuri sonore. Dacă nu le auziți apăsați butonul RESET și încercați din nou. Butonul RESET se află așezat în partea dreaptă a tastaturii, ceva mai jos decât aceasta, pentru a nu fi atins din greșeală, în timpul lucrului.

În momentul în care apăsați tastele, calculatorul primește comenzile dumneavoastră. Pentru a putea dialoga aveți nevoie de un dispozitiv de afișare. Cel mai simplu este un televizor alb/negru sau color.

3. Conectați cablul de televizor. Introduceți mufa RCA în locul notat tv, iar celălalt capăt în mufa televizorului.

ATENȚIE: Nu recomandăm folosirea televizoarelor pe tuburi.

4. Porniți televizorul și acordați-l pe canalul 8 pînă ce obțineți o imagine clară.

După instalare, pe ecran, în partea de jos, trebuie să apară un mesaj care reprezintă numele calculatorului, firma constructoare și anul de concepție al modelului. În caz că nu apare apăsați din nou butonul RESET. Din acest moment calculatorul este pregătit pentru dialog.

Dacă va aflați pentru prima dată în fața unui astfel de calculator e bine să aflați mai întîi posibilitățile sale, resursele hard și soft. Cel mai simplu pentru aceasta este folosirea casetei de demonstrații.

Pentru aceasta va este necesar un casetofon. Acesta nu trebuie să fie foarte sofisticat. Este necesar să prezinte o mecanică sigură, fără fluctuații de bandă și un cap cit mai puțin uzat și reglat pe un azimut corespunzător (asa cum îl livrează fabricantul).

5. Introduceți cablul de casetofon în calculator în locul notat cas, iar capătul celălalt în casetofon pe mufa LINE.

6. Introduceți caseta în casetofon și poziționați-o la început.

7. Priviți tastatura și localizați următoarele taste: J, P, SS, CR. Apăsați J apoi țineți cu un deget SS și apăsați de două ori tasta P. Pe ecran apare LOAD^m. Apăsați apoi CR.

Ecranul se face alb.

8. Porniți casetofonul. În acest moment calculatorul va schimba BORDER-ul în albastru și apoi în roșu. Pentru posesorii de televizoare alb/negru dintr-o culoare deschisă într-una mai închisă și invers, pînă cînd începe programul pe caseta. Din acest moment pe BORDER apar dungă colorate. Se încarcă programul. Urmăriți și executați mesajele de pe ecran. După ce v-ați familiarizat cu tastatura și caseta citiți mai departe manualul, dar mai înainte să lamurim niște termeni folosiți anterior:

-ROM: (Read-Only-Memory) este o memorie al cărui conținut este stabilit prin fabricație și care nu poate fi schimbat ci numai "citit". Veti constata că de cîte ori scoateți de sub tensiune calculatorul interpretorul BASIC nu dispăre.

-RAM: (Random-Access-Memory) este memoria de lucru curent a calculatorului. În ea se poate "scrie" și "citi" ceea ce doriți, ori de cîte ori apăsați calculatorul, atîta timp cît acesta este alimentat. La întreruperea alimentării pierde ce are înscris prin program de dumneavoastră.

-HARDWARE: Specialiștii numesc echipamentele ce alcătuiesc calculatorul în totalitatea lor.

-SOFTWARE: Este tot ce reprezintă programe.

Pentru a scrie ceva avem nevoie de hîrtie și cerneală. Pentru a defini acest lucru notăm partea activă a ecranului (SCREEN) cu PAPER = hîrtie, coală; ceea ce scriem notăm INK, iar pentru a separa PAPER-ul de marginile ecranului care ar putea ascunde la colțuri notițele noastre, folosim BORDER-ul care centrează PAPER-ul în așa fel încît în orice caracter de pe ecran să fie vizibil.

1.4. Tastatura

După cum desigur ați observat calculatorul are un număr de 40 de taste. Tastatura seamănă foarte mult cu claviatura unei mașini de scris, însă este mai complicată deoarece fiecare tastă are cel puțin șase semnificații. Prin semnificații înțelegem litere mici sau mari, cifre, caractere speciale (de exemplu +, -, ?, *, \$, %, etc.) sau cuvinte cheie (de exemplu INPUT, PRINT, RUN, etc.). Cuvîntul cheie este un cuvînt în limba engleză care are o semnificație foarte precisă pentru calculator. Pentru exemplificare tasta i are următoarele semnificații: i, I, INPUT, AT, CODE, IN.

Pentru a alege semnificația dorită de pe o tastă trebuie să cunoaștem "modurile de lucru ale calculatorului".

CODE	AT
I	
IN	INPUT

Calculatorul are cinci moduri de lucru: K, L, C, E și G.

Modul de lucru în care se găsește calculatorul ne este indicat de o literă mare clipitoare numită "cursor". Cursorul ne indică și locul de pe ecran unde va apărea următoarea semnificație.

MODUL K: modul cuvintelor cheie "keywords"
 Daca sintem in modul K si apasam o cifra, pe ecran apare cifra respectiva, cursorul raminand in K. Daca apasam o litera, pe ecran apare cuvintul cheie din dreapta jos a tastei (de exemplu INPUT pentru tasta i), cursorul trecind automat in modul L.

Atentie! Intotdeauna cuvintele cheie vor fi scrise direct apasind tasta corespunzatoare si nu litera cu litera.

In modul K se intra in urmatoarele cazuri:

- la inceputul fiecărei linii
- dupa semnul : (doua puncte), care separa instructiunile de pe aceiasi linie.
- dupa cuvintul cheie THEN.

MODUL L: literele mici si mari. Apare imediat dupa modul K sau E. In modul L, daca se apasa o cifra apare cifra respectiva, iar daca se apasa o litera apare litera mica respectiva, cursorul raminand in L. Daca dorim sa scriem litere mari atunci apasam simultan CAPS SHIFT (CS) si tasta respectiva.

MODUL C: numai litere mari, (capitals). Daca dorim sa scriem numai cu litere mari intram in modul C apasind simultan CS si 2.

Din C se iese apasind iarasi CS si 2.

OBSERVATIE: Daca dorim sa scriem semnificatia din dreapta sus trebuie sa fim in unul din modurile K,L sau C si sa apasam simultan SYMBOL SHIFT (SS) si tasta respectiva.

MODUL E: extins. Se utilizeaza pentru a scrie semnificatiile din stanga sus si jos ale tastelor. In modul E se intra apasind simultan CS si SS. Pentru a scrie semnificatia din stanga sus intram in modul E, dupa care apasam tasta corespunzatoare.

Pentru a scrie semnificatia din stanga jos intram in modul E, dupa care apasam simultan SS si tasta corespunzatoare. Dupa prima apasare pe tasta cursorul trece din modul E in modul L.

MODUL G: grafic "graphics". Apare dupa ce se apasa simultan CS si 9 si tine pina cind se apasa 9 sau iarasi CS si 9. In modul G se pot scrie simboluri grafice "mozaic", folosind tastele 1-8 cu si fara CS. Tasta 0 se utilizeaza pentru a sterge caracterul din stanga cursorului. Tot in modul G putem sa definim propriile noastre caractere grafice.

Daca o tasta este apasata mai mult de 0,7 secunde, ea va fi scrisa in mod repetat atita timp cit o apasam.

Ceea ce scriem la tastatura va apare in partea de jos a ecranului in timp ce se tasteaza, fiecare caracter fiind inserat chiar inaintea cursorului. Cursorul poate fi deplasat spre stanga cu CS si 5, si spre dreapta cu CS si 8, fara a sterge caracterele respective. Caracterul dinaintea cursorului poate fi sters indiferent in modul in care ne gasim, cu DELETE (CS si 0).

OBSERVATIE Tot ce am scris in partea de jos a ecranului poate fi sters apasind EDIT (CS si 1) urmat de ENTER (CR).

MODUL E: se apasa tasta

MODUL L si C

MODUL E: sa apasa SS si tasta

CODE	IN
I	
IN	INPUT

MODUL K, L si C
se apasa SS si tasta

MODUL K se apasa tasta

1.5. Limbaje de programare

Un calculator poate sa faca practic orice. Important este ca noi sa stim sa-i spunem ceea ce trebuie sa faca. Acest lucru se face prin realizarea unui "program". Programul reprezinta o insiruire de instructiuni asezate intr-o ordine foarte precisa, prin intermediul caruia dirijam calculatorul pas cu pas in ceea ce trebuie sa faca. Bineinteles ca cel putin deocamdata calculatorul nu intelege "limbajul natural", limbajul in care comunicam noi oamenii, de aceea fiind necesar sa invatam "limba" pe care o stie el si care se cheama "limbaj de programare".

Deoarece la ora actuala in lume exista mii de tipuri de calculatoare, cred ca intelegeti necesitatea existentei celor peste doua sute de limbaje de programare. De ce asa de multe? Nu era suficient un singur limbaj de programare?

Raspunsul consta in faptul ca de obicei un limbaj de programare acopera, cu eficienta maxima, doar un domeniu, fiind mai putin eficient in celelalte. De exemplu limbajul FORTRAN (FORMula TRANslator) este cel mai potrivit limbaj pentru rezolvarea problemelor tehnico-stiintifice. Pentru probleme de gestiune, deci economico-financiare, cel mai cunoscut este limbajul COBOL (COMmon Business ORiental Language).

Calculatoarele HC, care sint asa cum am aratat, calculatoare pentru acasa, deci pentru publicul larg, folosesc un limbaj accesibil tuturor numit BASIC (Beginners All-purpose Symbolic Instruction Code). Aceasta in romaneste s-ar putea traduce: limbaj de programare pentru incepatori. Dupa cum ii spune si numele, acest limbaj de programare poate fi invatat de toti cei care doresc sa patrunda in universul fascinant al calculatoarelor. Daca vreti sa utilizati calculatorul ca "beneficiar" cu programe "gata facute" nu mai aveti practic multe de invatat. Un scurt ghid de BASIC este suficient. De obicei insa in practica doriti sa aveti unele facilitati cu ajutorul calculatorului d-vs si pentru aceasta este necesar sa cunoasteti cit mai multe. In felul acesta puteti sa faceti singuri aceste programe. Pentru programe simple consultati manualul BASIC pentru HC sau altele care folosesc acelasi limbaj de programare (ZX Spectrum, CIP, Spectim, etc.). Pentru programe mai evolute folositi limbajul cod masina.

1.6. Cite ceva despre HC-uri

Asa cum am aratat mai sus calculatoarele HC lucreaza in limbaj BASIC. Acest interpretor este compatibil Sinclair. Din acest punct de vedere toate tipurile de HC-uri sint perfect compatibile BASIC. Singurele diferente sint din punct de vedere hard.

Va veti intreba probabil, si pe buna dreptate, daca sint compatibile soft, care este totusi diferenta intre HC85 si HC90, apoi HC90 si HC91 etc.? Raspunsul este foarte simplu: FELIX COMPUTER SA isi rezerva dreptul de a aduce imbunatatiri in constructia si functionarea calculatoarelor. Asadar, stimate cumparator, va lasam placerea sa descoperiti micile imbunatatiri aduse de la un calculator la altul. In cazul modificarilor substantiale este datoria firmei sa samnaleze acest lucru. Astfel HC90 fata de HC85 este reproiectat cu memorii de capacitate mai mare, miciorind numarul de componente si consumul, iar HC91 fata de HC90 are avantajul ca poate lucra in CP/M atunci cind se monteaza intefata 1.

Capitolul 2. ELEMENTE DE PROGRAMARE SI EDITARE

2.1. Utilizarea tastaturii

Am aratat in capitolul precedent, ca tastatura HC-ului este similara unei masini de scris. Am mai aratat ca o tasta are pina la sase semnificatii. Cum se tasteaza fiecare functie v-ati familiarizat deja in urma lecturarii primului capitol si vizionarii casetei de demonstratii. Mai ramine de mentionat ca la inscrierea simbolurilor pe tastatura au fost folosite urmatoarele prescurtari:

RAND	in loc de	RANDOMIZE
BRGT	in loc de	BRIGHT
INV	in loc de	INVERSE
CR	in loc de	ENTER
CS	in loc de	CAP SHIFT
SS	in loc de	SIMBOL SHIFT
SCR\$	in loc de	SCREENS
C O N T	in loc de	CONTINUE

2.2. Modul de afisare

Ecranul de afisare are 24 de linii, fiecare cu 32 de caractere. Ecranul are doua parti. Partea de sus de 22 de linii e folosita pentru listarea instructiunilor sau a rezultatelor programului. Cind aceasta parte este plina, calculatorul face "scroll". Pentru a putea vedea toate liniile, calculatorul se opreste si apare mesajul "scroll?".

Apasarea tastelor N, SPACE sau STOP va intrerupe programul si va afisa mesajul:

D BREAK - CONT repeats

Orice alta tasta determina calculatorul sa faca scroll. Partea de jos a ecranului este folosita pentru comenzi de intrare, linii de program, tiparirea datelor de intrare cit si pentru mesaje.

2.3. Programe, linii de program si editarea programelor utilizind EDIT si sagetile, RUN, PRINT, STOP, IN, INPUT, DATA, BREAK.

Limbajul BASIC admite doua tipuri de instructiuni: numerotate si nenumotate. Instructiunile nenumotate sint executate imediat dupa apasarea tastei CR. Instructiunile numerotate sint stocate ca linii de program. Numerele de linie trebuie sa fie intregi, intre 1 si 9999. Listarea si executia unui program se fac prin ordonarea programului dupa numarul de linie. De aceea este indicat ca la scrierea unui program sa se pastreze spatii intre numerele a doua linii consecutive, dind astfel posibilitatea inserarii cu usurinta de linii noi. O linie de program poate contine una sau mai multe instructiuni. Separarea instructiunilor dintr-o linie se face cu caracterul " "

Cursorul indica linia curenta asupra careia se pot face modificari sau dupa care se pot insera alte linii. De obicei, cursorul se afla pe ultima linie introdusa, dar pozitia lui poate fi deplasata in sus sau in jos prin apasarea simultana a tastei CAPS SHIFT si a sagetilor.

In continuare vor fi prezentate exemple de programe in care sint trecute in revista cîteva instructiuni BASIC, punindu-se accentul pe facilitatile de editare ale sistemului.

EXEMPLUL 1. Sa se tipareasca suma a doua numere.
Dupa ce se vor introduce liniile (in ordinea mentionata):

```
20 PRINT a
10 LET a = 10
```

Se constata ca programul se tipareste pe ecran in permanenta ordonat dupa numarul de linie. Pina acum s-a introdus primul numar. Pentru a-l introduce pe al doilea, se scrie linia:

```
15 LET b = 15
```

Pentru tiparirea sumei, este necesar ca linia 20 sa aiba forma:

```
20 PRINT a + b
```

S-ar putea rescrie linia, dar este mai usor sa se faca uz de facilitatea EDIT. Pentru aceasta se coboara cursorul de la linia 15 la linia 20, actionind tasta ↓. In continuare se actioneaza tasta EDIT; in partea de jos a ecranului va apare o copie a liniei curente (in exemplul prezentat, linia 20). Se actioneaza tasta → pina cind cursorul L se deplaseaza la sfirsitul liniei si apoi se introduc + b (fara CR).

Ultima linie a ecranului va arata acum astfel:

```
20 PRINT a + b
```

Cu CR, vechea linie 20 va fi inlocuita cu cea noua. Se executa acest program utilizind RUN si CR; ca urmare pe ecran va apare afisat rezultatul operatiei a + b. Apasind RUN si CR programul este executat identic. Dupa terminarea executiei programului ramine inregistrata in memorie ultima valoare a fiecarei variabile din program. Ele pot fi vizualizate printr-o instructiune PRINT neetichetata. Aceasta operatie este necesara la depanarea programului. Pentru a sterge ultima linie a ecranului se utilizeaza EDIT. Se introduce o succesiune de caractere (fara CR) care vor fi sterse folosind una din metodele:

1. actionarea tastei DELETE pina cind linia este stearsa in intregime.
2. actionarea tastei EDIT; pe ultima linie a ecranului apare o copie a liniei curente. Cu CR acum, linia curenta ramine nemodificata, iar ultima linie a ecranului este stearsa.

Presupunem ca se introduce din greseala linia:

```
12 LET b = 8
```

Ea va putea stearsa scriind:

```
12 (cu CR desigur).
```

Se observa ca a disparut cursorul programului. Daca se actioneaza ↑, cursorul va apare pe linia 10, in timp ce daca se actioneaza ↓ va apare la linia 15. Se scrie:

```
12 (si CR)
```

Din nou cursorul programului va fi ascuns intre liniile 10 si 15.

Actionind acum EDIT, linia 15 va apare in zona de editare. Cind cursorul programului este ascuns, intru doua linii, EDIT aduce in josul ecranului linia care are numarul de linie imediat urmator. Se scrie acum:

30 (si CR)

De aceasta data cursorul este ascuns dupa sfirsitul programului.
Cu comanda:

LIST 15

pe ecran se obtine:

```
15 LET b = 15
20 PRINT a + b
```

Instructiunea LIST 15 produce listarea incepind cu linia 15 si pune cursorul programului la linia 15. Pentru un program foarte lung, LIST va fi o metoda mai utilizata de mutare a cursorului decit sagetile. Aceasta ilustreaza o alta utilitate a numerelor de linie; ele actioneaza ca nume ale liniilor de program astfel incit se pot face referiri la ele in acelasi mod in care se fac referiri la numele de variabile. LIST (neurmat de un numar) determina listarea de la inceputul programului.

O alta comanda este NEW. Efectul ei consta in stergerea programelor si variabilelor din memoria calculatorului.

EXEMPLUL 2. Sa se scrie un program care transforma temperatura din grade Fahrenheit in grade Celsius.

```
10 REM conversia temperaturii
20 PRINT "grade F", "grade C"
30 PRINT
40 INPUT "introduceti gradele F. ", f
50 PRINT f, (f-32)*5/9
60 GO TO 40
```

Este necesar sa fie introdusa pe rind fiecare litera pentru a obtine "conversia temperaturii" in linia 10. In linia 60 se obtine GO TO actionind tasta G (desi contine spatiu, GO TO constituie un singur cuvint cheie).

Rulind programul, se va vedea pe ecran capul de tabel tiparit de linia 20. Linia 10 este ignorata de calculator, instructiunea REM introducind un comentariu in textul sursa. Comanda INPUT din linia 40 asteapta sa fie introdusa o valoare pentru variabila F; se introduce un numar si se actioneaza apoi CR. Calculatorul afiseaza rezultatul si nu se opreste din rulare, ci asteapta alt numar (datorita saltului din linia 60). Programul se poate opri prin actionarea tastei STOP in momentul in care pe ecran apare scris:

Introduceti gradele F.

Calculatorul intoarce mesajul:

H STOP in INPUT 40:1

care precizeaza de ce si unde s-a oprit din rulare (in prima instructiune din linia 40). Pentru a continua programul se introduce CONTINUE si calculatorul va astepta alt numar. CONTINUE determina rularea programului de la linia de la care se oprise executia (linia 40). Se scrie linia 60 sub forma:

60 GO TO 31

In executie aceasta varianta se comporta identic cu varianta precedenta. Daca numarul liniei intr-o comanda GO TO se refera la o linie inexistentă, atunci se sare la linia imediat următoare numărului dat. Acest lucru este valabil și pentru comanda RUN (de fapt RUN are același efect cu RUN 0). Dacă tipărim numere până când se umple ecranul calculatorului va muta întreaga parte de sus a ecranului cu o linie pentru a face loc, pierzind astfel capul de tabel. Când am terminat de tipărit, programul se poate opri cu STOP urmat de CR. Lista de instrucțiuni a programului se poate afișa după întrerupere apăsând CR. Se analizează instrucțiunea PRINT din linia 50. Virgula utilizată aici determină începerea tipăririi fie în marginea din stânga, fie în mijlocul ecranului, în funcție de ce urmează după virgula. În acest caz tipărirea temperaturii în grade Celsius are loc în mijlocul liniei.

Caracterul punct și virgula ";", determină tipărirea sirului următor imediat după sirul precedent. Se poate vedea aceasta dacă în linia 50 e înlocuit caracterul " " cu ";". Alt semn de punctuație ce poate fi utilizat în comenzi PRINT este apostroful "'". El determină saltul cursorului la începutul liniei următoare și continuarea tipăririi din acel punct, ca și cum elementele despartite prin "' ar fi fost sub incidenta unor comenzi PRINT succesive. Pentru ca instrucțiunea PRINT să nu determine saltul la linia următoare este necesar ca PRINT-ul precedent să se termine cu " " sau cu ";". Pentru exemplificare să se substituie linia 50 pe rând cu liniile:

```
50 PRINT f,
50 PRINT f;
50 PRINT f
50 PRINT f
```

Se constată că varianta cu " " împarte totul în două coloane, cea cu ";" scrie totul compact, cea fără semn de punctuație și cea cu "' scriu un număr pe o linie. În memorie pot exista simultan mai multe programe cu condiția ca numerele de linie să fie în intervale disjuncte.

EXEMPLUL 3.

```
100 INPUT n$
110 PRINT "Salut ";n$;" !"
120 GO TO 100
```

Acesta este un program care poate coexista în memorie cu programul din exemplul 2⁴ intrucit unul are numerele de linie în intervalul 0...60, iar celălalt în 100...120. Pentru lansarea în executie a programului din exemplul 3 se da comanda RUN 100. Executia unei comenzi RUN determină stergerea ecranului și a tuturor variabilelor, după aceasta executind sirul instrucțiunilor programului. Dacă nu se dorește inițializarea variabilelor și stergerea ecranului se poate utiliza comanda GO TO 100.

La executia programului din exemplul 3 se observă că pe ecran apare "L" care indică faptul că se dorește citirea unui sir de caractere. Sistemul admite ca o instrucțiune INPUT să se comporte similar cu o instrucțiune de atribuire, dar numai pentru cazul citirii de variabile de tip sir de caractere. Pentru aceasta se șterg ghilimelele (utilizând ← și DELETE) și se introduce numele unei variabile de același tip. Introducerea unui nume de variabilă determină cautarea valorii acelei variabile ce trebuia citită de la tastatură.

De exemplu dacă la executia programului din exemplul 3 la prima solicitare de sir de caractere se introduce "ANA", valoarea variabilei n\$ va deveni n\$ = "ANA" la următoarea

citire se introduce "MARIA", n\$ devine n\$ = "MARIA". La executia urmatoarei instructiuni INPUT se va introduce n\$; in acest caz se cauta valoarea vechii variabile n\$ si i se asociaza variabilei n\$.

Deci comanda se compota similar cu LET n\$ = n\$ in urma acestei instructiuni va fi n\$ = "MARIA", deci instructiunea PRINT din linia 110 va tipari:

Salut MARIA !

Uncori din greseala se scrie un program ce ruleaza la infinit, cum este urmatorul:

```
200 GO TO 200  
RUN 200
```

Pentru oprirea executiei se actioneaza BREAK (CAPS SHIFT si SPACE) si calculatorul raspunde cu mesajul:

L BREAK into program, 200:1

La sfirsitul fiecarei instructiuni programul verifica daca aceste taste sint actionate; daca da, este oprita rulara. Tasta BREAK poate fi utilizata de asemenea cind sint conectate casetofonul sau imprimanta, in cazul cind calculatorul asteapta ca aceste periferice sa efectueze o comanda. Mesajul produs in acest caz este diferit:

D BREAK - CONT repeats.

Comanda CONTINUE in cazul lucrului cu casetofonul sau imprimanta repeta instructiunea unde programul a fost oprit.

Listingerile automate sint acelea care nu rezulta in urma unei comenzi LIST, ci au loc dupa introducerea unei linii noi. De retinut este faptul ca linia curenta (cea cu) apare intotdeauna pe ecran si in mod normal in pozitia centrala. Calculatorul memoreaza numarul liniei curente si de asemenea, al primei linii din partea de sus a ecranului. Cind incearca sa listeze, primul lucru pe care-l face este sa compare prima linie de pe ecran cu linia curenta. Daca prima linie de pe ecran este mai mare decit linia curenta, atunci cursorul va apare pe prima linie a ecranului. Astfel listarea consta in tiparirea pe ecran in mod defilare a programului cuprins intre prima linie si linia curenta.

Oricum, mai intii se efectueaza un calcul aproximativ pentru a vedea cit timp ia listarea si daca aceasta este prea lung, linia din virf se muta mai jos pentru a fi mai aproape de linia curenta. Acum, avind stabilita linia din virf, listarea poate incepe. Daca linia curenta a fost listata, listarea se opreste cind s-a ajuns la sfirsitul programului sau la partea de jos a ecranului.

Capitolul 3. LIMBAJUL BASIC

3.1 Variabile si expresii aritmetice

Cuprins: Sume de variabile, expresii, notatii

Operatii: +, -, *, /

Versiunea BASIC a calculatoarelor HC admite pentru variabilele numerice nume formate din oricite caractere (litere sau cifre), care incep cu o litera. Printre caractere poate fi si blankul, care este insa ignorat. Prezenta lui face variabila mai usor de citit. Sistemul face filtrarea literelor mari, astfel incit atit litera mare cit si litera mica corespunzatoare sint interpretate la fel. Nu este indicata folosirea numelor foarte lungi deoarece sint greu de manipulat.

Variabilele speciale sint:

1. Variabilele folosite in instructiunile FOR, care trebuie sa fie reprezentate printr-o singura litera.

2. Variabilele de tip sir de caractere, al caror nume este format dintr-o litera urmata de "\$".

Expresiile numerice pot fi reprezentate si printr-un numar zecimal urmat de un exponent.

Exemplul 1. Sa se tipareasca numerele:

```
PRINT 2.3e0  
PRINT 2.34e1
```

si asa mai departe pina la

```
PRINT 2.34e15
```

Se observa ca dupa un timp calculatorul incepe sa foloseasca scrierea cu exponent deoarece nu se pot utiliza mai mult de 14 caractere consecutive pentru scrierea unui numar.

Se poate tipari in mod similar:

```
PRINT 2.34e-1  
PRINT 2.34e-2
```

si asa mai departe. Comanda PRINT afiseaza numai 8 cifre semnificative.

Exemplul 2.

```
PRINT 4294967295,4294967295-429e7
```

Acest exemplu demonstreaza ca toate cifrele numarului 4294967295 sint memorate, desi nu toate pot fi tiparite pe ecran.

HC-ul utilizeaza scrierea numerelor in virgula mobila.

Numerele sint reprezentate cu precizie de aproximativ noua cifre si jumatate. Cel mai mare intreg ce poate fi reprezentat cu precizie in memorie este $2e32-1 = 4294967295$.

Exemplul 3.

```
PRINT 1e10+1-1e10,1e10-1e10+1
```

Rezultatele afisate vor fi:

```
0 1
```

deoarece $1e10 + 1$ si $1e10$ au aceeasi reprezentare interna.

Operatiile aritmetice executate de calculator sint inmultirea, impartirea, adunarea si scaderea. Operatiile de inmultire "*" si impartire "/" au prioritate egala. De aceea, o expresie ce contine numai inmultiri si impartiri se executa de la stanga la dreapta. Adunarea si scaderea au de asemenea, prioritate egala dar mai mica decit a inmultirii si a impartirii. Pentru a modifica ordinea de executie a operatiilor se folosesc parantezele.

3.2 Siruri de caractere

Cuprins: Operatii cu siruri de caractere

Sirurile de caractere sint reprezentate prin secvente de caractere ASCII, incadrate intre ghilimele ("). Daca se doreste tiparirea in text a caracterului ghilimele, el trebuie sa fie dublat. Un sir de caractere poate fi atribuit ca valoare unei variabile sir sau poate fi tiparit cu o comanda **PRINT**.

Fiind dat un sir, un subsir al lui consta in citeva caractere consecutive continute in el, luate in secventa. De exemplu "string" este un subsir al lui "bigger string", insa "b string" nu este. Manipularea subsirurilor in BASIC se face cu:

```
s(n1 TO n2)
```

unde:

1. s este un sir de caractere sau o variabila sir.
2. n1,n2 sint numere intregi nenegative ce reprezinta ordinul caracterului de inceput, respectiv de sfirsit, din subsir. Daca n1n2, rezultatul este sirul vid ("").

Daca nu se precizeaza inceputul si/sau sfirsitul subsirului se iau implicit 1, respectiv lungimea sirului.

Exemplul 1.

```
"abcdef"(2 TO 5) = "bcde"  
"abcdef"( TO 5) = "abcdef"(1 TO 5) = "abcde"  
"abcdef"(2 TO ) = "abcdef"(2 TO 6) = "bcdef"  
"abcdef"( TO ) = "abcdef"(1 TO 6) = "abcdef"  
"abcdef"(3) = "abcdef"(3 TO 3) = "c"  
"abcdef"(5 TO 7) da mesaj de eroare deoarece sirul are numai sase caractere  
"abcdef"(8 TO 7) = ""  
"abcdef"(1 TO 0) = ""
```

Exemplul 2.

```
10 LET a$ = "Salut Ana !"  
20 FOR n = 1 TO 11  
30 PRINT a$(n TO 11), a$((12-n) TO 11)  
40 NEXT n  
50 STOP
```

Exemplul 3.

```
10 LET c$ = "Acesta este un calculator HC"  
20 LET c$(13 TO 25) = "hc"  
30 PRINT c$
```

Dupa executia programului pe ecran va apare mesajul:

```
Acesta este hc HC
```

Daca intr-o atribuire sirul din dreapta contine mai putine caractere decit sint specificate in subsirul din stanga, atunci diferenta de lungime va fi completata cu blankuri. O astfel de asignare se numeste "procusteana".

3.3 Tablouri

Cuprins: Tablouri de numere si siruri DIM

In limbajul BASIC al calculatoarelor HC se pot defini variabile de tip tablou cu oricite dimensiuni. Elementele tabloului pot fi numere reale, caz in care numele variabilei este reprezentat printr-o singura litera, sau de tip sir de caractere, numele variabilei fiind format dintr-o litera urmata de \$. Inainte de a utiliza un tablou, trebuie rezervat spatiu in calculator pentru el; aceasta se realizeaza utilizind instructiunea **DIM**, a carei forma este:

```
DIM m(n1,n2,...,nk)
```

unde:

1. m - este numele unei variabile de tip tablou.
2. n1,n2,...,nk - sint numerele maxime de componente corespunzatoare fiecarei dimensiuni a tabloului.

Printr-o comanda **DIM** poate fi definita numai o singura variabila de tip tablou. Aceasta instructiune are urmatorul efect:

1. rezerva spatiul necesar tabloului definit.
2. initializeaza elementele tabloului cu 0.
3. sterge orice tablou care are acelasi nume cu variabila definita prin instructiunea curenta.

Se mentioneaza ca pot coexista un tablou si o variabila simpla cu acelasi nume, fara sa apara confuzii.

Sirurile dintr-un tablou difera de sirurile simple prin aceea ca au lungime fixa si

assignarea lor este procusteana. Un alt mod de interpretare al unui tablou de siruri de caractere este ca tablou de caractere simple cu numarul dimensiunilor majorat cu 1 fata de cazul precedent. Un tablou de siruri si o variabila sir simpla nu pot avea acelasi nume (spre deosebire de cazul variabilelor numerice).

Pentru a defini un tablou a\$ de 5 siruri, trebuie stabilita mai intii lungimea sirului - spre exemplu 10 caractere.

Linia:

```
DIM a$(5,10)
```

defineste $5 \times 10 = 50$ caractere, dar fiecare rind poate fi interpretat ca un sir.

De exemplu a\$(1) este format din:

```
a$(1,1) a$(1,2) ... a$(1,10)
```

Daca sint utilizate doua dimensiuni, se obtine un singur caracter, dar daca este omisa a doua dimensiune, atunci se obtine un sir cu lungime fixa. Astfel a\$(2,7) e al saptelea caracter in sirul a\$(2); o alta notatie a aceluiasi element este a\$(2)(7).

Ultimul indice poate avea si forma unui selector de subsir. De exemplu, daca a\$(2) = "12345667890", atunci

```
a$(2,4 TO 8) = a$(2)(4 TO 8) = "45678"
```

Se pot defini variabile de tip tablou de siruri de caractere cu o singura dimensiune; in acest caz variabila se comporta ca o variabila simpla cu exceptia faptului ca are totdeauna lungime fixa iar assignarea ei este procusteana.

Exemplu

```
DIM a$(10)
```

3.4 Initializarea variabilelor

Cuprins: READ, DATA, RESTORE

Introducerea constantelor intr-un program se face prin grupul de instructiuni READ, DATA si RESTORE. Forma generala a unei instructiuni READ este:

```
READ n1,n2,...
```

unde n1,n2,... este lista variabilelor care trebuiesc initializate, ele fiind separate prin virgula. Instructiunea READ lucreaza la fel cu instructiunea INPUT, exceptind faptul ca valorile variabilelor sint luate dintr-o instructiune DATA, nu de la terminal.

Fiecare instructiune DATA este o lista de expresii numerice sau de tip sir de caractere, separate prin virgula. Instructiunile DATA pot fi puse oriunde in program, ele comportandu-se ca o lista unica realizata prin concatenarea tuturor instructiunilor DATA din program (lista DATA).

Cind calculatorul citeste prima variabila cu READ, ei ii este asociata prima valoare din lista DATA, si asa mai departe. Daca se incearca citirea mai multor variabile decit numarul valorilor din lista DATA, atunci apare eroare.

Este posibil sa se faca salturi in lista DATA, utilizind instructiunea RESTORE. Forma instructiunii este:

RESTORE n

Ea face ca instructiunea READ urmatoare sa citeasca datele de la o instructiune DATA aflata la linia "n" sau dupa aceasta. Daca "n" lipseste, se ia valoarea implicita 1.

Exemplul 1.

```
10 READ a,b,c
20 PRINT a,b,c
30 DATA 10,20,30
40 STOP
```

Rezultatele programului vor fi:

```
10 20 (a=10, b=20)
30 (c=30)
```

Exemplul 2.

```
10 READ d$
20 PRINT "Data este: ",d$
30 DATA "10 martie 1992"
```

Rezultatul acestui program este:

Data este: 10 martie 1992

Exemplul 3.

```
10 READ a,b
20 PRINT a,b
30 RESTORE 10
40 READ x,y,z
50 PRINT x,y,z
60 DATA 1,2,3
70 STOP
```

Rezultatele furnizate de acest program sint:

```
1 2 (a=1, b=2)
1 2 (x=1, y=2)
3 (z=3)
```

3.5 Operatii logice

Cuprins: =, <, >, <=, >=, <>
AND, OR, NOT

Operatiile aritmetice executate de calculator sint inmultirea, impartirea, adunarea si scaderea. Operatiile de adunare si scadere au prioritate egala dar mai mica decit a inmultirii si a impartirii.

Pentru sirurile de caractere s-a definit operatia de concatenare, notata cu "+".

Exemplul 1.

```
10 LET n$ = "Ionescu"  
20 LET p$ = "Ana"  
30 LET s$ = n$ + p$  
40 PRINT s$  
50 STOP
```

Programul prezentat va determina tiparirea pe ecran a textului:

Ionescu Ana

care reprezinta valoarea variabilei s\$.

Relatiile de ordine in multimea numerelor sint relatiile de egalitate si de inegalitate apelabile folosind notatiile "=", "<", ">", "<=", ">=", "<>".

In multimea sirurilor de caractere relatia de ordine folosita este ordonarea alfabetica, relatiile folosite fiind aceleasi ca la numere.

Pentru realizarea unor expresii complexe se pot utiliza si operatiile logice "OR", "AND" si "NOT" care admit operanzi de tip boolean. De exemplu instructiunea:

```
IF a$ = "DA" AND x0 THEN PRINT x
```

tipareste valoarea numarului "x" daca sint indeplinite simultan cele 2 conditii.

Similar se pot realiza expresii cu "OR" daca se doreste identificarea situatiei in care cel putin una dintre conditii este indeplinita. Operatia "NOT" produce ca rezultat inversul valorii argumentului sau.

Operatiile "OR", "AND", "NOT" pot fi aplicate si unor argumente numerice. Functiile definite astfel sint:

1. x AND y ia valoarea
x, daca y e nenul
0, daca y=0
2. x OR y ia valoarea
1, daca y e nenul
x, daca y=0
3. NOT x ia valoarea
0, daca x e nenul
1, daca x=0

In continuare sint prezentate operatiile recunoscute de limbajul BASIC in ordinea crescatoare a prioritaticilor:

1. "OR"
2. "AND"
3. "NOT"
4. relatiile conditionale
5. "+", "-", "*", "/"
6. "<", ">", "<=", ">=", "<>"

3.6 Functii

Cuprins: ↑, PI, EXP, LN, SIN, COS, TAN, ASN, ACS, ATN, DEF, LEN, STR\$, VAL, SGN, ABS, INT, SQR, FN

Functiile definite de calculator au prioritate mai mare decit operatiile. Daca in evaluarea unei expresii este necesara o alta ordine de executie a operatiilor si functiilor decit cea determinata de prioritatile lor, atunci se folosesc paranteze.

Functiile matematice definite in BASIC sint ridicarea la putere, functia exponentiala, functia logaritmica si functiile trigonometrice.

Functia ridicare la putere "↑" are prioritate mai mare decit inmultirea si impartirea. Ea necesita 2 operanzi dintre care primul este obligatoriu pozitiv. Intr-o insiruire de ridicari la putere, ordinea evaluarii este de la stinga la dreapta, ceea ce inseamna ca :

$$2\uparrow 3\uparrow 2 = 8\uparrow 2 = 64$$

Functia EXP defineste functia exponentiala:

$$\text{EXP } x = e^x$$

unde $e = 2,71\dots$

Functia LN calculeaza logaritmul natural al argumentului.

Ea poate fi utilizata la calculul unui logaritm in orice baza folosind formula:

$$\text{LOG}_a x = \text{LN } x / \text{LN } a$$

SIN, COS, TAN, ASN, ACS, ATN sint mnemonicele functiilor sinus, cosinus, tangenta, arcsinus, arccosinus si respectiv arc tangentă.

Sistemul pune la dispozitia utilizatorului numarul "pi", ce poate fi apelat apasind tasta PI. Comanda PRINT PI tipareste valoarea numarului "pi".

Functiile descrise in continuare sint disponibile in modul de lucru extins. Actionarea simultana a tastelor CAPS SHIFT si SYMBOL SHIFT determina trecerea din modul "L" in modul "E".

Functia LEN da lungimea unui sir.

Exemplul 1.

```
PRINT LEN "majuscule"
```

va determina tiparirea numarului 9.

Functia STR\$, converteste numere in siruri. Argumentul este un numar, iar rezultatul este sirul care ar apare pe ecran daca numarul ar fi afisat cu PRINT. Se observa ca numele functiei se sfirseste cu "\$" pentru a arata ca rezultatul ei este un sir.

Exemplul 2.

```
LET a$ = STR$ 1e2
```

Instructiunea de mai sus are acelasi efect cu:

LET a\$ = "100"

Comanda

PRINT LEN STR\$ 100.000

produce raspunsul 3, deoarece STR\$ 100.000 = "100".

Funcția VAL convertește șiruri de caractere în numere.

VAL "3.5" = 3.5

Dacă se aplică funcțiile STR\$ și VAL asupra unui număr, totdeauna se va obține numărul inițial, pe când dacă se aplică VAL urmat de STR\$ asupra unui șir de caractere nu se obține totdeauna șirul inițial. Evaluarea funcției VAL se face în 2 pași:

1. argumentul este evaluat ca șir.
2. ghilimelele sunt îndepărtate și caracterele rămase sunt evaluate ca numere.

Exemplul 3.

VAL "2*3" = 6

VAL ("2" + "*" + "3") = 6

Altă funcție similară lui VAL dar mai puțin utilizată este VAL\$. Și această funcție se evaluează tot în 2 pași; primul pas este la fel cu al funcției VAL, dar după înlăturarea ghilimelelor caracterele sunt evaluate ca alt șir.

VAL\$ ""fructe"" = "fructe"

Funcția SGN aplicată asupra variabilei x are următoarea definiție:

1. 1, dacă $x > 0$
2. 0, dacă $x = 0$
3. -1, dacă $x < 0$

Funcția ABS produce valoarea absolută a numărului pe care-l are ca argument.

ABS -3.2 = ABS 3.2 = 3.2

Funcția INT furnizează partea întreagă a argumentului său.

INT 3.9 = 3

INT -3.9 = -4

Funcția SQR calculează rădăcina pătrată a argumentului său care este un număr pozitiv.

SQR 0.25 = 0.5

SQR -4 = = generează mesaj de eroare

Sistemul permite definirea de funcții utilizator. Numele posibile pentru acestea sunt FN urmat de o literă (dacă rezultatul e un număr), sau FN urmat de o literă și \$ (dacă rezultatul

e un șir). Obligativ argumentul trebuie să fie inclus în paranteze. Definirea funcțiilor utilizator se face cu funcția predefinită DEF. Definirea funcției de ridicare la pătrat se poate face astfel:

DEF FN s(x) = x*x

Rotunjirea unui număr real la cel mai apropiat întreg poate fi făcută prin aplicarea funcției INT asupra argumentului marit cu 0.5:

20 DEF FN r(x) = INT(X + 0.5)

Exemplul 4.

10 LET x = 0: LET y = 0: LET a = 10

20 DEF FN p(x,y) = a + x*y

30 DEF FN q() = a + x*y

40 PRINT FN p(2,3), FN q()

Când este evaluată FN p(2,3), "a" are valoarea 10, deoarece e variabilă liberă, x are valoarea 2 deoarece este primul argument și y are valoarea 3 deoarece este al doilea argument. Rezultatul este $10 + 2*3 = 16$.

Când este evaluată funcția fără argumente FN q, a, x și y sunt variabile libere și au valorile: 10, 0 respectiv 0. Raspunsul în acest caz este $10 + 0*0 = 10$.

Schimbind linia 20 cu

20 DEF FN p(x,y) = FN q()

de această dată FN p(2,3) va avea valoarea 10.

O funcție poate avea până la 26 argumente numerice și în același timp până la 26 argumente de tip șir de caractere.

3.7 Decizii

Cuprins: IF, THEN, STOP

Instrucțiunea care realizează luarea deciziilor este de forma:

n IF condiție THEN comenzi

unde

1. "n" este numărul liniei.
2. "comenzi" este o secvență de instrucțiuni care trebuie să fie executată în cazul în care "condiția" este adevărată.
3. "condiție" este o relație operațională care în urma evaluării poate fi adevărată sau falsă. Dacă condiția este adevărată, atunci se execută secvența de instrucțiuni scrisă după THEN. Altfel, programul execută instrucțiunile de pe linia următoare.

Cele mai simple condiții compară două numere sau două șiruri de caractere. Ele pot testa dacă două numere sunt egale sau dacă unul este mai mare decât celălalt. Se poate testa

si egalitatea a doua siruri de caractere, sau daca in ordinea alfabetica unul apare inaintea celuilalt.

Exemplu:

```
10 REM Ghiciti numarul
20 INPUT a: CLS
30 INPUT "Ghiciti numarul", b
40 IF b = a THEN PRINT "Rezultat corect": STOP
50 IF b > a THEN PRINT "Prea mic! Mai incearca o data!"
60 IF b < a THEN PRINT "Prea mare! Mai incearca o data!"
70 GO TO 30
```

In acest program linia 40 compara variabilele a si b. Daca sint egale, programul este oprit cu comanda STOP. In partea de jos a ecranului apare mesajul

9 STOP statement, 40:3

care arata ca oprirea programului este cauzata de a treia instructiune din linia 40.

Linia 50 determina daca b este mai mic decit a, iar linia 60 opusul, adica daca b este mai mare decit a. Instructiunea CLS din linia 20 sterge ecranul si impiedica adversarul de joc sa vada ce numar s-a introdus.

3.8 Iteratii

Cuprins: FOR, NEXT, TO, STEP

In BASIC instructiunea de ciclare este FOR - NEXT. Forma generala a instructiunii FOR este:

```
FOR v = vi TO vf STEP p
  corp ciclu
NEXT v
```

unde

1. "v" este o variabila contor specifica ciclului FOR - NEXT; ea trebuie sa aiba numele format dintr-o singura litera.
2. "vi" este valoarea cu care este initializat contorul ciclului.
3. "vf" este valoarea maxima la care poate ajunge "v"; deci "v" "vf" (s-a presupus ca "p" > 0).
4. "p" este marimea pasului; el reprezinta diferenta intre doua valori succesive ale contorului.
5. "corp ciclu" este secventa de instructiuni ce se repeta. "vi", "vf" si "p" pot fi exprimate prin constante, variabile sau expresii de tip real.

In cazul in care "p" este negativ, regula de raminere in ciclu este "v" = "vf".

Doua cicluri FOR - NEXT pot fi imbricate sau complet separate. Este gresita suprapunerea partiala a doua cicluri. De asemenea trebuie evitat saltul din exterior in interiorul unei bucle FOR - NEXT deoarece contorul nu poate fi initializat decit printr-o instructiune FOR. Pentru a fi siguri ca nu se fac salturi in interiorul unui ciclu se pot scrie toate instructiunile ciclului pe o singura linie (daca spatiul permite).

Exemplul 1.

```
10 FOR n = 10 TO 1 STEP -1
20 PRINT n
30 NEXT n
```

Exemplul 2.

```
50 FOR m = 0 TO 6
60 FOR n = 0 TO m STEP 1/2
70 PRINT m;" ";n;" ";
80 NEXT n
90 PRINT
100 NEXT m
```

Exemplul 3.

```
100 FOR m = 0 TO 10: PRINT m: NEXT m
```

Exemplul 4.

```
FOR n = 0 TO 1 STEP 0: INPUT a: PRINT a: NEXT n
```

Aceasta comanda determina repetarea la infinit a instructiunii INPUT in modul de lucru imediat (deci nu prin program). Daca apare o eroare, comanda INPUT se pierde si deci pentru continuarea citirii trebuie rescrisa intreaga linie.

3.9 Subrutine

Cuprins: GOSUB, RETURN

Utilizarea subrutinelor este posibila prin utilizarea instructiunii GO SUB (go to subroutine-apel de subrutina) si RETURN (revenire din subrutina). Instructiunea GOSUB are forma:

```
GO SUB n
```

unde "n" este numarul primei linii din subrutina. Ea este asemanatoare instructiunii GO TO n, cu exceptia faptului ca in cazul instructiunii GO SUB este memorata adresa instructiunii, astfel incit dupa executarea subrutinei programul continua cu instructiunea urmatoare saltului la subrutina. Aceasta se realizeaza memorind numarul liniei si numarul instructiunii din linie (care impreuna formeaza adresa de revenire) intr-o stiva.

Instructiunea RETURN ia adresa din virful stivei GO SUB si merge la instructiunea care ii urmeaza.

In BASIC subrutinele sint recursive.

Exemplu:

```
10 INPUT a: CLS
20 INPUT "ghiciti numarul!", b
30 IF a = b THEN PRINT "corect !!!": STOP
40 IF a < b THEN GO SUB 90
```

```

50 IF ab THEN GO SUB 90
60 GO TO 20
90 PRINT "Mai incearca o data !"
100 RETURN

```

Instructiunea **GO TO** este foarte importanta deoarece sistemul semnaleaza eroarea daca, in executie, intilneste un **RETURN** care nu a fost precedat de un **GO SUB**.

3.10 Generarea numerelor aleatoare

Cuprins: *RND, RANDOMIZE*

Generarea numerelor aleatoare se face cu functia predefinita **RND**. Ea nu este o functie complet aleatoare ci o functie periodica cu perioada suficient de mare (65535), astfel incit efectul de periodicitate poate fi neglijat. In cadrul unei perioade, numerele generate sint complet aleatoare. In anumite privinte, **RND** se comporta ca o functie fara argumente: efectueaza calcule si produce un rezultat. De fiecare data cind e utilizata, rezultatul sau este un numar aleator nou, cuprins intre 0 si 1 (uneori poate lua valoarea 0, dar niciodata 1). Daca se doreste ca numerele aleatoare sa fie intr-un anumit domeniu de valori se poate proceda ca in exemplele urmatoare:

```

5*RND      genereaza numere intre 0 si 5;
1.3+0.7*RND  produce numere intre 1.3 si 2;
1+INT(RND*6) furnizeaza numere aleatoare intregi intre 1 si 6.

```

Exemplu

```

10 REM Program de simulare a aruncarii zarurilor
20 CLS
30 FOR n = 1 TO 2
40 PRINT 1+INT(RND*6);" ";
50 NEXT n
60 INPUT a$: GO TO 20

```

Linia 60 face sa fie generata o pereche de numere aleatoare dupa fiecare apasare a tastei **CR**.

Functia **RANDOMIZE** e utilizata pentru a face ca **RND** sa porneasca dintr-un punct definit al secventei de numere; argumentul sau este un numar intru 1 si 65535 care reprezinta numarul de ordine al viitorului apel al functiei **RND**. Efectul instructiunii **RANDOMIZE** se poate vedea in programul urmat.

```

10 RANDOMIZE 1
20 FOR n = 1 to 5 :PRINT RND :NEXT n
30 PRINT:GO TO 10

```

Dupa fiecare executie a instructiunii **RANDOMIZE 1**, **RND** va furniza o secventa de 5 numere ce incepe cu 0.0022735596, care este primul numar generat de functia **RND** (are numarul de ordine 1). **RANDOMIZE** poate fi folosit la testarea programelor ce contin functia **RND**, deoarece secventa numerelor aleatoare generate este mereu aceeaasi.

RANDOMIZE 0, ca si **RANDOMIZE**, are efect diferit de **RANDOMIZE** urmat de un numar. Aceasta instructiune utilizeaza timpul trecut de la punerea in functiune a calculatorului. Programul:

```

10 RANDOMIZE
20 PRINT RND: GO TO 10

```

determina tiparirea aceluiasi numar. Deoarece timpul de lucru al calculatorului a crescut cu aceeasi cantitate la fiecare executie a lui **RANDOMIZE**, urmatorul **RND** furnizeaza aproximativ acelasi rezultat.

Pentru a se obtine o secventa aleatoare se inlocuieste **GO TO 10** cu **GO TO 20**.

Exemplu

Programul determina frecventa de aparitie a "capului" si a "pajurei" la aruncarea unei monezi.

```

10 LET cap = 0:LET pajura = 0
20 LET moneda = INT(RND*2)
30 IF moneda = 0 THEN LET cap = cap + 1
40 IF moneda = 1 THEN LET pajura = pajura + 1
50 PRINT cap; ";";pajura
60 IF pajura > 0 THEN PRINT cap/pajura;
70 PRINT: GO TO 20

```

Daca timpul de rulare este suficient de mare, raportul cap/pajura devine aproximativ 1, deoarece numerele aleatoare generate sint uniform repartizate in intervalul 0,1.

3.11 Setul de caractere

Cuprins: *CODE, CHR\$, POKE, PEEK, USR, BIN*

Alfabetul utilizat de **HC** cuprinde 256 caractere si fiecare are un cod intru 0 si 255. Caracterele pot fi simboluri simple sau cuvinte cheie ca **PRINT**, **STOP**, etc.

Pentru conversia intre coduri si caractere, limbajul poseda doua functii: **CODE** si **CHR\$**. **CODE** se aplica unui sir si intoarce codul primului caracter al sirului (sau 0 daca sirul e vid).

CHR\$ se aplica unui numar si produce caracterul ce are acel cod.

Setul de caractere este format din: caracterele ASCII, cuvinte cheie, caractere grafice definite de utilizator.

Un caracter se deseneaza pe o retea de 8*8 puncte, fiecarui punct corespunzandu-i un bit in memorie. Pentru programarea unui caracter definit de utilizator este necesara descrierea starii fiecarui punct al matricii prin care se reprezinta caracterul respectiv:

- 0 corespunde unui punct alb
- 1 corespunde unui punct negru

Pentru definirea caracterului se folosesc 8 instructiuni **BIN**. O instructiune **BIN** descrie o linie a caracterului, argumentul sau fiind format din 8 cifre binare.

Cele 8 numere sint memorate in 8 octeti care corespund aceluiasi caracter.

Instructiunea **USR** converteste un argument de tip sir in adresa din memorie a primului octet al caracterului definit de utilizator corespunzator argumentului. Argumentul trebuie sa fie un singur caracter; el poate fi graficul definit de utilizator sau litera corespunzatoare (majuscula sau minuscula).

POKE memoreaza un numar direct intr-o locatie de memorie, fara sa faca apel la

mecanismele utilizate in mod obisnuit in BASIC. Opusul lui POKE este PEEK, care ne permite sa vizualizam continutul unei locatii de memorie, fara a-l modifica.

Pentru a defini caracterul grafic pi (care sa apara pe ecran la apasarea tastei P in mod grafic) se utilizeaza urmatoarea secventa de program:

```
10 FOR n=0 TO 7
20 INPUT acum: POKE USR "p" + n, acum
30 NEXT n
```

Datele introduse vor fi (in ordinea prezentata):

```
BIN 00000000
BIN 00000000
BIN 00000010
BIN 00111100
BIN 01010100
BIN 00010100
BIN 00010100
BIN 00000000
```

In cele ce urmeaza se prezinta modul de obtinere a cuvintelor cheie. Caracterele 0,...,31 sint caractere de control al modului de lucru. De exemplu CHR\$6 realizeaza tabularea pe orizontala (efect similar unei virgule intr-o instructiune PRINT).

```
PRINT 1; CHR$ 6; 2
```

are acelasi efect cu:

```
PRINT 1,2
```

si cu:

```
LET a$ = "1" + CHR$6 + "2"
PRINT a$
```

CHR\$8 determina mutarea cursorului inapoi cu o pozitie.

Exemplu:

```
PRINT "1234"; CHR$8; "5"
```

tipareste:

```
1235
```

CHR\$13 muta cursorul la inceputul liniei urmatoare.

Utilizand codurile pentru caractere putem extinde conceptul de ordine alfabetica pentru a acoperi siruri ce contin orice caractere, nu numai litere, folosind in locul alfabetului uzual de 26 litere, alfabetul extins de 256 caractere (la codificarea caracterelor s-a avut in vedere ca ordinea crescatoare a codurilor atasate literelor sa coincida cu ordinea alfabetica).

Este prezentata mai departe o regula de gasire a ordinii in care se afla doua siruri. Mai

intii se compara primele caractere. Daca sint diferite, unul dintre ele are codul mai mic decit celalalt si, deci, se poate decide care este ordinea alfabetica a sirurilor. Daca aceste coduri sint egale, se compara urmatoarele caractere.

Exemplu

```
5 LET b = BIN 01111100:LET c = BIN 00111000:LET d = BIN 00010000
10 FOR n = 1 TO 6: READ p$: REM 6 piese
20 FOR f = 0 TO 7: REM citeste piesele in octeti
30 READ a: POKE USR p$ + f,a
40 NEXT f
50 NEXT n
100 REM bishop
110 DATA "b", 0, 0, BIN 001001000, BIN 01000100
120 DATA BIN 01101100, c, b, 0
130 REM king
140 DATA "k", 0, d, c, d
150 DATA c, BIN 010001000, c, 0
160 REM rook
170 DATA "r", 0, BIN 01010100, b, c
180 DATA c,b,b,0
190 REM queen
200 DATA "q",0,BIN 01010100, BIN 00101000, d
210 DATA BIN 01101100, b, b, 0
220 REM pawn
230 DATA "p", b, 0, d, c
240 DATA c, d, b, 0
250 REM knight
260 DATA "n", 0, d, c, BIN 01111000
270 DATA BIN 00011000, c, b, 0
```

3.12 GRAFICE

Cuprins: PLOT, DRAW, CIRCLE, POINT

In acest capitol se prezinta trasarea descnelor cu HC-ul. Partea utilizabila a ecranului are 22 de linii si 32 de coloane ($22 \times 32 = 704$ pozitii de caractere). Fiecare pozitie de caracter e un patrat format din 8×8 puncte. Punctele se numesc pixeli (picture elements). Un pixel se specifica prin coordonatele sale. Coordonata "x" arata distanta fata de extrema stinga, iar coordonata "y" reprezinta distanta fata de baza ecranului. Coordonatele se scriu de obicei ca o pereche de numere, in paranteze. Astfel (0,0), (255,0), (0,175), (255,175) sint extremele stinga jos, dreapta jos, stinga sus, dreapta sus.

Instructiunea

```
PLOT x,y
```

deseneaza punctul de coordonate x,y.

Programul:

```
10 PLOT INT (RND *256), INT(RND *175):INPUTa$:GO TO 10
```

scrie aleator un punct pe ecran de fiecare data cind se actioneaza CR. Programul urmator traseaza graficul functiei SIN pentru valori intre 0 si $2 \times \pi$.

```

10 FOR n=0 TO 255
20 PLOT n,88+80*SIN(n/128*pi)
30 NEXT n

```

Calculatorul deseneaza linii drepte, cercuri si portiuni de cerc utilizand instructiunile **DRAW** si **CIRCLE**. Cu

DRAW x,y

se poate trasa o linie dreapta. Linia incepe din punctul in care se afla cursorul ultimei instructiuni **PLOT**, **DRAW**, sau **CIRCLE**. Comenzile **RUN**, **CLEAR**, **CLS** si **NEW** il reseteaza, aducindu-l pe pozitia (0,0).

DRAW determina lungimea si directia liniei. De remarcat ca argumentele unei instructiuni **DRAW** pot fi si negative.

```

PLOT 0,100: DRAW 80,-35
PLOT 90,150: DRAW 80,-35

```

Calculatorul HC are facilitati pentru a desena in culori. Urmatorul program demonstreaza acest lucru:

```

10 BORDER 0: PAPER 0: INK 7: CLS: REM tot ecranul este negru
20 LET x1=0: LET y1=0: REM inceputul liniei
30 LET c=1: REM prima culoare cu care se deseneaza este albastru
40 LET x2=INT(RND*255):LET y2=INT(RND*176): REM capatul liniei este
aleator
50 DRAW INK c; x2-x1,y2-y1
60 LET x1=x2: let y1=y2: REM urmatoarea linie incepe de unde s- a terminat
precedenta
70 LET c=c+1: IF c=8 THEN LET c=1: REM alta culoare
80 GO TO 40

```

Comenzile **PAPER**, **INK**, **FLASH**, **BRIGHT**, **INVERSE**, **OVER** pot apare in instructiuni **PLOT** sau **DRAW** in acelasi fel in care apar in **PRINT** si **INPUT**.

Comanda **DRAW** permite si trasarea de portiuni de cercuri. Forma generala este:

DRAW x,y,a

unde **x,y** semnifica punctul final al liniei iar **a** este numarul de radiani corespunzator circumferintei. Cind **a** este pozitiv portiunea de cerc se traseaza in sens antiorar in timp ce, pentru **a** negativ se deseneaza in sens orar. Pentru **a=pi** se traseaza un semicerc, indiferent de valorile luate de **x** si **y** (raza este functie de punctul initial si de cel final):

```

10 PLOT 100,100: DRAW 50,50,pi

```

Trasarea cercurilor se face cu o comanda **CIRCLE** a carei forma este:

CIRCLE x,y,r

unde **r** este raza cercului iar **(x,y)** sint coordonatele centrului cercului. Ca si instructiunea **PLOT** si **DRAW**, si **CIRCLE** admite comenzi de modificare a culorii.

Functia **POINT** arata daca un pixel are asociata culoarea **INK** sau culoarea **PAPER**. Ea are doua argumente numerice care reprezinta coordonatele pixel-ului care trebuie sa fie inchis intre paranteze. Rezultatul este:

1. 0 - daca punctul are culoarea fundalului (paper).
2. 1 - daca are culoarea **INK**.

CLS: PRINT POINT (0,0): PLOT 0,0: PRINT POINT(0,0)

Se scrie

PAPER 7: INK 0

Intr-o instructiune **PLOT x,y**, **REVERSE** si **OVER** afecteaza doar pixel-ul desemnat, nu si restul pozitiiilor din caracter. Deoarece aceste comenzi sint in mod normal dezactivate (0), pentru a le activa (1), trebuiesc incluse intr-o comanda **PLOT**. Se poate face ca punctul (x,y) sa ia culoarea "ink" prin

PLOT x,y;

PLOT INVERSE 1;

face ca pixel-ul (x,y) sa ia culoarea fundalului;

PLOT OVER 1; x,y

inverseaza culoarea pixel-ului specificat.

PLOT INVERSE 1; OVER 1; x,y

lasa pixel-ul nemodificat dar schimba pozitia de tiparire.

Alt exemplu de utilizare al instructiunii **OVER** este urmatorul:

-se umple ecranul scriind negru pe alb si apoi se tasteaza:

```

PLOT 0,0: DRAW OVER 1,255,175

```

-se traseaza astfel o linie (cu intreruperi acolo unde traverseaza caracterele tiparite pe ecran).

-reexecutind comanda, linia trasata anterior o sa dispara.

Avantajul instructiunii **OVER** este ca permite sa se deseneze si apoi sa se stearga desenele fara a afecta ce se afla anterior pe ecran.

Utilizind programul

```

PLOT 0,0: DRAW 255,175
PLOT 0,0: DRAW INVERSE 1; 255,175

```

se constata ca aceasta comanda sterge si partile din caracterele tiparite anterior.

Daca se scrie o linie cu:

```

PLOT 0,0: DRAW OVER 1; 250,175

```

se constata ca ea nu va putea fi stearsa cu:

```
DRAW OVER 1;-250,-175
```

deoarece parcurgerea dreptei intr-un sens si in celalalt nu se face exact prin aceleasi puncte. O linie se sterge pe aceeasi directie si in acelasi sens in care a fost trasata.

Pentru a extinde gama de culori se amesteca doua culori de baza pe un singur patrat, folosind un caracter grafic definit de utilizator. Programul urmatoare defineste un caracter grafic echivalent unei table de sah.

```
1000 FOR n=0 TO 6 STEP 2
1010 POKE USR "a"+n, BIN 01010101: POKE USR "a"+n+1, BIN 10101010
1020 NEXT n
```

3.13 Instructiuni de intrare-iesire

Cuprins: *PRINT, INPUT*

Utilizarea separatorilor ;, ,, *TAB, AT, LINE, CLS*

Expresiile folosite pentru a tipari valori cu instructiunea **PRINT** sint numite elementele instructiunii si sint separate intre ele cu virgula sau punct si virgula (separatori). Un element al instructiunii **PRINT** poate lipsi si in acest caz pot apare 2 virgule, una dupa alta.

Exista 2 elemente ale instructiunii **PRINT** care servesc la pozitionarea cursorului in vederea tiparirii. Acestea sint **AT** si **TAB**.

AT linie, coloana

deplaseaza cursorul (locul unde va fi tiparit urmatoarul element) la linia si la coloana specificate. Linile sint numerotate de la 0 la 21 (de sus in jos) si coloanele de la 0 la 31 (de la stinga la dreapta).

Exemplu

```
PRINT AT 11,16;"*"
```

imprima un asterisc in centrul ecranului. Instructiunea

TAB coloana

deplaseaza cursorul in coloana specificata. **TAB** determina deplasarea pe aceeasi linie pe care se gaseste cursorul, exceptind cazul cind pozitia de tiparire specificata se afla inaintea pozitiei de tiparire actuala; in aceasta situatie se face o deplasare la linia urmatoare.

Obs. : calculatorul considera coloanele din instructiunea **TAB** "modulo 32" (adica **TAB 33** este echivalent cu **TAB 1**).

Exemplul de mai jos arata cum se poate tipari inceputul paginii 1 a unei carti:

```
PRINT TAB 30;1;TAB 12; "Index"; AT 3,1;
"Capitol"; TAB 24; "Pagina"
```

Un exemplu din care rezulta reducerea modulo 32 a numarului din instructiunea **TAB** este urmatoarul:

```
10 FOR n=0 TO 20
20 PRINT TAB 8*n;n;
30 NEXT n
```

De retinut urmatoarele observatii:

1. Elementele de tiparire care urmeaza instructiunilor **TAB** sau **AT** sint de obicei terminate cu ";". Daca s-ar folosi "," sau nimic, cursorul, dupa ce este pozitionat, se deplaseaza.

2. Linile 22 si 23 ale ecranului nu pot fi folosite pentru tiparire. Ele sint rezervate pentru comenzi, pentru citirea datelor, mesaje, etc.

3. Tiparind cu **AT** intr-o pozitie deja scrisa, ultima tiparire o anuleaza pe precedenta.

CLS sterge tot ecranul, functie care mai este realizata si de comenzile **CLEAR** si **RUN** (care mai executa si alte functii). Cind calculatorul, in timp ce tipareste, ajunge la ultima linie a ecranului, executa "scrolling" anulind prima linie.

Exemplu:

```
CLS: FOR n=1 TO 22: PRINT n: NEXT n
```

si apoi,

```
PRINT 99
```

de mai multe ori.

In timpul tiparirii, dupa ce calculatorul a umplut complet ecranul, se opreste scriind in partea de jos:

```
scroll ?
```

Se raspunde cu "y" sau "n".

Instructiunea **INPUT**

O linie de **INPUT** este compusa dintr-o serie de elemente si de separatori care au aceeasi functie ca intr-o linie de **PRINT**. **INPUT** considera orice element care incepe cu o litera ca pe o variabila asignabila (careia urmeaza sa i se introduca valoarea de la tastatura). Instructiunea **INPUT** poate tipari si mesaje; pentru a tipari un sir de caractere este suficienta introducerea acestuia intre ghilimele. Daca contine si valori de variabile, mesajul se inchide intre paranteze.

Daca se doreste citirea unei variabile de tip sir de caractere, a\$, pe ecran apare caracterul ghilimele. Daca aceasta variabila trebuie sa ia valoarea unei alte variabile de tip sir definita in program, b\$, aceasta se face prin stergerea ghilimelelor si introducerea numelui variabilei (b\$).

Toate elementele instructiunii **PRINT** care nu sint supuse acestor reguli pot fi elemente ale instructiunii **INPUT**.

Exemplu

```
LET virsta mea = INT( RND*100): INPUT ("Eu am";  
virsta mea; "ani. "); "citi ani ai?"; virsta ta
```

Variabila "virsta mea" este continuta intre paranteze, deci valoarea sa se tipareste, in timp ce variabila "virsta ta" nu este intre paranteze, si deci valoarea sa se citeste de la tastatura.

O alta modalitate de citire a variabilelor sir consta in scrierea cuvintului cheie **LINE** dupa **INPUT** si inaintea variabilei sir de citit:

```
INPUT LINE a$
```

In acest caz calculatorul nu va tipari ghilimelele, care in mod normal sint tiparite cind se asteapta introducerea unei variabile sir, chiar daca se comporta ca si cum ar fi fost. Astfel, scriind carte ca variabila de intrare, a\$ va lua valoarea "carte". Deoarece ghilimelele nu sint tiparite, nu este posibila introducerea altui sir. De notat ca **LINE** nu poate fi folosit pentru variabile numerice.

Caracterele de control **CHR\$22** si **CHR\$23** functioneaza aproape similar lui **AT** si **TAB**. Caracterul de control pentru **AT** este **CHR\$22**. Primul caracter care il urmeaza specifica numarul de linie, iar al doilea numarul coloanei, astfel ca:

```
PRINT CHR$22 + CHR$1 + CHR$c;
```

este analog lui

```
PRINT AT 1,c;
```

CHR\$1 si **CHR\$c** (c=13) in mod normal au alta semnificatie, pe care insa si-o pierd cind urmeaza dupa **CHR\$22**.

Caracterul de control echivalent lui **TAB** este **CHR\$23** si cele doua caractere care-l urmeaza sint folosite pentru a indica un numar cuprins intre 0 si 65535 care specifica numarul de **TAB** ca si argumentul unei instructiuni **TAB**.

```
PRINT CHR$23 + CHR$a + CHR$b
```

este echivalent lui

```
PRINT TAB a + 256*b
```

Daca nu se doreste afisarea mesajului "scroll ?" la sfirsitul fiecarui ecran, se poate folosi:

```
POKE 23692,255
```

din cind in cind. Dupa aceasta linie calculatorul inhiba mesajul "scroll ?" pentru urmatoarele 255 linii.

3.14 CULORI

Cuprins: **PAPER, INK, FLASH, INVERSE, OVER, BORDER, ATTR**

Calculatorul HC are facilitati color. El foloseste 8 culori (numerotate de la 0 la 7). Lista culorilor in ordinea in care sint pe tastele numerice este urmatoarea:

- 0 - negru
- 1 - albastru
- 2 - rosu
- 3 - purpuriu (magenta)
- 4 - verde
- 5 - albastru deschis
- 6 - galben
- 7 - alb

Intr-un televizor alb-negru aceste numere corespund unor tonuri de gri ordonate de la inchis spre deschis.

Orice caracter are asociate 2 culori: culoarea caracterului propriu-zis si culoarea fondului (vezi subcapitolul Setul de caractere). La pornirea calculatorului, sistemul lucreaza in alb-negru, cu caractere negre pe fond alb. Tiparirea poate fi facuta normal, dar exista si posibilitatea sa apara pe ecran pilpiind (flash). Pilpiirea se obtine inversind continuu culoarea caracterului cu culoarea fondului. Deoarece atributele de culoare si pilpiire sint asociate caracterelor (deci matricilor de 64 puncte), nu este posibil ca intr-un caracter sa fie mai mult de doua culori. Valorile acestor atribute pot fi modificate cu instructiunile **INK, PAPER** si **FLASH**. Forma acestor instructiuni este:

```
PAPER n  
INK n  
FLASH m
```

unde

- 1. n este un numar cuprins intre 0 si 7.
- 2. m este un numar binar (0 pentru inactiv si 1 pentru activ).

Pentru ilustrarea modului de folosire al instructiunilor prezentate se propune programul:

```
20 FOR n=1 TO 10  
30 FOR c=0 TO 7  
40 PAPER c: PRINT " ";REM spatii colorate  
50 NEXT c: NEXT n  
60 PAPER 7  
70 FOR c=0 TO 3  
80 INK c: PRINT c; "  
90 NEXT c: PAPER 0  
100 FOR c=4 TO 7  
110 INK c: PRINT c; "  
120 NEXT c  
130 PAPER 7: INK 0
```

In afara de aceste valori de argumente a caror semnificatie a fost deja prezentata, mai

pot fi folosite valorile 8 si 9. 8 poate fi folosit ca argument pentru toate cele 4 comenzi si semnifica transparenta, fapt ce nu altereaza atributele pozitiei la tiparirea unui caracter. De exemplu:

PAPER 8

face ca la tiparirea unui caracter, culoarea fondului sa fie aceeași cu a caracterului tiparit anterior. 9 poate fi folosit numai cu comenzile PAPER si INK si indica contrastul. Culoarea "cernelii" sau a "hirtiei" (fundalului), in functie de comanda utilizata, este facuta sa contrasteze cu cealalta, punind alb pe o culoare inchisa (negru, albastru, rosu, magenta) si negru pe o culoare deschisa (verde, bleu, galben, alb).

```
INK 9: FOR c=0 TO 7: PAPER c: PRINT c: NEXT c
```

Rulind programul

```
INK 9: PAPER 8: PRINT AT 0,0; FOR n=1 TO 1000:  
PRINT n: NEXT n
```

dupa primul program din acest paragraf, culoarea cernelii este facuta mereu sa contrasteze cu vechea culoare pe care o avea fundalul in fiecare pozitie. Comanda

INVERSE 1

inverseaza fundalul cu cerneala pentru caracterul specificat.
Comanda

OVER 1

realizeaza supratiparirea. In mod obisnuit, cind ceva este scris intr-o pozitie de caracter, sterge complet ce era scris inainte; de data aceasta noul caracter va fi doar adaugat. Acest lucru este util in scrierea caracterelor compuse, cum ar fi literele cu accente. Trebuie utilizat in acest scop caracterul de control CHR\$8 pentru intoarcerea cu o pozitie.

Exista o alta posibilitate de a utiliza INK, PAPER, FLASH. Pot apare in PRINT urmate de ";" si fac exact acelasi lucru pe care l-ar face cind sint utilizate independent, exceptind faptul ca efectul lor este numai temporar.

Astfel daca se ruleaza:

```
PRINT PAPER 6; "x"; PRINT "y"
```

numai x va fi pe fond galben.

INK si celelalte comenzi nu afecteaza culorile partii de jos a ecranului. Aceasta foloseste culoarea marginii drept culoare a fundalului si codul 9 pentru a contrasta culoarea cernelii. Nu are posibilitatea de pilpiire si este cu luminozitate normala.

Marginea poate lua oricare din cele 8 culori (0-7) cu comanda

BORDER culoare

Se pot schimba culorile mesajului scris pe ecran cu comanda INPUT, inserind in aceasta comanda INK, PAPER, etc, ca si in cazul comenzii PRINT. Efectul lor este activ numai asupra comenzii urmatoare:

```
INPUT FLASH 1; INK 1; "text"; n
```

Comenzile pot fi schimbate utilizind caracterele de control ca si in cazul AT si TAB (vezi capitolul Instructiuni de intrare- iesire).

```
CHR$16 -- INK  
CHR$17 -- PAPER  
CHR$18 -- FLASH  
CHR$20 -- INVERSE  
CHR$21 -- OVER
```

Aceste caractere de control sint urmate de un caracter care indica culoarea prin intermediul codului sau. De exemplu:

```
PRINT CHR$16 + CHR$9; ...
```

are acelasi efect cu:

```
PRINT INK 9; ...
```

Funcția ATTR are forma:

ATTR (linie,culoana)

Rezultatul este un numar care arata atributele pentru caracterul aflat la linia si coloana precizata. Numarul este suma a patru numere, conform schemei:

1. 128 - daca pozitia pilpiie, 0 daca este stabila
2. 64 - daca pozitia este stralucitoare, 0 daca este normala
3. 8*n - n = codul fundalului
4. m - m = codul cernelii

Exemplu: Pentru o pozitie pilpiitoare, normala, cu fundal galben si cerneala albastra se obtine:

$$128 + 0 + 8*6 + 1 = 177$$

3.15 MISCAREA

Cuprins: PAUSE, INKEY\$, PEEK

Pentru a realiza o pauza in program in timpul careia nu se desfasoara nici o operatie se foloseste comanda:

```
PAUSE n
```

care opreste executia programului mentinind activ display-ul pe durata a n perioade de baleiaj ale ecranului (20 ms pentru fiecare ecran); n poate lua valoarea maxima 65535, careia ii corespunde o pauza de aproximativ 22 minute. Daca n=0 se opreste definitiv.

O pauza obtinuta in acest mod poate fi scurtata apasind orice tasta (cu exceptia lui SPACE si CAPS SHIFT care produce BREAK).

Programul urmat deseneaza cadranul unui ceas pe care se misca secundarul:

```

10 REM Mai intii este desenat cadranul.
20 FOR n=1 TO 12
30 PRINT AT 10-10*COS( n/PI), 16 + 10*SIN( n/PI)
40 NEXT n
50 REM Se porneste ceasul.
60 FOR t=0 TO 200000 :REM t este timpul in secunde
70 LET a=t/30*PI: REM a este unghiul secundarului in radiani
80 LET sx=80*SIN( a): LET sy=80*COS( a)
200 PLOT 128,88: DRAW OVER 1; sx, sy: REM Se deseneaza secundarul.
210 PAUSE 42
220 PLOT 128,88: DRAW OVER 1; sx, sy: REM Se sterge secundarul
230 NEXT t

```

Cu linia 210 se marcheaza trecerea unei secunde; s-a folosit $n = 42$ si nu $n = 50$ deoarece calculatorul foloseste un timp pentru scrierea liniilor ciclului FOR - NEXT; linia 210 opreste calculatorul doar pentru timpul care mai ramane.

O temporizare mai precisa se poate realiza citind continutul anumitor locatii de memorie cu PEEK. Expresia urmatoare:

$$(65536 * PEEK 23674 + 256 * PEEK 23673 + PEEK 23672) / 50$$

da numarul de secunde scurse de la aprinderea calculatorului pina la 3 zile si 21 ore, dupa care se reseteaza. Programul unui ceas mai precis este dat in continuare:

```

10 REM Se deseneaza cadranul
20 FOR n=1 TO 12
30 PRINT AT 10-10*cos(n/6*pi),16 + 10*SIN(n/6*PI);n
40 NEXT n
50 DEF FNT() = INT(65536*PEEK 23674 + 256*PEEK
23673 + PEEK 23672)/50: REM Numarul de secunde de la inceput
100 REM se porneste ceasul
110 LET t1 = FNT()
120 LET a = t1/30*PI: REM a este unghiul in radiani
130 LET sx = 72* SIN a: LET sy = 72*COS a
140 PLOT 131,91: DRAW OVER 1; sx, sy: REM Se deseneaza secundarul
200 LET t = FNT()
210 IF t = t1 THEN GO TO 200
220 PLOT 131,91: DRAW OVER 1; sx, sy: REM Se sterge vechiul secundar
230 LET t1 = t: GO TO 120

```

Acest ceas se opreste temporar de cite ori se executa BEEP ori se utilizeaza imprimanta, casetofonul. Numerele PEEK 23674, PEEK 23673 si PEEK 23672 sint folosite pentru a numara in incremente de 20 ms. Fiecare variaza de la 0 la 255, dupa care se reincepe. Cel mai rapid se incrementeaza locatia 23672 (cu 1 la fiecare 20 ms); cind se trece de la 255 la 0, locatia 23673 se incrementeaza cu 1; analog pentru 23674. Presupunind ca cele 3 numere sint 0 (pentru PEEK 23674), 255 (pentru PEEK 23673) si 255 (pentru PEEK 23672), au trecut deci circa 21 minute de la pornirea calculatorului. Expresia devine:

$$(65536 * 0 + 256 * 255 + 255) / 50 = 1310.7$$

Pentru a pozitiona ceasul pe ora 10 se procedeaza astfel:

$$10 * 60 * 60 * 50 = 1800000 = 65536 * 27 + 256 * 119 + 64$$

si se memoreaza numerele 27, 119 si 64 cu:

```
POKE 23674,27: POKE 23673,119: POKE 23672,64
```

Functia INKEY\$, fara argument, da caracterul apasat pe tasta in momentul apelarii sale. Cu programul urimator calculatorul devine o masina de scris:

```

10 IF INKEY$ "" THEN GO TO 10
20 IF INKEY$ = "" THEN GO TO 20
30 PRINT INKEY$;
40 GO TO 10

```

Linia 10 asteapta sa se elibereze ultima tasta apasata; linia 20 asteapta apasarea uneia noi. Spre deosebire de INPUT, INKEY\$ nu asteapta apasarea lui CR sau a unei taste.

3.16 MEMORIA

Cuprins: CLEAR

Fiecarui octet ii este asociata o adresa care este un numar intre 0 si FFFFH. Memoria este impartita in trei zone distincte:

1. 0 - 4000H zona ROM
in aceasta zona se gaseste memoria ROM in care este inregistrat interpretorul BASIC.
2. 4000H - 7FFFH zona RAM video
in aceasta zona se gaseste memoria video cit si o parte din memoria RAM de program.
3. 8000H - FFFFH zona RAM suplimentar
aceasta zona nu este neaparat necesara. Ea este folosita pentru marirea capacitatii de memorie. Ea difera de zona video printr-un timp de acces mai mic.

ROM	RAM VIDEO	RAM SUPLIMENTAR	
0	4000H = 16384	8000H = 32768	FFFFH = 65535

Fig. 3.1

Continutul memoriei poate fi vizualizat cu functia PEEK care are ca argument o adresa. Exemplul urimator vizualizeaza primii 21 octeti din memoria ROM si adresele lor:

```

10 PRINT "Adresa"; TAB 10; "Octet"
20 FOR a=0 TO 20
30 PRINT a TAB 10; PEEK a
40 NEXT a

```

Schimbarea continutului memoriei RAM se poate face cu instructiunea POKE, care are forma:

POKE adresa, continut nou

unde "adresa" si "continut nou" sint expresii numerice.

POKE 31000, 57

determina incarcarea valorii 57 la adresa 31000. Cu

PRINT PEEK 31000

se va tipari 57. "Continut nou" trebuie sa aiba valoarea intre -255 si 255. Daca e numar negativ, se aduna 256.

De importanta pentru utilizator este organizarea memoriei RAM. Memoria este impartita in zone specifice stocarii unui anumit gen de informatie. Zonele sint suficient de mari pentru ca informatia continuta actualmente sa poata fi reorganizata daca se insereaza ceva intr-un anumit punct (de exemplu prin adaugarea unei linii de program sau a unei variabile). La inserare, spatiul necesar este creat prin mutarea in sus a tot ce se afla deasupra. Daca se sterge informatie, atunci totul este mutat in jos.

Fişier display	Atribute	Buffer imprimanta	Variabile sistem	Harta disc
16384	22528	23296	23552	23734 CHANS

Informatii de canal	80H	Program BASIC	Variabile	80H
CHANS		PROG	VARS	E-LINE

Comanda sau linia program in curs de introducere	NL	80H	Date in INPUT	NL	Spatiu de lucru temporar
E-LINE			WORKSP		STKBOT

Stiva calculator	Nefolosit	Stiva PROC	Stiva GOSUB	?	3EH	Caractere grafice definite de utilizator
STKBOT	STKEND	RAMTOP	UDG		P-RAMT	

Variabilele sistem (PROG, CHANS, VARS, ELINE, etc.) contin diferite informatii necesare pentru gestiunea interna a memoriei. Ele indica limitele pentru diverse zone de memorie. Ele nu sint variabile BASIC si deci nu pot fi recunoscute de calculator.

Fişierul display stocheaza imaginea televizorului. In loc de PEEK si POKE, pentru imaginea display-ului se pot utiliza SCREEN\$ si PRINT AT sau PLOT si POINT.

Atributele sint culorile, etc. pentru fiecare pozitie de caracter (se afla cu instructiunea ATTR). Ele sint stocate linie cu linie in ordinea dorita.

Buffer-ul imprimantei stocheaza caracterele destinate imprimantei.

Informatiile de canal sint necesare cind se lucreaza cu dispozitive de intrare-iesire. Si lucrul cu tastatura necesita aceasta zona deoarece partea de jos a ecranului functioneaza ca un port de intrare, in timp ce restul ecranului se comporta ca un port de iesire.

Orice linie de comanda are forma:

2 bytes	2 bytes	00001101
n	m	t
		e

unde:

1. n - este numarul liniei curente
2. m - este lungimea textului + CR
3. t - este textul liniei
4. e - este codul caracterului CR

Modul de memorare al variabilelor numerice este:

Nume	Exp	Mantisa
------	-----	---------

unde:

1. Nume - este un numar de octeti egal cu numarul de caractere ce formeaza identificatorul variabilei
2. Exp - este un octet ce contine exponentul numarului
3. Mantisa - este un grup de 4 octeti ce contine mantisa numarului. Bitul cel mai semnificativ al primului octet este bitul de semn.

3.17 PRODUCEREA SUNETELOR

Cuprins: BEEP

Pentru producerea sunetelor, se foloseste instructiunea:

BEEP d,i

unde:

1. d - este o expresie numerica ce indica durata in secunde a sunetului respectiv
2. i - este o expresie numerica ce reprezinta inaltimea sunetului, masurat in semitonuri relativ la DO central.

Pentru a transcrie muzica este indicat sa se scrie pe marginea fiecarui spatiu si linie a portativului inaltimea corespunzatoare, tinind cont de armura cheii.

Exemplu:

```
10 PRINT "Frere Gustav"
20 BEEP 1,0:BEEP 1,2:BEEP .5,3:BEEP .5,2:BEEP 1,0
30 BEEP 1,0:BEEP 1,2:BEEP .5,3:BEEP .5,2:BEEP 1,0
40 BEEP 1,3:BEEP 1,5:BEEP 2,7
50 BEEP 1,3:BEEP 1,5:BEEP 2,7
60 BEEP .75,7:BEEP .25,8:BEEP .5,7:BEEP .5,5:BEEP .5,3:
BEEP .5,2:BEEP 1,0
70 BEEP .75,7:BEEP .25,8:BEEP .5,7:BEEP .5,5:BEEP .5,3:
BEEP .5,2:BEEP 1,0
80 BEEP 1,0:BEEP 1,-5:BEEP 2,0
90 BEEP 1,0:BEEP 1,-5:BEEP 2,0
```

Pentru alcatuirea programului s-a procedat dupa cum urmeaza:

1. s-au adaugat mai intii deasupra si dedesubt cite o linie de referinta
2. s-au numerotat liniile si spatiile, observind ca mi bemol din armura cheii afecteaza nu numai mi de sus (coborin-du-l de la 16 la 15) cit si mi de jos (coborindu-l de la 4 la 3)

Pentru a schimba cheia partituri, trebuie sa se adune la inaltimea fiecarei note o variabila (de exemplu "Cheie") careia trebuie sa i se atribue valoarea adecvata inaintea executiei piesei.

Linia 20 a programului devine:

```
20 BEEP 1, Cheie 0:BEEP 1
```

In acest exemplu variabila "Cheie" trebuie sa aiba valoarea 0 pentru DO minor, 2 pentru RE minor, 12 pentru DO minor in octava superioara, etc.

Cu acest sistem este posibila acordarea calculatorului cu un alt instrument, folosind valori zecimale pentru variabila "Cheie". De asemenea, este posibil sa se execute piese cu viteze diferite. In exemplul dat "o patrima" a fost programata sa dureze o secunda. Daca se introduce o variabila "PATRIME" analog cu "Cheie", linia 20 devine:

```
20 BEEP patrima, cheie + 0: BEEP patrima,
cheie + 2:BEEP patrima/2, cheie + 3:BEEP patrima/2,
cheie + 2:BEEP patrima, cheie + 0
```

In acest fel este posibila executia aceluiasi program in orice cheie, cu orice acordare. Programul de mai jos:

```
FOR n=0 TO 1000: BEEP 0.5, n: NEXT n
```

va produce note din ce in ce mai acute, pina la limita posibilitatilor calculatorului, cind acesta va tipari mesajul:

B integer out of range

Tiparind n se obtine inaltimea notei celei mai acute care poate fi produsa. Procedul poate fi repetat pentru notele joase.

Sunetele din gama medie sint cele mai potrivite pentru a fi redade.

Sunetele grave se aud ca niste pacanituri. Ele pot fi prelungite pentru a deveni mai

naturale, cu comanda:

```
POKE 23609, m
```

cu m = 0, ..., 255.

3.18 UTILIZAREA CODULUI MASINA

Cuprins: *USR*

Calculatorul HC poate fi dotat cu un asamblor inregistrat pe caseta sau in EPROM. Introducerea programului scris in limbaj masina (functie executata in general de asamblor) se face in general cu specificarea adresei de inceput (cel mai bine este ca aceasta adresa sa se afle intre zona BASIC si zona caracterelor grafice definite de utilizator).

La pornirea unui calculator HC inceputul memoriei RAM, **RAMTOP** se afla la adresa 65366 (vezi fig. 3.2), dar se poate deplasa **RAMTOP** cu comanda **CLEAR 65266** obtinindu-se neutilizarea de catre sistem a 100 octeti incepind cu adresa 65267 (vezi fig. 3.3).

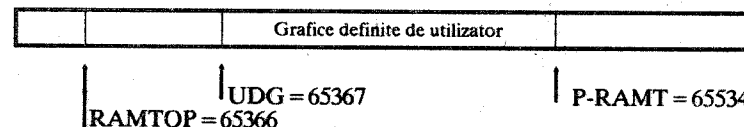


Fig. 3.2

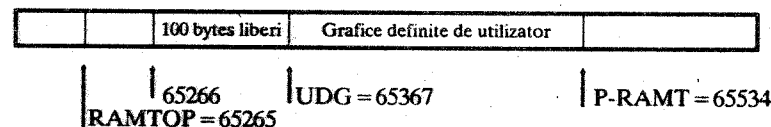


Fig. 3.3

Pentru a insera codurile obiect in memorie, se poate utiliza un program de genul:

```
10 LET a = 32500
20 READ n: POKE a, n
30 LET a = a + 1: GO TO 20
40 DATA 1,99,0,201
```

care introduce programul:

```
LD bc,99
RET
```

transpus in cod masina ca:

1, 99, 0 (pentru LD bc,99) si 201 (pentru RET).

Cind se termina cei 4 octeti specificati, apare mesajul:

E Out of DATA

Rularea programului introdus in cod masina se face cu instructiunea:

USR adresa de inceput

In exemplul de mai sus, cu:

```
PRINT USR 32500
```

se tipareste valoarea 99 din perechea de registre BC.

Adresa de revenire in BASIC se memoreaza cu instructiunea Z80 RET. In rutinele scrise in limbaj masina nu se pot folosi registrele index IY si IX.

Calculatorul HC are scoase in exterior magistralele de date, adrese si de control prin intermediul unui conector de extensie.

Un program in limbaj masina poate fi memorat ca o informatie de tip byte; deci cu:

```
SAVE "nume" CODE 32500,4
```

se memoreaza programul exemplu.

Un program in limbaj de asamblare nu se poate lansa automat, odata incarcat; el poate fi insa lansat de un program in BASIC ca in exemplul:

```
10 LOAD "" CODE 32500,4  
20 PRINT USR 32500
```

Dupa aceasta se executa:

```
SAVE "nume" LINE
```

si apoi

```
SAVE "nume" CODE 32500,4
```

Rebobinind caseta si scriind:

```
LOAD "nume"
```

se incarca si se executa programul BASIC care, la rindul sau, va apela programul in limbaj masina.

3.19 UTILIZAREA PORTURILOR INPUT, OUTPUT

Cuprins: IN, OUT

Calculatorul HC dispune de 65536 adrese de memorie de tip RAM si ROM organizate pe opt biti. El poate sa scrie cuvinte in memoria de tip RAM si poate sa citeasca cuvinte din memoriile de tip RAM si ROM. Analog sint 65536 porturi de INPUT si de OUTPUT. Aceste porturi sint folosite de procesor pentru a comunica cu exteriorul. Instructiunile

sint:

IN adresa port

care preia bitul citit de la acel port;

OUT adresa port, valoare

inscrie valoarea in portul de adresa specificat. Exista un ansamblu de adrese de intrare care citeste tastatura si conectorul de casetofon. Tastatura este impartita in 8 semipagini de 5 taste fiecare. Lista porturilor utilizate este:

IN 65278 citeste semipagina CAPS SHIFT - v

Aceste adrese sint $254 + 256 * (255 - 2^n)$ cu $n = 0, \dots, 7$

Bitii d0, ..., d4 sint asociati celor 5 taste din semipagina specificata. D6 este asociat conectorului de casetofon.

Portul de iesire cu adresa 254 controleaza difuzorul (D4), conectorul de casetofon (D3) si determina culoarea chenarului (D2, D1, D0). Portul de adresa 251 controleaza imprimanta in scriere si citire; la citire verifica daca imprimanta este gata sa imprime o noua linie si la scriere trimite linia care trebuie sa fie tiparita. Porturile de adrese 254, 247 si 239 sint folosite pentru echipamentele suplimentare (capitolul Alte periferice).

3.20 INREGISTRAREA PE CASETA

Cuprins: SAVE, VERIFY, LOAD, MERGE

Calculatorul HC are posibilitatea sa inregistreze programe de pe banda magnetica cu orice casetofon.

Conectarea calculatorului la casetofon se face cu ajutorul unui cablu special.

Pentru a memora un program pe banda, acesta trebuie sa primeasca un nume compus din maximum 10 caractere, litere si/sau cifre. Comanda este:

```
Save "nume"
```

Calculatorul raspunde cu mesajul:

Start tape then press any key.

La terminarea inregistrarii apare mesajul:

```
0 OK.
```

Pentru verificare se regleaza volumul casetofonului la nivel mediu si se conecteaza cablul; se pozitioneaza banda in punctul in care a inceput inregistrarea. Comanda este:

```
VERIFY "nume"
```

In acest fel se verifica daca programul si variabilele inregistrate pe caseta sint identice cu cele din memoria calculatorului. Daca programul a fost inregistrat si chemat corect, pe ecran apare:

Program "nume"

(In timpul cautarii programului specificat, calculatorul tipareste numele tuturor programelor pe care le intilneste) si la sfirsit mesajul:

0 OK.

In cazul unei erori de inregistrare (eroare ce apare la VERIFY) se afiseaza mesajul:

R Tape loading error

si se incarca o noua inregistrare. Incarcarea unui program memorat pe caseta se face cu comanda:

LOAD "nume"

Aceasta comanda sterge vechiul program (si variabilele sale) din calculator inainte de a incarca unul nou.

LOAD ""

fara a fi urmat de un nume de program incarca primul program gasit pe caseta.

Comanda MERGE incarca un program inregistrat pe caseta in memoria calculatorului, dar spre deosebire de comanda LOAD, anuleaza din vechiul program, inaintea inceperii transferului doar acele linii si variabile cu numere, respectiv nume deja existente in programul ce urmeaza a fi incarat. Daca instructiunile VERIFY, LOAD si MERGE sint urmate de un sir vid ca nume al fisierului cautat, calculatorul va lucra asupra primului program pe care il intilneste.

Este posibil sa se inregistreze un program pe caseta, astfel incit atunci cind este reincarat in memorie, el se lanseaza automat de la o linie specificata. Instructiunea este:

SAVE sir LINE numar

si face ca programul incarat cu LOAD (dar nu si cu MERGE) sa fie rulat automat de la linia specificata cu "numar". Daca nu este loc suficient in memorie, programul vechi si vechile variabile nu sint sterse si apare eroare:

Out of memory

In afara de programe si variabile se mai pot memora matrici si octeti. Pentru memorarea unei matrici se foloseste instructiunea:

SAVE sir DATA matrice()

unde:

1. sir - este numele de pe banda al matricii
2. matrice - specifica numele matricii care va fi memorata (numerica sau sir de caractere).

Example:

SAVE "test" DATA b()

In acest caz se cauta pe caseta o matrice cu numele "test". Cind o gaseste trimite mesajul:

Number array: test

Matricea gasita este comparata cu matricea B din memorie.

LOAD "test" DATA b()

Se cauta matricea pe banda si daca este memorie libera suficienta, anuleaza o eventuala matrice B preexistenta, si incarca noua matrice pe banda denumind-o B.

MERGE nu poate fi folosit la inregistrarea matricilor pe banda.

Memorarea tip octet este folosita pentru orice tip de data, fara vreo referire asupra utilizarii acestei date. Memorarea tip octet se face cu:

SAVE sir CODE primul octet, numarul de octeti

Acest mod de memorare copiaza o parte din memoria interna a calculatorului, asa cum este, pe banda. Transferul in sens invers se face cu:

LOAD sir CODE adresa de inceput, lungime

Cind nu se specifica lungimea sirului de octeti, calculatorul va incarca toti octetii inregistrati pe caseta.

Exemplu:

Zona de memorie in care se pastreaza imaginea pentru display incepe la adresa 16384 si are 6912 octeti. Comanda:

SAVE "imagine" CODE 16384,6912

copiază imaginea de pe ecran in momentul executiei comenzii, pe banda, cu numele imagine.

CODE 16384,6912 este folosita frecvent; de aceea a fost abreviata sub forma:

SCREEN\$

La memorarea imaginii video nu poate fi folosita comanda VERIFY.

3.21 IMPRIMANTA

Cuprins: LLIST, LPRINT, COPY

Comenzile LPRINT si LLIST sint identice cu PRINT si LIST, tiparind pe imprimanta, nu pe televizor.

Comanda COPY tipareste la imprimanta o copie a ecranului televizorului. COPY nu are efect in cazul listarilor automate (de cite ori se apasa CR).

Pentru a obtine un listing se poate folosi LIST urmat de COPY sau numai LLIST.

Imprimanta poate fi oprita in timpul unei tipariri actionind **BREAK**.

3.22 VARIABLE DE SISTEM

Octetii din memorie de la adresa 23552 la adresa 23733 sint rezervati pentru operatii specifice ale sistemului. Ei pot fi cititi pentru a afla diferite lucruri despre sistem, iar citiva din ei pot fi si modificati. Acesti octeti se numesc variabile de sistem, si au cite un nume, dar nu trebuie confundati cu variabilele utilizate de BASIC. In cazul variabilelor formate din mai multi octeti, primul va fi octetul cel mai putin semnificativ. Variabilele de sistem sint date in lista de mai jos. Abrevierile din coloana 1 au urmatoarea semnificatie:

X aceasta variabila nu poate fi modificata deoarece sistemul va functiona eronat
 N modificarea acestei variabile nu are un efect asupra functionarii normale a sistemului
 n numarul de octeti din variabila

Tip	Adresa	Nume	Continut
NB	23552	KSTATE	Folosita in citirea tastaturii
N1	23560	LAST K	Retine ultima tasta apasata
1	23561	REPDEL	Durata (in 1/50 sec) cit trebuie tinuta apasata o tasta pentru a se repeta
1	23562	REPPER	Timpul (in 1/50 sec) dupa care se repeta o tasta apasata
N2	23563	DEFADD	Adresa argumentelor functiilor definite de utilizator
N1	23565	K DATA	Al doilea octet pentru controlul culorii introdus de la tastatura
N2	23566	TVDATA	Controlul culorii, al lui AT si TAB pentru TV
X38	23568	STRMS	Adresa canalului atasat caii
2	23606	CHARS	Adresa generatorului de caractere minus 256
1	23608	RASP	DURata sunetului de eroare
1	23609	PIP	Durata sunetului la apasarea unei taste
1	23610	ERR NR	Codul de mesaj minus 1
X1	23611	FLAGS	Diferiti indicatori de control ai sistemului BASIC
X1	23612	TVFLAG	Indicatori asociati cu televizorul
X2	23613	ERR SP	Adresa elementului din stiva masinii utilizat ca adresa de intoarcere in caz de eroare
N2	23615	LIST SP	Adresa de intoarcere la listarile automate
N1	23617	MODE	Specifica cursorul (K, L, C, E, G)
2	23618	NEWPPC	Linia la care se sare
1	23620	NSPPC	Numarul instructiunii in linie la care se sare
2	23621	PPC	Numarul liniei pentru instructiunea in executie
1	23623	SUBPPC	Numarul instructiunii din linie in executie
1	23624	BORDCR	Culoarea border-ului
2	23625	E PPC	Numarul liniei curente

X2	23627	VARs	Adresa variabilelor
N2	23629	Dest	Adresa variabilelor asignate
X2	23631	CHANS	Adresa datelor de canal
X2	23633	CURCHL	Adresa informatiei curente folosite pentru intrare sau iesire
X2	23635	PROG	Adresa programului BASIC
X2	23637	NXTLIN	Adresa urmatoarei linii din program
X2	23639	DATADD	Adresa ultimului element din lista DATA
X2	23641	E LINE	Adresa comenzii introduse
2	23643	K CUR	Adresa cursorului
X2	23645	CH ADD	Adresa urmatorului caracter care urmeaza sa fie interpretat
2	23647	XPTR	Adresa caracterului dupa semnul intrebarii
X2	23649	WORKSP	Adresa spatiului de lucru temporar
X2	23651	STKBOT	Adresa inferioara a stivei calculator
X2	23653	STKEND	Adresa de inceput a spatiului liber
N1	23655	BREG	Registrul B al calculatorului
N2	23656	MEM	Adresa spatiului folosit pentru memoria calculatorului
1	23658	FLAGS2	Alti indicatori
X1	23659	DF SZ	Numarul liniilor din partea de jos a ecranului
2	23660	S TOP	Numarul liniei de sus a programului la listarea automata
2	23662	OLDPPC	Numarul liniei la care sare CONTINUE
1	23664	OSPPC	Numarul din linie la care sare CONTINUE
N1	23665	FLAGX	Diversi indicatori
N2	23666	STRLEN	Lungimea asignata sirului
N2	23668	T ADDR	Adresa urmatorului element din tabela sintaxa
2	23670	SEED	Variabila pentru RND
3	23672	FRAMES	Contorul de cadre
2	23675	UDG	Adresa primului grafic definit de utilizator
1	23677	COORDS	Coordonata x a ultimului punct plot-at
1	23678		Coordonata y a ultimului punct plot-at
1	23679	P POSN	Numarul pozitiei de scriere pe ecran
1	23680	PR CC	Octetul mai putin semnificativ al adresei pentru noua pozitie la care se imprima prin LPRINT
1	23681		Nefolosit
2	23682	ECHO E	Numarul coloanei si al liniei
2	23684	DF CC	Adresa de afisare pe ecran prin PRINT
2	23686	DFCCL	Aceslasi lucru pentru partea de jos a ecranului
X1	23688	S POSN	Numarul coloanei pentru PRINT
X1	23689		Numarul liniei pentru PRINT

X2	23690	SPONSNL	Ca S POSN pentru partea de jos a ecranului
1	23692	SCR CT	Numara defilarile de ecran
1	23693	ATTR P	Culoarea curenta
1	23694	MASK P	Folosit pentru culori transparente
N1	23695	ATTR T	Culori temporare
N1	23696	MASK T	Ca MASK P dar temporar
1	23697	PFLAG	Alti indicatori
N30	23696	MEMBOT	Arie memorie calculator
2	23728		Nefolosit
2	23730	RAMTOP	Adresa ultimului byte din aria sistemului BASIC
2	23732	P-RAMT	Adresa ultimului octet de RAM

3.23 CANALE I/O SI CAI

Cuprins: *INPUT#*, *PRINT#*, *OPEN#*, *CLOSE#*, *LIST#*, *INKEY\$#*

Pentru fiecare echipament periferic sau port I/O este asignata o linie de comunicatie numita canal. Fiecarui canal existent i se poate asocia o parte componenta software numita cale. Pentru a transmite informatii pe un canal oarecare este suficient sa transmitem informatiile pe calea asignata acestui canal.

Exemplu:

INPUT# s; 'lista variabile'

citeste date de la portul asignat caii s si le asociaza variabilelor din lista de variabile.
Similar

PRINT# s; 'lista variabile'

trimite date catre portul asociat caii s.

Asignarea unei cai la un echipament I/O se face cu instructiunea *OPEN#* s,c unde:

s este numarul caii

c este un sir care specifica canalul

Instructiunea *OPEN#* realizeaza si initializarea echipamentului I/O. Unui canal i se pot asocia mai multe cai.

In configuratia de baza calculatorul HC recunoaste trei canale:

canalul K - claviatura

canalul S - ecran

canalul P - imprimanta

Canalele S si P sint canale pe care se poate doar scrie la echipamentul I/O.

Exemplu:

10 *OPEN#* 5,"K"
20 *PRINT#* 5,"hc 88"
30 *GO TO* 20

trimite date la iesirea caii 5 care este asociata prin instructiunea *OPEN#* partii de jos a ecranului.

Pentru a anula asignarea caii s la un canal se foloseste instructiunea *CLOSE#* s. Dupa instructiunea *CLOSE#* calea s poate fi asociata altui canal.

La initializarea sistemului se deschid automat caile 0-3, cu urmatoarea asignare:

calea 0 - canalul K

calea 1 - canalul K

calea 2 - canalul S

calea 3 - canalul P

Instructiunea *LIST#* s,n listeaza programul incepind cu linia n pe calea s.
Comanda *INKEY\$#* s citeste un octet de pe calea s.

3.24 ALTE ECHIPAMENTE

Retea

Poate fi folosita o periferie de tip retea pentru conectarea mai multor calculatoare HC intre ele.

Interfata seriala

Interfata standard RS-232 permite conectarea unui HC cu alt calculator sau alte periferice inzestrate cu aceasta interfata. Utilizarea se realizeaza folosind cuvintele cheie *OPEN#*, *CLOSE#*, *MOVE*, *ERASE*, *CAT* si *FORMAT*.

Interfata disc flexibil

Interfata de disc flexibil permite cuplarea a unui sau doua minidrive-uri. Acestea au avantajul unei operatiuni de incarcare-salvare mult mai sigura si mai rapida in comparatie cu caseta.

Interfata de creion optic si Kempston

Aceasta interfata da posibilitatea utilizatorului sa cupleze un Joystick tip KEMPSTON, pentru jocuri sau aplicatii practice si cuplarea unui creion optic folosit pentru desenat.

CONECTORI

CONECTOR AUDIO

- 1.4 MIC Iesire 500mV
- 2 GND Masa
- 3 EAR Intrare 1-5V
- 5 +5V

CONECTOR JOYSTICK

- 1.STINGA
- 2.DREAPTA
- 3.SUS
- 4.FOC
- 5.N.C.
- 6.N.C.
- 7.Selectie 1
- 8.Selectie 2
- 9.JOS

CONECTOR VIDEO

- 1.SHVL
- 2.MASA
- 3.R (rosu)
- 4.G (verde)
- 5.B (albastru)
- 6.BRGH. (intensitate)
- 7.VIDEO OUT
- 8.Hsync.
- 9.Vsync.

CONECTOR ALIMENTARE

- 1,4.a si 1,4.b - +9Volti
- 2,3.a si 2,3.b - GND (masa)

*NOTA

1^o Cele doua manete tip joystick sint simple contacte normal deschise care se suprapun peste tastele numerice de pe tastatura. Acestea se inchid cind sint manevrate intr-o directie anume (sus, jos, stinga, dreapta). Prin selectie 1 sau 2 se alege grupul de taste cu care se lucreaza: 1,2,3,4 si 5 pentru 1 si respectiv 6,7,8,9 si 0 pentru 2.

2^o Conectorul VIDEO este o mufa RACK 9 contacte mama. Acesta permite cuplarea unui monitor alb/negru sau color, PAL, pe un semnal video complex (pin.7) sau a unui monitor RGB.

3. Comanda COPY descrisă la pagina 45 funcționează numai cu imprimanta tip SPECTRUM SINCLAIR.

CONECTOR DE EXTENSIE

A11 -	28	-
A9 -	27	- A10
BUSACK -	26	- A8
ROMCS -	25	- RFSH
A4 -	24	- M1
A5 -	23	-
A6 -	22	-
A7 -	21	- WAIT
RESET -	20	-
BUSRQ -	19	- WR
-	18	- RD
-	17	- IORQ
-	16	- MREQ
-	15	- HALT
GND -	14	- NMI
IORGE -	13	- INT
A3 -	12	- D4
A2 -	11	- D3
A1 -	10	- D5
A0 -	9	- D6
CLK -	8	- D2
GND -	7	- D1
GND -	6	- D0
SLOT -	5	- SLOT
-	4	-
+5V -	3	- D7
A12 -	2	- A13
A14 -	1	- A15

INTEPRINDEREA DE CALCULATOARE ELECTRONICE

CERTIFICAT de GARANTIE Nr. *36796*

Produsul: HOME COMPUTER FELIX HC 91

Seria: *39296*

Data fabricatii: *03/94*

Produsul este garantat pe o perioada de 6 luni de la instalare si maxim 12 luni de la data vnzarii. Acest termen se prelungeste cu perioada de timp in care produsul a stat in reparatie. Orice defectiune care se datoreaza unei utilizari necorespunzatoare sau manipulari defectuoase sau neconforma cu instructiunile de instalare anuleaza garantia. Nu se considera in garantie produsele care au fost desigilate de beneficiar. Cumparatorul are obligatia de a prezenta certificatul de garantie la intocmirea reclamatiei privitoare la defectarea produsului.

Produsele care sint reparate in termenul de garantie de catre intreprinderi neautorizate prin "agreement" de Intreprinderea de Calculatoare Electronice isi pierd garantia.

Sef serviciu CTC

S-au facut probe de functionare, am verificat configuratia standard si certificatul de garantie.

Semnatura beneficiar.....

Data vnzarii: *20 XII 94*

Semnatura si stampila unitatii de desfacere