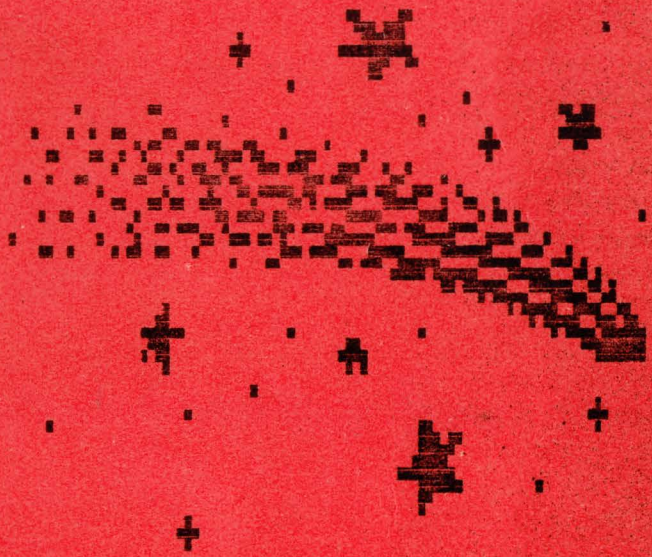
Tim-S Plus

șapte calculatoare într-unul singur

Pănescu Dumitru
Sîrbescu Doinița & Sîrbescu Ștefan



Vol. I

Timișoara, 1990

Tim-S Plus

sapte calculatoare intr-unul singur

Panescu Dumitru

Sirbescu Doinita & Sirbescu Stefan

Societatea μHard

Fabrica de memorii electronice si
componente pentru tehnica de calcul

Editura "TM", Timisoara 1990

Tim-S Plus

sapte calculatoare intr-unul singur

Compatibilitate soft:

Calculatorul personal ZX-Spectrum 48K
Calculatorul personal ZX-Spectrum +
Calculatorul personal ZX-Spectrum 128K
Calculatorul personal ZX-Spectrum +2
Calculatorul personal ZX-Spectrum +3
Interface I
Sistemul de operare CP/M V2.2

Panescu Dumitru

Sirbescu Doinita & Sirbescu Stefan

Facultatea Electrotehnica
Institutul Politehnic Traian Vuia

Societatea µHard

Fabrica de memorii electronice si
Componente pentru tehnica de calcul

Muzica: Panescu Dumitru

Timisoara, 04.09.89

Catedra de calculatoare a Institutului Politehnic;
B-dul Vasile Pirvan, nr. 2, tel. 961.12330.371, Timisoara

Fabrica de memorii electronice si componente pentru tehnica de calcul; B-dul Gh.Lazar, nr. 9, tel. 961.30078, Timisoara

Cuvint inainte

Ca urmare a succesului obtinut pe plan international de calculatoarele familiei ZX-Spectrum - 48K, Spectrum+, Spectrum 128 si reprezentantele noii generatii, Spectrum+2 si Spectrum+3 - si, nu in ultimul rind, ca urmare a receptivitatii si interesului de care s-au bucurat si inca se mai bucura replicile romanesti ale calculatorului ZX Spectrum 48K - calculatoarele HC-85 si Tim-S - avem placuta sarcina de a va prezenta Tim-S Plus, calculator care imbina cele mai bune trasaturi ale modelelor anterioare cu comoditatea pe care o implica accesul rapid la informatia de pe discurile flexibile.

Totalitatea noutatilor pe care un echipament le aduce fata de modelele anterioare reprezinta una din conditiile ce stau la baza viabilitatii lui ca produs. Spectrum 128 si +2 au adus noi in familie posibilitati de editare si sunet superioare, Spectrum+3 a completat aceste posibilitati cu avantajele care decurg din cresterea capacitatii de stocare si a vitezei de manevrare a informatiei. La rindul lui, Tim-S Plus vine sa-si completeze predecesorii prin compatibilizarea cu sistemul de operare CP/M, cel mai raspindit si cunoscut soft utilizat azi in tara pe microcalculatoare de 8 biti. Implementarea acestui sistem de operare, la care se adauga trasaturile modelelor anterioare, fac din calculatorul Tim-S Plus cel mai complet echipament, ca posibilitati de utilizare, in cadrul familiei ZX.

Scopul principal al acestei carti este punerea la dispozitia tuturor celor care au preocupari in domeniul calculatoarelor personale pe opt biti si in special viitorilor posesori ai calculatorului Tim-S Plus a unor informatii cit mai complete ce caracterizeaza simbioza hard-soft a acestui calculator. Ne exprimam speranta ca parametrii functionali ai acestei simbioze vor corespunde exigentelor tot mai numeroase si ridicate ce caracterizeaza actualele generatii de utilizatori de calculatoare personale din tara si am dori ca o confirmare a acestei stari de spirit sa o constituie insasi implicarea calculatorului Tim-S Plus in cit mai multe activitati de cercetare si proiectare, care sa permita implementarea de noi aplicatii, ce vor avea la baza posibilitatile functionale ale acestui calculator. In acest sens, ne exprimam convingerea ca domeniul invatamintului va fi printre primii lui beneficiari, cu atit mai mult cu cit conceperea si finalizarea actualei structuri are la baza o sustinuta activitate de cercetare si proiectare, depusa de cadre de cercetare si studenti ai Catedrei de Calculatoare a Institutului Politehnic din Timisoara de-a lungul a cinci ani de activitate.

Cu siguranta ca nu numai calculatorul Tim-S Plus, ci majoritatea calculatoarelor personale concepute si realizate in ultimii ani in tara (PRAE, aMIC, Spectim, Tim-S,...) n-ar fi devenit astazi certitudini fara aportul substantial si competent pe care actualul producator al acestor calculatoare - Fabrica de memorii electronice si componente pentru tehnica de calcul din Timisoara - l-a adus la realizarea lor. Remarcind in mod deosebit relatiile de colaborare ce s-au stabilit la sfirsitul anului 1984 intre Catedra de calculatoare si Fabrica de Memorii din Timisoara, ne exprimam convingerea ca tocmai bunul mers al acestei colaborari reprezinta piatra de temelie care a stat - si speram ca va mai sta - la baza activitatii de proiectare si realizare a calculatoarelor personale din Timisoara.

Se afirma de multe ori ca realizarea, in decursul ultimilor cinci ani, a cinci calculatoare personale in Timisoara se datoreaza unei conjuncturi favorabile. Daca-i asa, atunci nu trebuie uitat ca o asemenea conjunctura este, in primul rind, rezultatul

activitatii omenesti, sub toate aspectele ei constructive. A existat, asadar, si pentru Tim-S Plus un concurs de imprejurari care a favorizat posibilitatea conceperii si realizarii lui in forma care vi se prezinta dumneavoastra. Dar este bine si corect sa apreciem, in acest sens, ca Tim-S Plus nu este rodul muncii unui grup restrins. Este bine si corect sa-i apreciem si sa vi-i prezentam si pe aceasta cale pe cei care au adus contributii remarcabile la finalizarea actualei lui versiuni.

Exista, in faza de geneza a oricarei realizari mai deosebite, un spirit organizator, care reuseste de cele mai multe ori sa aseze si sa directioneze pe calea cea buna activitatile, netezind asperitatile diferentelor de opinii, mai ales tehnice, prin mentinerea in permanenta pe o pozitie care sa-i permita rezolvarea problemelor care se ivesc la granita dintre discipline, sau a celorlalte probleme - mult mai dificile si delicate - pe care le creeaza granita dintre caractere; care intelege ca nu se poate face cercetare si proiectare fara o baza materiala adecvata si care, mai mult de-atit, actioneaza conform acestei intelegeri; care intelege ca activitatea de conceptie se poate desfasura, in anumite situatii, mult mai bine daca se ocolesc constringerile organizatorice. Un astfel de spirit l-a constituit, pentru proiectul Tim-S Plus, ing.Dorin Daraban.

Reusita unei realizari este conditionata, pe linga factorii esentiali pe care-i reprezinta o baza materiala corespunzatoare si o buna organizare, si de modul in care generatiile reusesc sa se puna de acord, mai exact de modul in care realizatorii reusesc sa combine, pe ansamblu, vigoarea si experienta maturitatii cu fantezia si entuziasmul tineretii. Unei asemenea fantezii si unui asfel de entuziasm ii datoreaza in mare parte existenta nu numai calculatorul Tim-S Plus, ci si mai bine de jumatate din filele acestei carti, care ar fi prins cu siguranta forma si in lipsa contributiilor sustinute si nemijlocite de editare pe care acestea le-au adus, dar,... mult mai tirziu. Ioan Sima, Adi Zamfirescu, Florin Zmaranda, Laszlo Illyes, Ladislau Csosz,... Sa-i apreciem si sa-i pretuim fiindca ei au stiut sa dea o noua calitate conditiei de student, sustinid proiectul Tim-S Plus prin eforturi care depasesc cu mult cerintele unei asemenea conditii. Sa-i apreciem si sa-i pretuim cu-atit mai mult cu cit ei reprezinta generatiile care vin si care constituie pentru cei mai multi dintre noi sperante pentru viitorul national al tehnicii de calcul.

A mai existat, de asemenea, un grup de ingineri, care odata conturat Tim-S Plus ca prototip, au transformat acest prototip in produs serie, contribuind la atingerea acestui stadiu prin numeroase solutii tehnice, rezultat al experientei acumulate in ultimii ani in cadrul Fabricii de memorii din Timisoara, in domeniul constructiei calculatoarelor: Mircea Radita, Remus Telescu, Dan Petrar, Dan Slimovschi, Dan Bostan,... Aspectul esential care trebuie apreciat in activitatea pe care au depus-o in acest sens este acela ca solutiile pe care le-au adus au venit, de cele mai multe ori, din partea unor ingineri care, fara a-si manifesta veleitati de proiectant, au acumulat o bogata experienta practica, ce le permite, nu de putine ori, sa indrepte greseli pe care le fac proiectantii. Este incontestabil ca existenta calculatorului Tim-S Plus ca produs serie se datoreaza si acestor "ingineri de productie", care, poate tocmai prin faptul ca reusesc de multe ori sa repare greselile celorlalti prin solutii proprii - in conditiile mai putin favorabile cercetarii pe care le ofera productia de serie - fac parte din panoplia adevaratilor proiectanti de calculatoare din tara. Panoplie la nivelul careia vor avea intodeauna acces pe usa din fata.

Se cuvin aici subliniate si energia, daruirea si constiinciozitatea tehnica pe care domnisoara Helga Graf le-a investit in implementarea modelelor experimentale ale tuturor celor cinci calculatoare amintite. Fara aportul unor astfel de calitati calea spre finalizarea calculatoarelor respective ca produs serie ar fi fost incontestabil mai lunga si mai grea.

Un calculator personal nu trebuie privit ca o simpla unealta de calcul, cum ar fi, de pilda, o socotitoare cu bile. Structura lui cuprinde elemente mult mai complexe, care cer conditii pretentioase de functionare si care, implicit, necesita atentie sporita si mult discernamint in proiectare. Un grup aparte in schema unui astfel de calculator il reprezinta elementele anexe, care, avind rol in cresterea substantiala a posibilitatilor de utilizare ale calculatorului, ii sint de cele mai multe ori apropiate ca importanta. Unele dintre aceste elemente anexe le constituie interfetele care permit afisarea informatiei pe terminale de tip video: monitor, televizor, etc. Rezolvarea problemelor legate de implementarea acestor interfete in schema calculatorului Tim-S Plus se datoreaza, in exclusivitate, experientei de mai bine de 25 de ani a tehn.pr.Constantin Nanasi. Tot cu privire la categoria elementelor anexe remarcam contributiile deosebite pe care ing.Emil Badilescu le-a adus atit la proiectarea sursei de alimentare, cit si la elaborarea sectiunii acestei carti care descrie modul ei de realizare si de punere in functiune.

Pentru majoritatea celor care au lucrat cu un calculator este limpede ca, oricit de ingenioase si sigure ar fi solutiile tehnice adoptate la realizarea lui fizica si oricit de bogata ar fi gama posibilitatilor sale constructive, competitivitatea lui fata de produse similare este de neconceput atita timp cit nu exista o baza consistenta si diversificata de programe, care sa permita exploatarea acestor posibilitati. In acest sens, este putin probabil ca interesul pentru Tim-S Plus sa fi fost azi la fel de mare daca n-ar fi existat sansa adaptarii la actuala configuratie a unor programe cu caracter aplicativ, care au sporit considerabil posibilitatile lui de utilizare, mergind uneori pina acolo incit s-a reusit implementarea, pe structura actuala, a unor programe dedicate proiectarii asistate de calculator care sint comparabile ca performante cu programe similare functionale pe calculatoarele de 16 biti ale familiei... IBM-PC, asa cum este cazul pachetului de programe realizate de catre ing.Alice si Dan Sfirlea. Tot la acest capitol trebuie amintit si efortul reusit al ing.Horatiu Moldovan de a realiza o extensie grafica a sistemului de operare CP/M.

Dar au mai existat si alte sanse care au facut posibila finalizarea proiectului Tim-S Plus. Sanse oferite prin acele rare si neasemuite exemple de daruire pe care numai o pasiune voluntara le poate oferi. Sanse numite ing.Corneliu Barbulescu, principalul animator al compatibilizarii sistemului de operare CP/M pe Tim-S Plus, pentru care a si implementat programele pentru tratarea consolei in standard VT52. Sanse numite ing.Daniela si Adrian Spilca, ce au dat un exemplu unic in felul lui, sustinind - in conditii oarecum vitrege si la vremea cind Tim-S Plus inca nu era conturat ca produs serie - viitorul acestui calculator, prin implementarea primei versiuni functionale a modulului de lucru Interface I, a carui prezentare, care constituie tema capitolului 5 al acestei carti, le apartine aproape in exclusivitate.

Nu trebuie uitat nici sprijinul competent si nemijlocit pe care realizatorii l-au primit - inca de la inceputurile compatibilizarii sistemului de operare CP/M pe Tim-S Plus - din partea ing.Petru Iovescu, prof.Ionel Jian si prof.Mircea Holban, unii dintre putinii adepti ai acelei conceptii care priveste informa-

tia tehnica drept un bun al cunoasterii universale, a carei circulatie trebuie sa ramina straina de barierele pe care i le pot ridica in cale interesele personale.

Datoreaza mult calculatorul Tim-S Plus si tineretii spirituale si energiei cu care prof.Cornel Secu vine in intimpinarea tuturor celor care au preocupari in domeniul calculatoarelor personale, prin organizarea de activitati lunare de prezentare - in cadrul Clubului Programatorilor al Casei Universitarilor din Timisoara - a ultimelor realizari in acest domeniu, reusind astfel sa adune intr-un mozaic - pe linga preocupari in domeniul artei si culturii - si preocupari de inalta tinuta stiintifica. O reusita care-i face cinste.

Si mai merita subliniat un aspect, urmare fireasca a rememorarii celor care au contribuit decisiv la realizarea calculatorului Tim-S Plus: in marea lor majoritate acestia reprezinta absolventi sau actuali studenti care s-au realizat la scoala catedrelor de Automatizari si Calculatoare sau Electronica si Masuri ale Institutului Politehnic din Timisoara. Imi exprim convingerea ca acesta constituie un prilej cum nu se poate mai potrivit de nedisimulata mindrie pentru cei care, in acest cadru, i-au format ca viitori ingineri, reusind sa le imprime o farima din acea putere si ratiune superioare fara de care lucrurile se pot impinge cu greu mai departe pe calea evolutiei.

...Asadar, daca acest calculator se va dovedi util scopurilor pentru care a fost gindit si realizat, atunci nu trebuie uitat ca finalizarea lui ca produs de serie se datoreaza eforturilor multiple, complexe, consistente si de multe ori de neinlocuit pe care cei implicati in elaborarea lui le-au depus. Fiecare in felul lui, fiecare cu posibilitatile lui, fiecare cu ciudateniile lui. Apreciem ca unul dintre cele mai importante elemente care a facut posibila o asemenea finalizare este ca toti acestia au reusit sa depaseasca dificultatile inerente pe care le implica diferentele tehnice de opinii si stil care apar in cadrul unei colaborari. Preluind o idee a lui Emil Racovita, se poate spune despre acestia ca n-au fost nici bogati, nici puternici, nici renumiti, dar constienti de ceea ce trebuiau sa faca. Au faptuit greseli si au fost prada slabiciunilor omenesti, dar au lucrat cit au stiut mai bine si au facut tot ce au putut, reusind sa duca la bun sfirsit lucruri a caror realizare nu sta chiar la indemina oricui. In ce priveste finalitatea eforturilor pe care le-au depus, se naste o intrebare fireasca: "A meritat, este oare util rezultatul lor?" Ar fi lipsit de sens sa incercam sa apreciem noi daca rezultatul acestor eforturi - concretizate in posibilitatile de utilizare ale calculatorului - sint justificate sau nu. Consideram ca dumneavoastra sinteti singurii indreptatiti sa aprecieze si sa dea un raspuns corect - pozitiv sau negativ - acestei intrebari, raspuns care va trebui sa decurga direct din modul in care Tim-S Plus va reusi sa faca sau nu fata pretentiilor dumneavoastra, si care poate confirma sau nu daca asemenea eforturi, mari sau mici, n-au fost in zadar.

Este posibil ca raspunsul pe care-l veti da sa nu fie conform sperantelor realizatorilor cu privire la acest calculator. Nu ma-ndoiesc ca si-n acest caz vor avea puterea sa pastreze drept prilej de satisfactie un crimpei din acel amalgam de esecuri si impliniri, de renuntari si bucurii pe care numai procesul cunoasterii si al creatiei autentice il poate oferi, indiferent daca este vorba de cunoastere si creatie materiala sau spirituala. Fiindca, nu uitati, ei cel putin au avut curajul sa incerce. Iar indraznetii intodeauna inving!

Panescu Dumitru

Timisoara 03.31.44, Septembrie, 1989

Cuprins

1 Introducere

Scopul de baza care a determinat editarea acestei carti consta in sistematizarea unui material documentar cit mai complet cu privire la structura si posibilitatile de lucru ale calculatorului Tim-S Plus. Daca acest scop a fost indeplinit, reusita se datoreaza in mare parte faptului ca, pe linga contributiile proprii pe care autorii le-au adus la conceperea si finalizarea cartii, acestia au beneficiat si de existenta unui bogat si eficient sistem tehnico-informational. Amintim, in acest sens, documentatia CP/M a calculatoarelor din familia M18 si M118, Junior, MS100, IBM-PC, accesibila noua mai ales prin intermediul discurilor flexibile. Multumim si pe aceasta cale celor care s-au straduit sa elaboreze asemenea tip de "documentatie disc", a carei popularizare ne exprimam convingerea ca nu poate decit sa imbogateasca si sa consolideze experienta romaneasca in domeniul tehnicii de calcul. Se cuvine subliniat ca toata aceasta documentatie a fost restructurata si - de multe ori - completata conform cerintelor calculatorului.

Autorii au mai beneficiat, de asemenea, de existenta unui material documentar consistent cu privire la utilizarea calculatoarelor familiei Spectrum. Majoritatea publicatiilor care tin de acest material au fost traduse si - cu mici transformari - sintetizate si adaptate acestei carti.

Daca "documentatiile disc" vizeaza in special aspecte legate de utilizarea calculatorului Tim-S Plus in modul de lucru CP/M, in schimb traducerile trateaza modul de lucru Spectrum. Am cautat sa furnizam informatii cit mai cuprinzatoare cu privire la ambele moduri de lucru, nestiind care vor fi preferintele dumneavoastra in acest sens.

Am cautat, de asemenea, sa va punem la dispozitie informatii cu privire la folosirea unor programe utilitare. In acest scop am facut o prezentare succinta a unui numar minim de astfel de programe - atit pentru modul de lucru Spectrum, cit si CP/M -, care sa va permita sa elaborati si dezvoltati singuri programe de aplicatii, mai ales in limbaj de asamblare Z-80. In aceasta categorie de programe amintim grupul MONS-GENS pentru modul Spectrum si grupul ED-M80-L80-ZSID pentru CP/M.

Sesizind ca utilizarea calculatoarelor Tim-S Plus din primele serii a necesitat, in majoritatea cazurilor, transfer de informatii intre aceste calculatoare si celelalte tipuri de mini si microcalculatoare existente in tara (Independent, MS100, IBM-PC, etc...), am atasat acestei carti si un submanual de prezentare a unuia din programele utilitare cele mai des folosite in acest scop, programul KERMIT.

Cei care nu dispun de sansa de a intra in posesia actualei versiuni a acestei carti, au dezlegare unanima din partea autorilor in privinta utilizarii oricaror posibilitati - rezonabile - care sa le permita achizitionarea ei.

Daca veti avea rabdare sa-i parcurgeti continutul, veti constata cu siguranta ca prezentarea hard a calculatorului este cu mult inferioara, ca intindere, prezentarii soft. Promitem ca la editia urmatoare vom inlatura aceasta nedreptate!

P.D.

2 Prezentare generala

2 Prezentare generala

2.1 Arhitectura

Din punct de vedere tehnologic, calculatorul Tim-S Plus este realizat pe doua placi de circuit imprimat, dublu strat, care contin urmatoarele circuite si blocuri functionale:

a)Placa de baza (pentru amplasare vezi fig.28):

- microprocesor Z80-A sau Z80-B;
- memoria RAM sistem - 192KO;
- memoria RAM video - 16KO sau 64KO;
- amplificatoare de magistrale;
- circuitele asociate prelucrarii logice a informatiei video (automatul video);
- dispozitivul de comanda si paginare a memoriei;
- circuite asociate protocolului microprocesor-RAMTV;
- trei generatoare de tact: 12MHz, 14MHz, 16MHz;
- interfata cu una sau doua unitati de disc; cupla pentru disc;
- interfata cu tastatura mecanica; cupla intermediara;
- interfata paralela de intrare pe 8 biti;
- interfata paralela de iesire pe 8 biti;
- interfata serie RS232;
- numaratorul programabil cu trei canale distincte;
- doua circuite 8255, utilizate ca interfete si pentru paginare;
- dispozitive de comutare magistrala de comanda;
- interfata audio tip ZX-Spectrum 48K;
- sloturi pentru extensii; 2 + 4;
- cupla de interfata cu placa video;
- cupla pentru masca din fata (led, buton RESET, difuzor).

b)Placa video (de dimensiuni mai mici) contine:

- interfata audio realizata cu circuitul specializat AY-3-8912A;
- interfata pentru amplasare in retea tip Interface I;
- interfata cu casetofonul;
- interfata cu monitor color;
- interfata cu monitor monocrom;
- interfata cu televizor monocrom (modulator);
- comutator tact.

Ambele placi sint fixate in interiorul unei cutii care mai contine:

- una sau doua unitati de disc de 5.25";
- adaptoare de magistrala;
- placuta circuit imprimat pentru masca din fata;
- cupla tastatura;
- cupla alimentare;
- cupla interfata paralela intrare;
- cupla interfata paralela de iesire;
- cupla RS232;
- cupla televizor monocrom;
- cupla monitor monocrom;

- cupla monitor color;
- cupla audio mono si stereo;
- cupla interfata cu casetofonul;
- 3 cuple de interfata retea;
- 4 cuple Canon, de doua tipuri, pentru interfata cu exteriorul; 50 de contacte.

Placuta de pe masca din fata contine:

- buton RESET;
- led semnalizare tensiune de +5V;
- difuzor.

2.2 Compatibilitate soft

Tim-S Plus este primul calculator personal romanesc care reuneste laolalta compatibilitatea cu calculatoarele familiei ZX-Spectrum - incepind cu ZX Spectrum 48K si mergind pina la ZX Spectrum +3 - si compatibilitatea cu sistemul de operare CP/M.

Desprindem, din multitudinea de cauze care au determinat pe autori sa asigure posibilitatea utilizarii acestui calculator si ca sistem CP/M, urmatoarele doua ca fiind motivatia si - in acelasi timp - justificarea implementarii acestei optiuni de lucru:

- universalitatea sistemului de operare CP/M; aceasta universalitate este subliniata si la nivel national, unde CP/M-ul a devenit unul dintre cele mai utilizate sisteme de operare pe microcalculatoarele M18, M118, CUB-Z, Junior, TPD...

- ideea de a respecta traditia realizatorilor versiunilor de calculatoare ZX-Spectrum si a proiectantilor autohtoni de modele de calculatoare compatibile cu familia ZX; esenta acestei traditii consta in dotarea fiecarui model nou din familie atit cu toate posibilitatile de utilizare ale celor anterioare, cit si cu altele noi, care au darul de a-l face superior predecesorilor.

Tim-S Plus poate fi utilizat cu softul scris pentru toate modelele familiei de calculatoare ZX Spectrum, incepind cu varianta ZX Spectrum 48K si sfirsind cu ZX Spectrum +3. Poate fi de asemenea utilizat in versiunea de lucru ZX-Spectrum 48K cu Interface I atasat.

Asigurarea compatibilitatii hard/soft a calculatorului cu un sistem CP/M permite utilizarea citorva sute de programe, mai ales utilitare, care intra in biblioteca sistemului de operare CP/M. Asigurarea compatibilitatii cu calculatoarele familiei ZX-Spectrum permite utilizarea citorva mii. Acest lucru inseamna ca exista deja o cantitate vasta de soft scrisa pentru Tim-S Plus. Asadar, exista literal mii de titluri de programe validate acoperind domenii diverse de aplicatie - jocuri, utilitare, muzica, stiinta, educatie si multe multe altele - functionale pe acest calculator.

2.3 Instalare

Calculatorul se alimenteaza prin intermediul sursei exterioare, de la reteaua de 220 V. Pentru evitarea pericolului de accidentare, calculatorul trebuie sa fie alimentat numai de la prize prevazute cu legatura la pamint.

Pentru instalare este nevoie de un minim de echipamente, care constituie configuratia de baza a calculatorului. Exista doua configuratii de baza. Acestea sint formate din urmatoarele

parti componente:

- sursa de alimentare externa;
- calculator;
- tastatura;
- una sau doua unitati de disc de 5.25", functie de numarul configuratiei de baza.

2.3.1 Configuratii ale sistemului

Calculatorul Tim-S Plus prezinta diverse configuratii de livrare. Alegerea unei anumite configuratii se face, in general, pe baza cerintelor beneficiarului. Toate configuratiile de livrare se obtin prin suplimentarea uneia din configuratiile de baza (descrise sintetic la paragraful anterior si care sint de asemenea livrabile) cu diverse echipamente periferice. Lucrul cu echipamentele periferice suplimentare (imprimanta, casetofon, etc...) se realizeaza prin intermediul programelor concepute pe Tim-S Plus in mod de lucru Spectrum sau CP/M. Prezentam in continuare detalii despre toate configuratiile de livrare ale calculatorului Tim-S Plus pe care beneficiarii le pot solicita.

Codificarea variantelor se face sub forma generala:

Calculator personal Tim-S Plus XYZ

unde:

X = 2 pentru echipare cu 2 discuri flexibile;
Y = 0 fara imprimanta;
  = 1 pentru echipare cu imprimanta 80 caractere/rind;
  = 2 pentru echipare cu imprimanta 130 caractere/rind;
Z = 1 pentru echipare cu monitor monocrom;
  = 2 pentru echipare cu monitor color.

Configuratiile de livrare sint urmatoarele:

Var	Pret livrare	Cas	Mon A-N	Mon Color	Imprimanta 80c	Imprimanta 130c	Set docum
201	135850	1	1	-	-	-	1
202	171500	1	-	1	-	-	1
211	171500	1	1	-	1	-	1
212	223500	1	-	1	1	-	1
221	287000	1	1	-	-	1	1
222		1	1	1	-	1	1

2.3.2 Instalare electrica

Pornirea calculatorului presupune efectuarea operatiilor urmatoare:

a) Se introduce cupla de alimentare, care va face legatura intre sursa si calculator.

b) Se introduce cupla de tastatura, lateral, dreapta.

c) Se introduce cablul de monitor TV, atit in TV cit si in

calculator, in cupla de iesire pentru tipul monitorului TV utilizat (monocrom, color), vezi anexa B.

d) Se introduce cablul de alimentare a sursei in priza de 220 V. Similar se procedeaza cu cablul de alimentare al ventilatorului din cutia calculatorului.

e) Se conecteaza sursa externa prin apasarea butonului de pe masca din fata a sursei (sub leduri).

f) Se conecteaza monitorul si celelalte echipamente periferice (imprimanta, casetofonul,... daca exista). Este bine sa se astepte cca. 20 s dupa alimentarea perifericelor, dupa care se poate trece la faza urmatoare.

g) Se verifica daca toate ledurile de pe masca sursei sint aprinse. Daca nu, vezi anexa.

h) Se introduce discul sistem in unitatea de disc din stinga (unitatea A) si se inchide clapeta unitatii.

i) Se apasa butonul RESET, de pe masca din fata, in vederea initializarii calculatorului.

j) Se apasa tasta corespunzatoare selectiei modului de lucru solicitat (vezi paragraful 3.3.1).

k) Se verifica daca (dupa RESET) portiunea de ecran delimitata de BORDER se coloreaza conform modului de lucru selectat (vezi paragraful 3.3.1). Daca nu, se consulta paragraful 10.13.

l) Se elibereaza tasta si se asteapta schimbarea culorii BORDER-ului. Daca dupa cca. 10 secunde de asteptare, BORDER-ul devine clipitor, inseamna ca nu exista soft operational pe disc si se consulta paragraful 10.13.

m) Functie de modul de lucru selectat, se continua cu operatiile de la paragrafele 3.3.2 ... 3.3.5.

Pentru pornire, operatiile de mai sus se vor executa in ordine, incepind cu a). Pentru oprire, se vor executa operatiile:

n) Se scot discurile flexibile din unitati.

o) Se deconecteaza perifericele.

p) Se deconecteaza sursa externa prin apasarea pe butonul acesteia.

q) Se intrerupe alimentarea sursei externe si a ventilatorului de la priza de 220 V.

Ori de cite ori se va dori trecerea dintr-un mod de lucru operational intr-un alt mod de lucru, se va relua instalarea incepind cu h).

3 Caracteristici ale calculatorului Tim-S Plus

3 Caracteristici ale calculatorului Tim-S Plus

Solutionarea problemelor de compatibilizare a calculatorului Tim-S Plus, atit cu versiunile calculatoarelor familiei ZX-Spectrum, cit si cu sistemul de operare CP/M a dus la aparitia anumitor particularitati de utilizare ale acestui calculator. Subliniind ca aceste particularitati sint specifice numai calculatorului Tim-S Plus, le vom reuni sub numele de caracteristici si vom incerca in continuare sa prezentam citeva elemente esentiale ce privesc aceste caracteristici.

3.1 Utilizarea tastaturii

Tastatura la Tim-S Plus este detasabila de calculator, legatura dintre cele doua realizindu-se prin intermediul unui cablu de cca. 1m, prevazut la capatul dinspre calculator cu o cupla de interconectare. Arhitectura tastaturii consta din dispunerea a doua grupe de taste mecanice, intr-o carcasa mecanica. Aceste doua grupe joaca rol distinct in functionarea calculatorului.

Primul grup, format din 4*10+3 taste (stinga) poate fi utilizat sub controlul softului specific familiei de calculatoare ZX-Spectrum. Dispunerea si functiile acestor taste respecta in mare parte dispunerea si functiile tastelor versiunii ZX-Spectrum 48K.

O alta utilizare a primului grup este posibila la Tim-S Plus sub controlul softului sistemului de operare CP/M, mai exact sub controlul componentei BIOS a acestui soft.

La implementarea sistemului de operare CP/M pe calculatorul Tim-S Plus s-a optat pentru tratarea perifericelor de tip CONIN (CONSOLE IN) si CONOUT (CONSOLE OUT) in standardul VT-52. Realizarea programelor din BIOS, care au permis aceasta standardizare - pe de o parte - si existenta unor taste suplimentare pe vechea tastatura a calculatorului Tim-S Plus - pe de alta parte - au fost factorii care au stat la baza reconfigurarii grupului de taste din dreapta la Tim-S Plus. S-au facut schimbari atit la nivel de cablaj in zona matricii din dreapta cit si la nivelul de schema de scanare a tastaturii si soft dedicat acestei scanari, astfel incit sa se ajunga la o suplimentare a matricii cu inca doua coloane. Pe aceste doua coloane s-au dispus cele 11 taste care tin de grupul din dreapta. Mare parte din functiile acestor taste sint tipice CP/M (CTRL, ESCAPE,...) restul de 4 fiind taste functionale, rezervate unei viitoare utilizari.

Asadar, tastatura la Tim-S Plus este formata dintr-o matrice de fire 8*7 (vezi fig.27).

Fiecare tasta este legata la intersectia unei linii cu o coloana, astfel incit prin apasarea ei un fir orizontal se conecteaza cu unul vertical. Firele orizontale (rinduri) sint conectate la partea superioara a magistralei de adrese A8...A15 (corespunzator in fig.27 liniilor IT8...IT15) prin intermediul unor circuite separatoare (fig.26). Cele 7 fire verticale (coloane) sint conectate la magistrala de date prin intermediul unui circuit separator.

La fiecare intrerupere primita, CPU scaneaza tastatura, executind instructia IN (input) de la portul adresat cu #FE, in conditiile in care pe adresa superioara o singura linie de adresa este pe "0". Daca o tasta a fost apasata, octetul venit de la portul #FE contine un zero logic ("0") pe bitul de date corespunzator tastei apasate.

3.2 Moduri de afisare

Afisarea informatiei pe un terminal video la Tim-S Plus a fost conceputa in ideea de a satisface doua moduri specifice de lucru ale acestui calculator: mod de lucru Spectrum si mod de lucru CP/M.

Primul mod de lucru nu a implicat aproape nici o schimbare la nivelul schemelor care compun dispozitivul de prelucrare a informatiei in vederea afisarii (fig.14...20), fata de schema similara de la Tim-S.

Nu acelasi lucru se poate spune despre al doilea mod de lucru - CP/M - a carui solutionare la nivel de afisare a presupus modificari substantiale in schema. Sintetic, aceste modificari se pot grupa astfel:

a) dublarea frecventei de serializare a datelor video (pixelilor);
b) extinderea ferestrei de afisare a informatiei pe orizontala;
c) extinderea memoriei RAM video utilizata de la 8KO la 16KO si multiplexarea suplimentara de adrese pentru acces la aceasta memorie;
d) implementarea optiunii de scroll hard vertical la nivel de linie TV.

Primele trei modificari au asigurat, odata implementate, afisarea informatiei pe TV, sub CP/M, in formatul 80 de caractere pe rind TV, cu 8*8 pixeli /caracter. Vom numi acest mod de afisare dubla rezolutie, plecind de la ideea dublarii frecventei de serializare, care a restrins la jumatate dimensiunea orizontala a unui pixel. Daca vom numerota caracterele de pe linie de la stinga la dreapta cu valori consecutive intre 0 si 79, atunci informatia necesara conturarii unui caracter par va fi preluata din prima pagina de 8KO a memoriei video, iar pentru conturarea unui caracter impar din a doua pagina. Aceasta regula este valabila pentru situatia in care bitul B3 = 0 (vezi paragraful 8.1 aliniat a). Pentru B3 = 1 asocierea de mai sus intre caracterele afisate si pagina se inverseaza.

3.3 Utilizarea discurilor flexibile

Tim-S Plus este un calculator a carui utilizare se bazeaza pe interfata cu unitatile de disc flexibil de 5.25". Discurile flexibile au mai multe roluri in utilizarea calculatorului, functie de momentul utilizarii si de modul de lucru selectat al calculatorului. Vom prezenta in cele ce urmeaza citeva tipuri de discuri utilizate subliniind, acolo unde este cazul, momentul si scopul utilizarii lor:

- Disc sistem (SD): folosit in faza de initializare a sistemului; contine fisiere cu soft de baza (versiuni de BASIC, CP/M, Interface I); unul din aceste fisiere este preluat si incarcat in memorie in faza de initializare a sistemului. Alegerea se face pe baza unei optiuni la nivelul tastaturii. Capacitatea totala a discului este de 720KO.

- Disc BASIC +3 (BD): disc de lucru sub controlul programelor din softul de BASIC +3. Contine fisiere organizate conform standardului adoptat pentru discurile calculatorului ZX Spectrum +3. Prima versiune de disc BASIC +3 utilizata pe Tim-S Plus prezinta capacitate de utilizare de 2*173KO pe doua fete. La urmatoarele versiuni aceasta capacitatea a crescut la 706KO, restul pina la 720KO fiind rezervat.

- Disc Interface I (ID): disc de lucru sub controlul programelor din softul de 48 BASIC si Interface I. Contine fisiere organizate intr-o forma care sa permita utilizarea lor prin intermediul instructiunilor de lucru cu discul din softul de Interface I. Practic s-a realizat o deviere a fazei de tratare a acestor instructiuni de pe unitatea de Microdrive (prezenta la Interface I original) pe unitatile de disc de 5.25". Prima versiune de disc Interface I utilizata pe Tim-S Plus prezinta capacitate de utilizare de 128KO. La urmatoarele versiuni aceasta capacitate va creste de circa 5 ori.

- Disc CP/M (CD): disc de lucru sub controlul programelor din softul CP/M, ver.2.2. Capacitatea discului este de 720KO, din care 698KO rezervati utilizatorului (pentru orice disc CP/M nou formatat).

3.3.1 Selectia modurilor de lucru

Calculatorul Tim-S Plus fiind un echipament care permite diverse moduri de utilizare, vom descrie in cele ce urmeaza citeva aspecte legate de selectia unui anumit mod de lucru. Asa cum reiese si din prezentarea operatiei de instalare (paragraful 2.3.2), selectia modului de lucru se face in faza de initializare a sistemului. In aceasta faza, sub controlul unui program incarcator, existent in memoria EPROM, se preia de pe SD un fisier de date, selectat dintr-un grup de fisiere existente pe SD, pe baza apasarii unei anumite taste. Pentru a elimina incertitudinea preluarii si interpretarii corecte a unei taste apasate, s-a prevazut tot in cadrul programului incarcator o modalitate de a marca acest aspect. Astfel, imediat dupa apasarea butonului RESET, daca totul decurge normal, BORDER-ul va fi negru. Dupa apasarea tastei, BORDER-ul ramine pe culoarea asociata tastei respective. Vom descrie in continuare cum se face asocierea intre "mod de lucru", "tasta" si "culoare BORDER".

Tasta	Culoare	Mod de lucru
1	Albastru	CP/M
2	Rosu	+2 BASIC cu Interface I
3	Violet	+3 BASIC

3.3.1.1 Mod de lucru +3 BASIC

Dimensiunea softului preluat de pe discul sistem si care trateaza acest mod de lucru se cifreaza la 64KO. Conceptual acest soft este divizat in patru pagini distincte a cite 16KO fiecare. Accesul la programele acestor pagini se face sub controlul lui +3 BASIC, secvential, in cadrul primului sfert (#0000...#3FFF), iar memorarea lor se face in cadrul blocului de memorie RAM dinamic BR2. Precizari suplimentare cu privire la selectia acestor pagini se gasesc la paragraful 8.1. Modul de lucru +3 BASIC permite utilizarea tuturor perifericelor atasate calculatorului Tim-S Plus, cu exceptia interfetei serie si a RAM-discului suplimentar, care cer o tratare speciala, la nivel de program utilitar.

Exista posibilitatea utilizarii unui RAM disc local, de 58KO, sub controlul instructiunilor din +3 BASIC. O caracteristica distincta a lui +3 BASIC fata de modelele anterioare din familie o prezinta sistemul de operare pentru lucrul cu discul, care prezinta comenzi tipice de acces la unitatile de disc floppy

de 5.25", la nivel de fisier. In acest context, s-a considerat ca nu mai este necesara compatibilizarea softului +3 BASIC cu Interface I.

Imediat dupa lansarea in executie a softului de +3 BASIC, acesta va afisa pe ecran, in centru, un chenar cu 4 optiuni de lucru, selectabile cu ajutorul tastelor care permit deplasarea cursorului pe verticala (modul de lucru Spectrum 48K), urmate de tasta ENTER. Prezentarea detaliata a particularitatilor fiecarei optiuni de lucru din cele 4 selectate este obiectul capitolului 4.

3.3.1.2 Mod de lucru +2 BASIC si Interface I

Softul de +2 BASIC se intinde pe lungimea a 32KO de memorie si se poate divide in doua pagini distincte a cite 16KO: 128 BASIC si 48 BASIC. Fata de versiunea +3, +2 BASIC nu mai prezinta sistemul de operare cu discul in standardul +3. Exista totusi si la acest nivel posibilitatea utilizarii discului flexibil, prin intermediul softului destinat lucrului cu unitatiile microdrive (ZX-Spectrum 48K). Practic s-a realizat o deviere a rutinelor de tratare a accesului de pe unitatile microdrive pe unitatile de disc de 5.25".

Precizam ca s-a urmarit ca pe cit posibil schimbarile necesare devierii sa nu afecteze esential compatibilitatea cu softul scris pentru versiunea de lucru cu microdrive. De asemenea, precizam ca softul de tratare a optiunii Interface I este operational sub +2 BASIC, atit in modul de lucru 128 BASIC cit si in modul 48 BASIC.

Imediat dupa lansarea in executie a programelor +2 BASIC, ca si la +3, se afiseaza un chenar de optiuni de lucru asemenea celui de la +3 BASIC. Optiunile se selecteaza similar cu descrierea facuta la 3.3.1.1.

3.3.1.3 Mod de lucru CP/M

Versiunea de CP/M implementata pe Tim-S Plus este CP/M ver.2.2.

O prima particularitate a acestei versiuni este ca pe linga compatibilitatea pe care o prezinta cu standardul VT52 pentru dialogul cu consola, permite definirea unor culori la nivel de ecran, pentru BORDER, PAPER si INK.

Selectia acestor moduri se poate face astfel:

Optiune	Faza 1	Faza 2	Faza 3	Faza 4
BORDER	[CS]+[SS]	[B]	[n]	[ENTER]
PAPER	[CS]+[SS]	[C]	[n]	[ENTER]
INK	[CS]+[SS]	[X]	[n]	[ENTER]

[CS] - tasta CAPS SHIFT;
[SS] - tasta SYMBOL SHIFT;
[X] - tasta X;
[n] - tasta numerica cu numere cuprinse intre 0 si 7, functie de culoarea dorita.

Este stiut ca CP/M-ul este un sistem de operare care se bazeaza pe utilizarea discurilor.

Discurile CP/M de pe Tim-S Plus prezinta compatibilitate cu

discurile de pe calculatorul CUB-Z. Dam mai jos citeva trasaturi ale unui astfel de disc:

- organizat dubla fata, pe 80 de cilindri;
- 1 cilindru este compus din 9 sectoare pe o fata a discului si inca 9 sectoare pe cealalta, pentru aceeasi pozitie a capetelor de scriere/citire (2 piste fata/spate = 1 cilindru);
- 9 sectoare/pista;
- dubla densitate, 512 octeti/sector;
- primii 2 cilindri sint rezervati.

Din analiza parametrilor de mai inainte rezulta ca un disc CP/M la Tim-S Plus cuprinde 720K, din care 698K sint disponibili la nivelul utilizatorului.

In cazul in care se doreste prelucrarea unor informatii sub CP/M, dar organizate in format de 40 piste (tip Junior), se poate face o conversie de fisiere de pe sistemul 40 de piste pe sistemul 80 de cilindri. In acest scop se utilizeaza programul C408088, operational sub CP/M-ul de pe Tim-S Plus.

3.3,1.4 Diferente intre modurile de lucru

Intre cele 4 moduri de lucru BASIC (48K, 128K, +2 si +3) exista diferente exprimind evolutia facilitatilor oferite de familia ZX.

Fata de Tim-S 48K BASIC, modurile mai evoluate prezinta urmatoarele optiuni:

- editor de BASIC pe tot ecranul (litera cu litera);
- optiune de CALCULATOR care permite calcule prompte;
- posibilitate utilizarii unui meniu principal de selectie;
- posibilitatea utilizarii unui meniu de editare, cu toate facilitatile oferite;
- posibilitatea de principiu a generarii sunetului complex.

Intrucit modul 128K este identic cu modul +2 (exceptie facind optiunea "Tape Tester" din meniul principal) vom face referire numai la +2.

Meniurile de selectie (absente la Tim-S) aduc posibilitatea selectiei simple a unei suboptiuni de lucru din mai multe posibile (vezi 4.1).

La modul +2 meniul principal are ca prima optiune "Tape Loader" (incarcare de pe caseta), iar la modul +3 optiunea "Loader" (incarcare de pe disc sau caseta).

Exista diferenta intre modurile de lucru +2 si +3 si la nivelul meniului de editare. Astfel prima optiune a acestui meniu la +2 este "128 BASIC", in timp ce la +3 gasim "+3 BASIC". Fiecare din cele doua optiuni apeleaza versiuni de interpretor BASIC diferite, versiunea corespunzatoare modului +3 fiind cea mai evoluata.

Modul +2 pune la dispozitie discul RAM de capacitate 62K, utilizabil cu ajutorul instructiunilor **SAVE!**, **LOAD!**, **MERGE!**, **CAT!** si **ERASE!**, caracterizate prin aditionarea semnului ! fata de sintaxa de la lucrul cu caseta tip Tim-S. Tim-S Plus mai pune de asemenea la dispozitie extensia de limbaj tip **Interface I** (utilizabila si in modul 48 BASIC), cu facilitatile respective (vezi cap.5). La modul +3 discul RAM este discul **M:**, cu care se poate lucra prin intermediul sintaxei **+3DOS** (descrisa in cap.4.27, 4.28).

Suboptiunea "48 BASIC" din meniul principal a fost pastrata pentru motiv de compatibilitate cu vechea versiune de Tim-S, pentru cei care vor sa lucreze in modul cunoscut la acest calculator. Facilitatile oferite de versiunile evoluate nu sint acce-

sibile in modul "48 BASIC", motiv pentru care nu se recomanda lucrul in acest mod. Exceptie fac programele scrise special pentru modul de lucru "48 BASIC". In general, programele scrise in modul "48 BASIC" (Tim-S) si extensia "BASIC TIM-EXT 01" sint executabile in modul +2; programele scrise in modul +3, folosind +3DOS, sint executabile doar sub +3; programele scrise in modul CP/M sint executabile doar in acest mod.

4 Limbajul +3 BASIC

4 Limbajul +3 BASIC

Limbajul +3 BASIC este constituit dintr-un pachet de programe care ocupa un volum de memorie de 64KO, rezidente la Tim-S Plus in cadrul blocului BR2 de memorie RAM. Scopul acestui capitol este prezentarea integrala a sectiunilor limbajului +3 BASIC, in vederea utilizarii calculatorului in modul de lucru Spectrum +3.

4.1 Introducere in +3BASIC

Din sumar:

Meniul principal
Meniul de editare
Renumerotarea unui program BASIC
Listarea la imprimanta
Introducerea unui program
Lansarea in executie a unui program
Comenzi si instructiuni
Operatii simple cu discul: formatare, salvare, incarcare, catalog

4.1.1 Meniul principal

Imediat dupa lansarea in executie a softului de +3 BASIC, se va afisa pe ecran meniul principal, compus din patru optiuni. Totodata acest meniu indica ce unitati de disc sint operationale.

Selectia unei optiuni din meniu se face mutind bara de selectie cu ajutorul tastelor "cursor sus" (CAPS SHIFT si 7) si "cursor jos" (CAPS SHIFT si 6) in dreptul optiunii dorite, dupa care se tasteaza ENTER.

Prezentam in continuare cele 4 optiuni.

4.1.1.1 Loader

Alegeti aceasta optiune atunci cind doriti sa incarcati programe pentru +3, +2 sau 128 (Spectrum) de pe banda. Optiunea mai poate fi utilizata si in cazul in care se doreste incarcarea de soft de aplicatie dedicat, de pe disc (pentru detalii vezi sectiunea 4.20.5).

4.1.1.2 +3 BASIC

Selectati aceasta optiune atunci cind doriti sa lucrati cu programe in limbajul +3 BASIC.

4.1.1.3 Calculator

Optiune selectata in situatia in care se doreste utilizarea calculatorului Tim-S Plus ca si un... "calculator de buzunar".

4.1.1.4 48 BASIC

Alegeti aceasta optiune cind doriti sa lucrati cu programe pentru Tim-S (48K Spectrum), sau daca vreti sa utilizati Tim-S Plus ca pe un Tim-S.

4.1.2 Meniul de editare

Calculatorul +3 are un editor avansat, pentru introducerea, modificarea si executia programelor BASIC. Pentru a intra in editor, selectati optiunea +3 BASIC din meniul principal (daca nu stiti cum, cititi sectiunea 4.1.1).

Meniul principal va dispare de pe ecran si se va afisa in partea de jos mesajul "+3 BASIC" si in coltul stinga sus va apare cursorul (clipitor).

La baza ferestrei de afisare exista o linie de culoare neagra, la inceputul careia este scris momentan **+3 BASIC**, acesta fiind numele editorului sub al carui control ne aflam. In general informatia inscrisa pe aceasta linie ne spune in care din cele patru sectiuni ale softului de **+3BASIC** ne aflam.

Tot in aceasta zona se rezerva loc pe lungimea a doua linii de text pentru afisarea mesajelor de eroare. Aceasta zona o vom numi **ecranul mic**. In cadrul ecranului mic se va face editarea in cazul selectarii optiunii de lucru **Screen**.

Acum tastati EDIT. Veti observa disparitia cursorului si aparitia unui nou meniu. Acest meniu se numeste meniul de editare.

Optiunile din meniul de editare se selecteaza la fel ca cele din meniul principal (utilizind tastele cursor si ENTER).

Sa luam optiunile pe rind:

4.1.2.1 +3 BASIC

Aceasta optiune anuleaza meniul de editare si restaureaza cursorul. Daca se apasa EDIT din greseala, atunci aceasta optiune permite reintoarcerea in program fara a afecta programul.

4.1.2.2 Renumber

In programele BASIC se utilizeaza numere de linie pentru a stabili ordinea de executie a instructiunilor. Veti completa asemenea numere (pot fi orice numar intreg cuprins intre 1 si 9999) la inceputul fiecarei linii de program pe care o veti introduce. Selectarea optiunii Renumber, face ca numerele liniilor de program BASIC sa inceapa cu 10 si sa aiba pasul 10. Instructiunile BASIC care refera numere de linie (GO TO, GOSUB, LINE, RESTORE, RUN si LIST) vor avea aceste referinte renumerotate.

Daca renumerotarea nu se poate face din diverse motive, cum ar fi, de exemplu, situatia in care nu este program in calculator, sau renumerotarea ar genera numere de linie mai mari de 9999, atunci se genereaza un sunet si meniul dispare.

4.1.2.3 Screen

Aceasta optiune muta cursorul in partea de jos a ecranului si permite ca liniile de program BASIC sa fie introduse si editate in aceasta zona. Aceasta este foarte util cind se lucreaza si grafic, intrucit editarea nu distruge imaginea de pe ecran.

Pentru revenirea la partea de sus a ecranului, selectati din nou optiunea Screen din meniul de editare.

4.1.2.4 Print

Daca este conectata o imprimanta, aceasta optiune va lista programul la imprimanta. Cind listarea este gata, meniul va dispare si cursorul va reveni. Daca din anumite motive calculatorul nu poate face listarea (de exemplu daca imprimanta nu este conectata), atunci apasind tasta BREAK se revine la editor.

4.1.2.5 Exit

Aceasta optiune va reintoarce la meniul general; calculatorul retine programul existent in memorie. Daca doriti sa reveniti la program, selectati optiunea +3BASIC din meniul general. Daca selectati optiunea 48BASIC (sau daca apasati butonul de RESET) atunci orice program existent in memorie se va pierde. Puteti utiliza optiunea calculator din meniul general, fara sa pierdeti nimic.

4.1.3 Introducerea unui program si lansarea lui in executie

Acum resetati calculatorul si selectati +3BASIC; introduceti linia de mai jos:

```
10 for f=1 TO 100 step 10
```

si tastati ENTER. Atentie! Spre deosebire de modul "clasic" de lucru cu tastatura de la ZX Spectrum 48K, in +3 BASIC instructiunile se introduc litera cu litera. Calculatorul va reafisa cuvintele FOR, TO si STEP cu litere mari; linia va arata astfel:

```
10 FOR f=1 TO 100 STEP 10
```

Calculatorul a emis totodata si un scurt sunet si va muta cursorul la inceputul urmatoarei linii de program.

Daca linia ramine afisata cu litere mici si auziti un sunet emis de calculator, aceasta semnaleaza ca ati introdus ceva gresit. Observati ca se schimba si culoarea cursorului in rosu, cind este detectata o eroare,, aceasta insemnind ca trebuie sa corectati linia inainte de a fi acceptata de calculator. Pentru aceasta utilizati tastele cursor pentru pozitionare si apoi, cu DELETE, stergeti caracterele gresite si introduceti caracterele corecte. In final tastati ENTER.

Acum introduceti urmatoarea linie (semnul doua puncte se obtine cu SYMB SHIFT si Z, iar semnul minus se obtine cu SYMB SHIFT si J):

```
20 plot 0,0:draw f,175:plot 255,0:draw -f,175
```

si tastati ENTER. Pe ecran veti vedea:

```
10 FOR f=1 TO 100 STEP 10
20 PLOT 0,0:DRAW f,175:PLOT 255,0:DRAW -f,175
```

Ultima linie de introdus pentru acest program este:

```
30 next f
```

si tastati din nou ENTER.

Dupa aceasta, cursorul se va pozitiona dupa linia 30, ultima introdusa. Acum, ca un exercitiu, editati linia 10 si schimbati-i numarul in 100 si apoi in 255 si la loc in 10.
Acum tastati:

```
run
```

si apoi ENTER si observati ce se intimpla. Prima data linia cu mesaj si liniile programului sint sterse si apoi se executa programul, desenind figura pe ecran si apoi se opreste cu mesajul:

```
0 OK, 30:1
```

Nu va faceti probleme din cauza mesajului! Tastati ENTER. Ecranul se va sterge si bara (linia) cu mesajul va reveni ca dealtfel si liniile programului. Aceasta dureaza aproximativ o secunda, timp in care calculatorul nu poate primi intrari de la tastatura.

4.1.4 Comenzi si instructiuni

Pina acum ati facut majoritatea operatiilor necesare pentru a programa si utiliza calculatorul. Mai intii i-ati spus calculatorului ce sa faca, dindu-i instructiuni. Fiecare instructiune sau comanda are cel putin un cuvint cheie BASIC. Cuvintele cheie alcatuiesc vocabularul calculatorului si multe din ele au nevoie de parametri. In comanda DRAW 40,200 de exemplu, DRAW este cuvintul cheie, iar 40 si 200 parametri (argumentele) care spun calculatorului unde sa deseneze linia.

Acum tastati EDIT si selectati optiunea Screen. Tastati ENTER si introduceti run si din nou ENTER. Programul se va executa la fel ca mai inainte doar ca, daca apasati din nou ENTER, ecranul nu se sterge, puteti deplasa liniile programului in sus si in jos, fara a deranja imaginea de pe ecran.

Acum schimbati linia 10 in:

```
10 FOR f=1 TO 255 STEP 7
```

si introduceti:

```
go to 10
```

si apoi ENTER. Programul modificat va desena o figura putin diferita de cea anterioara. Puteti modifica in continuare programul pentru a obtine si alte figuri.

Notati ca tastind CAPS SHIFT si tastele numerice obtineti urmatoarele:

CAPS SHIFT si 5,6,7 sau 8 muta cursorul;
CAPS SHIFT si 1 cheama meniul de editare;
CAPS SHIFT si 0 sterge un caracter;
CAPS SHIFT si 2 este echivalent cu CAPS LOCK;
CAPS SHIFT si 9 selecteaza modul grafic.

4.1.5 Operatii simple cu discul

Ati vazut pina acum modul in care se introduce un program in calculator. Dar daca dupa aceasta, apasati butonul de RESET sau opriti calculatorul, programul se sterge din memoria calculatorului si la urmatoarea sesiune de lucru va trebui sa-l introduceti din nou. Pentru a evita acest lucru, programele se salveaza din memoria calculatorului pe disc si la punerea ulterioara a calculatorului in functiune, se va incarca programul de pe disc in memoria calculatorului.

Aceasta parte din capitol trateaza aceste doua operatii de salvare (memorare) (SAVE) si incarcare (LOAD) de pe disc. Insa, inainte de a efectua oricare din aceste operatii, veti vedea cum trebuie pregatit discul. Aceasta operatie de pregatire se numeste formatare.

Pentru a exersa cele expuse in continuare, este necesar sa aveti un disc nou, gol.

Chiar daca sinteti familiarizat cu salvarea si incarcarea de pe banda, este bine sa retineti urmatoarele doua lucruri, atunci cind lucrati cu discul.

In primul rind, un disc nou, gol, inainte de a fi inregistrat, trebuie sa fie formatat. Retineti ca procesul de formatare sterge discul.

In al doilea rind, este important ca numele fisierelor de pe disc sa fie corecte. Pe banda, numele de fisier pot varia ca lungime sau pot fi chiar omise. Pe disc insa, numele de fisier, trebuie sa fie conform cu anumite standarde (veti citi despre aceasta pe scurt in sectiunea acestui capitol intitulata "Numele de fisier").

4.1.5.1 Formatarea unui disc

Formatarea unui disc trebuie privita ca un proces, care creaza pe disc o retea organizata pe care, ulterior, datele vor putea fi puse sau de pe care pot fi luate. Se dau in continuare citeva detalii de formatare referitoare la prima versiune de disc compatibil ZX Spectrum +3.

Procesul de formatare, divide discul in 360 zone separate. Sint 40 de piste concentrice, care incep cu numerotarea din exteriorul discului (pista 0 in exterior, iar 39 cea dinspre interior) si fiecare pista la rindul ei este divizata in 9 sectoare.

Fiecare sector poate memora pina la 512 octeti de date; deci spatiul total disponibil pe o fata de disc este de 180 kiloocteti (180K). 7K din cei 180K sint rezervati pentru a putea fi utilizati de calculator si deci restul de 173K sint disponibili pentru programele dvs.

In continuare, vom formata un disc nou si vom salva programul de mai jos:

```
10 FOR f=1 TO 255 STEP 7
20 PLOT 0,0:DRAW f,175:PLOT 255,0:DRAW -f,175
30 NEXT f
```

care ar trebui sa fie in memorie de la exercitiul anterior. (Verificati ca programul de mai sus este in memoria calculatorului, tastind ENTER si apoi

```
list
```

si din nou ENTER. Daca programul nu este in memorie (sau daca ati oprit calculatorul) atunci porniti calculatorul, selectati optiu-

nea +3BASIC si introduceti programul de mai sus.

Introduceti discul (discheta) in unitatea de disc din stinga si tastati:

format "a:"

si ENTER. Indicatorul luminos read/write al discului va incepe sa clipeasca (aceasta inseamna ca se lucreaza cu discul) si dupa citva timp va apare mesajul:

0 OK,0:1.

In acest moment se poate considera incheiata operatiunea de formatare a discului.

Odata formatat un disc, nu mai este necesara nici o formatare ulterioara.

Daca nu apare mesajul de mai sus (sau apare in loc alt mesaj), cititi sectiunea "Mesaje de eroare" de la sfirsitul acestui capitol.

4.1.5.2 Salvarea unui program

Avind discul formatat, acum se poate trece la salvarea programelor.

Pentru ca programele sa poata fi identificate trebuie ca fiecarui fisier la salvare sa i se dea un nume. De exemplu, fiindca programul de mai sus deseneaza o figura, sa il salvam utilizind numele "desen.fig". Pentru aceasta tastati:

save "desen.fig"

si ENTER. Dupa citeva secunde veti primi mesajul:

0 OK,0:1.

Acum programul este salvat pe disc. Daca nu primiti mesajul de mai sus (si apare alt mesaj in loc), cititi sectiunea intitulata "Mesaje de eroare" de la sfirsitul capitolului.

4.1.5.3 Numele de fisier

Un nume de fisier pe disc consta din doua parti (cimpuri). Primul cimp este obligatoriu si poate sa contina pina la 8 caractere (litere si cifre, dar fara spatiu sau semne de punctuatie). In exemplul de mai sus acest cimp este "desen". Al doilea cimp este optional. Poate avea pina la 3 caractere (din nou fara spatiu sau semne de punctuatie). In exemplul de mai sus "fig " este al doilea cimp; acest cimp mai poarta si denumirea de extensie.

Daca utilizati doua cimpuri intr-un nume de fisier, atunci ele trebuie separate prin punct (in cazul de mai sus "desen.fig").

4.1.5.4 Catalogul unui disc

Un catalog de disc (in ordine alfabetica) poate fi vizualizat tastind:

cat

si apoi ENTER. Toate numele fisierelor de pe o parte de disc vor fi afisate, impreuna cu lungimea fisierului (la numarul intreg de kiloocteti cel mai apropiat). Spatiul ramas liber pe disc va fi de asemenea afisat (pentru exemplul de mai sus, veti obtine):

DESEN.FIG 1K
172 K free.

4.1.5.5 Incarcarea unui program

Sa presupunem ca ati oprit calculatorul si dupa citva timp doriti sa incarcati programul salvat. Aceasta se face prin apasarea butonului de RESET si selectarea optiunii +3BASIC din meniul general. Introduceti:

load "desen.fig"

si ENTER (avind discheta operationala in unitatea din stinga). Dupa citeva secunde veti vedea mesajul:

0 OK,0:1

Acum programul este incarcat de pe disc in memoria calculatorului. Tastati ENTER si veti vedea afisate liniile programului.

(Daca nu primiti mesajul de mai sus, si apare alt mesaj in loc, cititi sectiunea intitulata "Mesaje de eroare").

Odata incarcat programul, poate fi lansat in executie tastind:

run

si ENTER.

4.1.5.6 Mesaje de eroare

Daca nu tastati corect instructiunile, este foarte probabil ca veti primi diverse mesaje de eroare. Daca se intimpla asa, identificati mesajul (din cele expuse mai jos), cititi explicatiile date si luati in consecinta actiunea corectiva.

Drive not ready

Inseamna ca probabil nu ati introdus discul in unitatea de discuri. Daca discul este introdus, retrageti-l si introduceti-l din nou.

Disk is write protected

Inseamna ca ati incercat sa formatati sau sa salvati pe un disc care are orificiul marginal de protectie la scriere inchis (acoperit). Retrageti discul, deschideti orificiul de protectie si reintroduceti discul.

Bad filename
sau
Invalid filename

Inseamna ca ati incercat sa incarcati sau sa salvati utilizind un nume de fisier eronat (sau fara nume de fisier). Cititi sectiunea intitulata 'Numele de fisier' din acest capitol si incercati din nou.

Disk is already formatted
A to abandon, other key to continue

Inseamna ca ati incercat sa formatati un disc care a fost deja formatat. In general, un disc are nevoie de formatare o singura data, la inceputul utilizarii lui. In cazuri rare un disc poate fi distrus si necesita o noua formatare. Exceptind acest caz, tastati intotdeauna A cind vedeti mesajul de mai sus.
Nota: Daca nu tastati A, procesul de formatare va incepe si intregul disc va fi sters indata ce apasati alta tasta. Daca vedeti ca un anumit disc cere mereu formatarea, atunci este preferabil sa nu il mai utilizati intrucit este foarte probabil ca discul in sine este distrus.
Unele comenzi vor produce mesaje care dau optiunile:

Retry, Ignore or Cancel?

Daca primiti aceste optiuni, atunci:
- tastind R (după ce ati luat masura corectiva necesara) veti determina calculatorul sa reincerce executia comenzii;
- tastind I veti determina calculatorul sa ignore motivul care a cauzat mesajul de eroare la comanda tastata si sa continue (tastarea lui I nu este recomandata, doar daca stiti exact ceea ce faceti);
- tastind C abandoneaza comanda (aceasta poate fi urmata de aparitia altui mesaj).

Alte informatii (impreuna cu detaliile despre utilizarea memoriei RAM disc si despre utilizarea unitatii de caseta) le gasiti in sectiunile 4.20.10,...4.20.12. Un ghid pentru +3DOS (Sistemul de operare pe disc pentru +3) il veti gasi in sectiunile 4.27.* (toate sectiunile ce tin de 4.27).

4.2 Notiuni simple de programare

Din sumar:
Programe
Editarea programelor
PRINT, LET
RUN, LIST, NEW
REM, INPUT, GO TO, CONTINUE
Oprirea unui program

Introduceti urmatoarele prime doua linii ale unui program (care va afisa suma a doua numere):

```
20 print a
10 let a=10
```

Veti observa pe ecran urmatoarele:

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

Intrucit liniile sint numerotate, nu se executa imediat, ci se memoreaza pentru executia ulterioara. Ati observat ca liniile

se afiseaza (si se executa) in ordinea crescatoare a numerelor liniilor; liniile sint sortate de +3 in momentul terminarii introducerii fiecarei linii.

Notati de asemenea ca ati introdus liniile cu litere mici, iar cuvintele cheie (PRINT si LET) au fost convertite in litere mari in momentul in care linia a fost acceptata de +3. De aici inainte cuvintele cheie vor fi tiparite cu litere mari (dvs. puteti continua sa le introduceti cu litere mici).

Acum mai introduceti:

```
15 LET b=15
```

si tastati ENTER.

Schimbati linia 20 in:

```
20 PRINT a+b
```

Puteti introduce din nou toata linia 20, dar este mai usor sa o editati. Mutati cursorul (utilizind tastele cursor) dupa litera a si introduceti: +b.

Verificati linia 20 si daca este in regula, apasati ENTER. De data aceasta cursorul se va muta in josul liniilor si ecranul va arata astfel:

```
10 LET a=10
15 LET b=15
20 PRINT a+b
```

Ce faceti cu acest program?

- atribuiti (cu instructiunea LET) valoarea 10 variabilei a;
- atribuiti valoarea 15 variabilei b;
- spuneti calculatorului (cu instructiunea PRINT) sa afiseze suma acestor doua valori (sa adune continutul celor doua variabile).

Lansarea in executie a acestui program se face tastind RUN si apoi ENTER. Va tipari suma celor doua numere: 25.
Acum introduceti:

```
PRINT a,b
```

si tastati ENTER. Observati ca valorile variabilelor a si b s-au pastrat si veti vedea pe ecran.

```
10          15
```

Greseli... tipice

Daca din greseala introduceti o linie, de exemplu:

```
12 LET b=8
```

si doriti sa stergeti linia, tastati:

```
12
```

si apoi ENTER. Linia 12 va dispare si cursorul va apare unde era linia 12.

Acum introduceti:

```
30
```

si tastati ENTER. Din moment ce linia 30 nu exista, cursorul va fi pozitionat dupa ultima linie din program. Daca introduceti un numar de linie inexistent (ca 30), atunci +3 va plasa cursorul acolo unde ar fi fost linia daca ea ar fi existat. Acesta poate fi un mod util de pozitionare in cadrul programelor mari, dar ATENTIE! este si periculos, intrucit daca linia a existat intr-adevar inainte de introducerea numarului, ea sigur nu va mai exista dupa aceasta manevra.

Pentru a lista un program pe ecranul TV, introduceti

LIST

si apoi ENTER. In cazul in care doriti listarea de la o anumita linie, puneti numarul liniei dupa LIST, de exemplu:

LIST 15

si apoi ENTER.

Ati observat la programul anterior ca am inserat linia 15 intre liniile 10 si 20; aceasta nu s-ar fi putut face daca in loc de 10 si 20 am fi avut 1 si 2. In practica, se recomanda sa se lase locuri libere intre numere. Retineti ca numerele liniilor trebuie sa fie cuprinse intre 1 si 9999. Daca la un moment dat ajungeti in situatia in care doriti sa inserati o noua linie si nu aveti spatiu, puteti utiliza optiunea Renumber din meniul de editare. Pentru aceasta, tastati EDIT si alegeti optiunea Renumber din meniu; aceasta pune spatiul dintre linii la valoarea 10. Incercati procedeul cu programul anterior.

Comanda NEW

Aceasta comanda sterge orice program existent in memorie si variabilele. Tastati

NEW

si apoi ENTER si veti vedea ce se intimpla. De acum inainte nu vom mai mentiona si tastarea lui ENTER la sfirsitul comenzii sau liniei.

Dupa NEW, va apare primul meniu din care alegeti +3BASIC. Acum introduceti urmatorul program, care transforma temperaturile din grade Fahrenheit in grade Celsius.

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

Daca lucrati cu litere mici, veti observa de exemplu la linia 10 ca numai cuvintul REM va fi inlocuit cu litere mari. Instructiunea GO TO nu este necesar sa aiba spatiu, poate fi si intr-un singur cuvint GOTO.

Lansati programul in executie cu RUN. Veti observa pe ecran scris gr F si gr C. Linia 10 a fost ignorata de +3, ea fiind folosita pentru comentarea programului. Tot ce urmeaza intr-o linie de program dupa cuvintul cheie REM, va fi ignorat de calculator.

Linia 30 tipareste o linie goala.

Instructiunea INPUT din linia 40 asteapta introducerea unui numar, ca valoare pentru variabila f. Tastati un numar (apoi ENTER) si programul va tipari rezultatul dupa care asteapta din nou introducerea unui numar. Acest ciclu de introducere si tiparire se formeaza cu ajutorul instructiunii GO TO 40, care spune calculatorului sa continue executia de la linia 40.

Un mod de oprire a executiei programului si de iesire din acest ciclu este urmatorul: in loc de numar, tastati instructiunea STOP (SYMBOL SHIFT si A, apoi ENTER). Va apare mesajul:

```
H STOP in INPUT in line 40:1
```

Mesajul da urmatoarele informatii: din ce cauza s-a oprit executia programului si unde (grupul :1 afisat dupa 40 semnifica prima instructiune din linia cu numarul 40). Daca doriti continuarea executiei programului, tastati:

```
CONTINUE
```

si calculatorul va cere un alt numar. La CONTINUE, calculatorul are memorat numarul de linie din ultimul mesaj (se presupune ca mesajul nu a fost OK) si face salt la acea linie, deci in cazul nostru la linia 40, la instructiunea INPUT.

Acum, opriti din nou programul si inlocuiti linia 70 cu:

```
70 GO TO 31.
```

Nu veti sesiza nici o deosebire in executia programului, cu toate ca linia 31 nu exista; in asemenea cazuri se face salt la prima linie existenta dupa acest numar. La fel se intimpla si pentru comanda RUN (de fapt RUN inseamna RUN0).

Continuati introducerea numerelor pina cind se umple ecranul; in acest moment intregul ecran se va deplasa in sus, pierzindu-se prima linie de sus si raminind o linie goala in partea de jos. Aceasta actiune se numeste "scroll" (rulare).

Alta problema care trebuie evidentiata este virgula din instructiunea PRINT din linia 50. Virgula se foloseste pentru a incepe tiparirea sau in marginea din stinga a liniei sau la mijlocul ecranului. Astfel linia 50 face ca gradele Fahrenheit sa fie tiparite in stinga, iar gradele Celsius la mijlocul liniei.

Daca in loc de virgula se utilizeaza ; urmatorul numar sau caracter va fi tiparit imediat dupa precedentul. Alt semn de punctuatie ce poate fi folosit cu PRINT este apostroful ('). Utilizind apostrof, tot ce trebuie tiparit, va apare la inceputul urmatoarei linii ecran. Aceasta se intimpla implicit la sfirsitul fiecarei instructiuni PRINT. Daca nu doriti acest lucru, puteti pune o virgula sau punct si virgula la sfirsitul instructiunii PRINT. Pentru exemplificare inlocuiti linia 50 cu fiecare din urmatoarele linii si observati ce se intimpla:

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

Ultima linie 50 este echivalenta cu 50 PRINT f'. A nu se confunda virgula sau punct si virgula cu doua puncte (:), semn utilizat pentru concatenarea mai multor instructiuni in aceeasi linie. De exemplu:

```
PRINT f: GO TO 40.
```

Acum adaugati la program urmatoarele linii:

```
100 REM program
110 INPUT "Cum va numiti?",n$
120 PRINT "Buna ziua ";n$;"!"
130 GO TO 110
```

Acesta este un program separat de cel anterior, dar pot coexista amindoua in calculator. Pentru executie folositi RUN 100.

Intrucit instructiunea INPUT din linia 110 asteapta un sir (un caracter sau mai multe caractere) veti vedea ca sint afisate ghilimele " " pentru a evita erorile. Introduceti numele si apasati ENTER. La urmatoarea introducere, vor aparea din nou ghilimelele. Puteti sa le stergeti (mutind cursorul la dreapta si apoi DELETE de doua ori) si introduceti n$. Din moment ce nu sint ghilimele, calculatorul va cauta variabila n$ si va atribui sirul introdus de dvs. (executa instructiunea LET n$=n$).

Daca doriti sa opriti programul, stergeti ghilimelele si introduceti STOP.

Sa analizam acum comanda RUN 100. Cu aceasta calculatorul porneste executia de la linia 100. Prezentam acum diferenta dintre RUN 100 si GO TO 100. RUN 100 sterge mai intii toate variabilele si ecranul si dupa aceea actioneaza ca si GO TO 100. GO TO 100 nu sterge nimic si se foloseste atunci cind se doreste executia unui program fara a-i sterge variabilele. Alta diferenta o constituie faptul ca se poate tasta RUN fara numar de linie si se va incepe executia cu prima linie a programului pe cind GO TO trebuie intotdeauna urmata de un numar de linie.

Introduceti linia:

```
200 GO TO 200
```

si tastati:

```
RUN 200.
```

Pe ecran nu se va vedea nimic, dar calculatorul este in executie. Pentru a-l opri, apasati tasta BREAK. Calculatorul se va opri din executie cu mesajul:

```
L BREAK into program
```

La sfirsitul fiecarei instructiuni, calculatorul testeaza daca a fost apasata tasta BREAK si, in caz ca da, opreste executia programului in curs si afiseaza mesajul de mai sus. Tasta BREAK poate fi utilizata si cind se lucreaza cu casetofonul, sau imprimanta sau alte periferice conectate la Tim-S Plus. In acest caz mesajul va fi:

```
D BREAK - CONT repeats
```

Instructiunea CONTINUE in acest caz reia executia programului incepind cu instructiunea unde acesta a fost oprit cu BREAK. Lansati din nou programul in executie cu RUN 100 si introduceti un sir pentru n$ (dupa ce ati sters ghilimelele). Intrucit variabila n$ nu a fost definita, veti primi mesaj de eroare:

```
2 Variable not found
```

Acum introduceti:

LET n$="galaxie"

la care veti primi mesajul **0 OK, 0:1**, apoi tastati

CONTINUE

si veti constata ca acum puteti introduce date pentru **n$**. In acest caz instructiunea CONTINUE a executat un salt la instructiunea INPUT din linia 110; nu se ia in considerare mesajul de la instructiunea LET intrucit era OK si sare la instructiunea referita in mesajul de eroare anterior si anume 110. Acest lucru trebuie retinut, deoarece va ajuta sa corectati un program oprit cu eroare si apoi cu CONTINUE sa-i continuati executia de la punctul ramas.

De retinut, ca la mesajul **'L BREAK into program'**, instructiunea CONTINUE executa prima instructiune dupa cea in timpul careia s-a generat mesaj de eroare, iar in cazul mesajului **'D BREAK - CONT repeats'** instructiunea CONTINUE repeta insructiunea care a generat mesaj de eroare.

Instructiunile PRINT, LET, INPUT, LIST, RUN, GO TO, CONTINUE, NEW si REM pot fi utilizate si drept comenzi. De altfel RUN, LIST, CONTINUE si NEW vor fi folosite destul de rar ca instructiuni in program.

4.3 Ramificatii

Din sumar:

CLS, IF, STOP
=, <, >, <=, >=, <>

Programele prezentate pina acum s-au executat liniar, de la prima pina la ultima instructiune si apoi salt din nou la inceput. Acest salt, efectuat cu instructiunea **GO TO**, se numeste salt neconditionat. In practica insa, de cele mai multe ori se pune problema testarii unei anumite conditii si in cazul in care aceasta conditie este indeplinita, numai atunci se face salt. Acest salt se numeste salt conditionat. Instructiunea de test a conditiei are urmatoarea forma:

IF conditie este adevarata (sau nu) THEN executa ceva.

Sa exemplificam aceasta cu un program. Cu **NEW** stergeti programul anterior, selectati **+3BASIC** din meniu si apoi introduceti urmatorul program (un joc ce poate fi jucat de catre 2 jucatori):

```
10 REM Ghiciti numarul
20 INPUT "Introduceti un numar",a:CLS
30 INPUT "Ghiciti numarul",b
40 IF b=a THEN PRINT "Ati ghicit!":STOP
50 IF b<a THEN PRINT "Este prea mic; mai incercati"
60 IF b>a THEN PRINT "Este prea mare; mai incercati"
70 GO TO 30
```

Instructiunea CLS din linia 20 sterge ecranul. Am folosit aceasta instructiune pentru ca al doilea jucator sa nu vada numarul dupa ce a fost introdus. Observati ca instructiunea IF ia forma:

IF conditie THEN xxx

unde **xxx** reprezinta o instructiune sau o succesiune de instruc-

tiuni concatenate cu semnul : (doua puncte).

Conditia este o expresie ce trebuie evaluata la adevarat sau fals; in cazul in care este adevarat, se executa instructiunea (sau instructiunile) ce urmeaza dupa THEN; altfel se sare peste aceasta parte si se executa instructiunea urmatoare.

Cele mai simple conditii sint de comparare intre doua numere (daca sint egale, sau unul este mai mare decit celalalt) sau intre doua siruri. Pentru aceasta se utilizeaza semnele =, <, >, <=, >= si <> utilizate sub numele de operatori relationali.

= inseamna egal;
< inseamna mai mic decit;
> inseamna mai mare decit;
<= inseamna mai mic sau egal;
>= inseamna mai mare sau egal;
<> inseamna diferit (nu este egal).

In programul expus mai sus se compara cele doua numere a si b. Daca sint egale (linia 40) dupa afisarea mesajului este oprita executia programului cu instructiunea STOP. La intilnirea acestei instructiuni, calculatorul afiseaza mesajul:

```
9 STOP statement, 40:3
```

ceea ce inseamna ca instructiunea STOP din linia 40 a provocat oprirea executiei programului.

Linia 50 testeaza daca b este mai mic decit a si linia 60 daca b este mai mare decit a. Daca una din aceste conditii este indeplinita, se va afisa mesajul corespunzator si programul ajunge la linia 70, in care se face salt inapoi la linia 30 si jocul incepe din nou.

Incercati urmatorul program:

```
10 LET a=1
20 LET b=1
30 IF a>b THEN PRINT a;"este mai mare"
40 IF a<b THEN PRINT b;"este mai mare".
```

4.4 Bucle

Din sumar:

FOR, NEXT, TO, STEP

Sa presupunem ca doriti sa introduceti 5 numere pe care vreti sa le adunati. O varianta de program ar putea fi urmatoarea (nu trebuie sa-l introduceti):

```
 10 LET total=0
 20 INPUT a
 30 LET total=total+a
 40 INPUT a
 50 LET total=total+a
 60 INPUT a
 70 LEt total=total+a
 80 INPUT a
 90 LET total=total+a
100 INPUT a
110 LET total=total+a
120 PRINT total
```

Dupa cum se observa, aceasta varianta de program nu este practica; este dificil de scris si de controlat pentru sute de numere de adunat.

Este mai usor sa se initializeze o variabila (un contor) care sa numere pina la 5 si apoi sa se opreasca: introduceti urmatorul program:

```
10 LET total=0
20 LET contor=1
30 INPUT a
40 REM contor numara cite numere au fost introduse
50 LET total=total+a
60 LET contor=contor+1
70 IF contor <=5 THEN GO TO 30
80 PRINT total
```

Observati la aceasta varianta ca numarul de introduceri se poate schimba in linia 70 la un numar oricit de mare, restul programului raminind neschimbat. In limbajul BASIC exista insa doua instructiuni, FOR si NEXT, care simplifica scrierea unui program ca cel de mai sus; aceste doua instructiuni se vor folosi intotdeauna impreuna. Introduceti urmatorul program (care face acelasi lucru ca si anteriorul):

```
10 LET total=0
20 FOR c=1 TO 5
30 INPUT a
40 REM c numara cite introduceri s-au efectuat
50 LET total=total+a
60 NEXT c
80 PRINT total
```

(Pentru a obtine acest program din anteriorul, editati liniile 20, 40 si 60 si stergeti linia 70).

Observati ca am schimbat variabila "contor" in "c" deoarece variabila de control dintr-o bucla FOR...NEXT trebuie sa aiba ca nume o singura litera. In cadrul acestui program, variabila c merge ca valoare de la 1 pina la 5 (1, 2, 3, 4, 5) si pentru fiecare din aceste 5 valori se executa liniile 30, 40 si 50. Cind c a ajuns la 5 se executa linia 80. Practic, instructiunea NEXT c din linia 60 executa urmatoarele: incrementeaza variabila de control a buclei, c (LET c=c+1) dupa care testeaza daca este mai mare decit 5 si daca nu, reia bucla (linia 30) cu noua valoare a lui c; daca da, executa urmatoarea instructiune de dupa NEXT.

Acest lucru il puteti verifica executind programul de mai sus cu RUN, dupa care introduceti:

```
PRINT c
```

si veti obtine valoarea 6. Deci pentru c=5 s-a executat bucla, iar la executia instructiunii NEXT c dupa cum am specificat, c se incrementeaza, deci devine 6, apoi se face testul c>5 (6>5) si cum conditia e indeplinita, se executa urmatoarea instructiune (linia 80), iar c ramine cu valoarea 6.

Nu este necesar ca pasul (valoarea cu care se incrementeaza contorul c) sa fie 1; el poate fi schimbat la orice valoare utilizind cuvintul cheie STEP ca parte componenta a instructiunii FOR. Cu aceasta, forma generala a instructiunii FOR este:

FOR contor = valoare initiala TO valoare finala STEP pas

unde variabila de contor (contor) are numele format dintr-o singura litera, iar valoarea initiala, valoarea finala si pasul sint numere sau ceva (variabile sau expresii) ce poate fi evaluat la numar. Daca schimbati linia 20 in:

```
20 FOR c=1 TO 5 STEP 3/2
```

contorul **c** se va incrementa de data asta cu 1.5, deci valorile lui vor fi 1, 2.5, 4 si 5.5 care este deja mai mare decit 5, deci bucla se va opri dupa a treia introducere. In loc de 3/2 in instructiune se poate pune **STEP 1.5** sau se atribuie valoarea 1.5 unei variabile **s** si se pune **STEP s.** Daca optiunea STEP lipseste, pasul va fi implicit 1.

Incercati sa rulati acest program cu valorile 2 si 10 pentru pas.

Mai incercati sa introduceti linia:

```
1 INPUT f
```

unde f sa fie numarul de introduceri pe care doriti sa le faceti (cite numere doriti se adunati) si modificati linia 20 cu:

```
20 FOR c=1 TO f
```

Executati din nou programul.

Acum incercati ca pentru variabila f sa introduceti valoarea 0 si observati ce se intimpla.

Valorile pentru pas pot fi si negative si pozitive. Incercati pentru exemplificare urmatorul program:

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

Acest program afiseaza numerele de la 1 la 10 in ordine inversa. Am observat la programul dinainte ca bucla FOR...NEXT s-a executat atit timp cit contorul este mai mic sau egal cu valoarea finala. In cazul unui pas negativ, bucla se executa atit timp cit contorul are valoare mai mare sau egala cu valoarea finala.

Schimbati acum in linia 10 a acestui program valoarea 10 cu 100 si lansati programul in executie. Se vor afisa numerele de la 100 la 79 dupa care va apare mesajul "scroll?" in partea de jos a ecranului. Daca apasati **N, BREAK** sau **spatiu**, veti primi mesajul "D BREAK - CONT repeats" si programul se opreste. Daca apasati orice alta tasta, se vor afisa inca 22 linii (inca 22 numere) dupa care va apare din nou mesajul "scroll?".

Acum stergeti linia 30 si executati programul. Va tipari primul numar si se va opri cu mesajul "0 OK". Daca tastati "NEXT n" se va mai executa odata programul si va afisa urmatorul numar.

Atentie mare cind aveti doua sau mai multe bucle; in acest caz, fiecare bucla trebuie sa fie cuprinsa in interiorul alteia. Introduceti pentru exemplificare urmatorul program:

```
10 FOR m=0 TO 6                  --
20 FOR n=0 TO m     --           |
30 PRINT m;":";n;" "; !bucla     | bucla
40 NEXT n           --  cu n     | cu m
50 PRINT                         |
60 NEXT m                        --
```

Observati ca bucla cu contorul n este in intregime cuprinsa in bucla cu contorul m. Aceasta inseamna ca sint inlantuite corect. Nu intercalati buclele, ca de exemplu in urmatorul program:

```
5 REM program eronat
10 FOR m=0 TO 6
20 FOR n=0 TO m
30 PRINT m;":";n;" ";
40 NEXT m
50 PRINT
60 NEXT n
```

Buclele FOR...NEXT trebuie sa fie sau incluse una in alta sau complet separate. Alt lucru care trebuie evitat este saltul in interiorul buclei, din afara buclei. Daca se intimpla asa ceva, este foarte probabil sa obtineti mesajul de eroare "NEXT without FOR" sau "Variable not found".

O bucla FOR...NEXT se poate utiliza si ca o comanda; de exemplu:

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

Mai puteti utiliza si urmatoarea varianta, care exclude saltul in interiorul buclei, intrucit nu exista decit un singur numar de linie:

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

Pasul aici pus 0, face ca bucla sa se repete la infinit.

4.5 Subrutine

Din sumar:
GO SUB, RETURN

Se intimpla deseori ca mai multe parti din program sa faca acelasi lucru si sinteti pus in situatia sa introduceti aceleasi linii de mai multe ori. Pentru a simplifica acest lucru, utilizati asa numitele subrutine apelabile cu ajutorul instructiunii GO SUB.

Forma generala a instructiunii GO SUB este urmatoarea:

```
GO SUB xxx
```

unde xxx este un numar de linie, de fapt numarul primei linii din subrutina.

Ce se intimpla cind calculatorul intilneste instructiunea GO SUB ?; executia programului se continua de la linia cu numarul xxx si in continuare pina la intilnirea instructiunii RETURN (instructiune de revenire din subrutina); in acest moment se face un salt inapoi la prima instructiune dupa cea de salt in subrutina (prima instructiune dupa GO SUB). Aceasta se poate, deoarece la intilnirea instructiunii GO SUB calculatorul isi memoreaza adresa urmatoarei instructiuni intr-un loc de memorie (intr-o stiva), in virful "stivei GO SUB". La intilnirea instructiunii RETURN, aceasta adresa este luata din stiva GO SUB si se face salt la aceasta adresa. Sa luam programul cu ghicirea numarului si sa-l introducem modificat:

```
10 REM program modificat
20 INPUT "Introduceti un numar",a:CLS
30 INPUT "Ghiciti numarul",b
40 IF b=a THEN PRINT "Corect": STOP
50 IF b<a THEN GO SUB 100
60 IF b>a THEN GO SUB 100
70 GO TO 30
100 PRINT "Mai incercati"
110 RETURN
```

Urmatorul program utilizeaza o subrutina (liniile 100 la 150) care afiseaza un tabel in functie de valoarea parametrului n.

```
10 REM tabele pentru 2, 5, 10 si 11
20 LET n=2 : GO SUB 100
30 LET n=5 : GO SUB 100
40 LET n=10 : GO SUB 100
50 LET n=11 : GO SUB 100
60 STOP
70 REM Sfirsit program principal; incepe subrutina
100 PRINT n;"  tabel"
110 FOR  t=1 TO 9
120 PRINT t;" x ";n;" = ";t*n
130 NEXT t
140 PRINT
150 RETURN
```

Puteti apela o subrutina apelata tot dintr-o subrutina. (O subrutina care se apeleaza pe ea insasi se numeste recursiva).

4.6 Date in program

Din sumar:

READ, DATA, RESTORE

Am vazut in programele prezentate anterior ca informatiile sau datele pot fi furnizate programului direct, prin utilizarea instructiunii INPUT. Uneori aceasta metoda este obositoare, poate genera usor erori si trebuie repetata pentru fiecare noua executie a programului. Se poate evita aceasta prin utilizarea instructiunilor READ, DATA, RESTORE.

Exemplu:

```
10 READ a, b, c
20 PRINT a, b, c
30 DATA 1, 2, 3
```

O instructiune READ consta din cuvintul cheie READ urmat de o lista de nume de variabile, separate prin virgula. Are acelasi rol ca si INPUT, doar ca datele nu le primeste de la tastatura (nu le introduceti dvs. de la tastatura) ci si le cauta intr-o instructiune DATA. Fiecare instructiune DATA contine o lista de expresii, numerice sau sir, separate prin virgula. Liniile DATA pot fi puse oriunde in program. Trebuie sa va imaginati ca expresiile din toate instructiunile DATA din program sint puse una dupa alta, formind o lista lunga de expresii, lista DATA. Cind calculatorul ajunge sa execute o instructiune READ, citeste mai intii expresia din DATA; data urmatoare ia urmatoarea expresie si

asa mai departe, pentru instructiuni READ succesive citeste toata lista de instructiuni DATA. Daca numarul de expresii din lista de DATA este mai mic decit numarul de instructiuni READ (se ajunge la sfirsitul listei DATA) se genereaza mesaj de eroare. Daca numarul expresiilor din DATA este mai mare, cele care sint in plus, se ignora.

Nu folositi DATA ca si comanda deoarece instructiunea READ nu le va gasi. Instructiunile DATA trebuie sa fie puse in program.

In programul ultim introdus, linia 10 citeste trei date si le atribuie variabilelor a, b, c. In linia 20 aceste variabile se afiseaza. Instructiunea DATA din linia 30 furnizeaza valorile pentru variabilele a, b, c.

Introduceti acum urmatorul program:

```
10 DATA 2,4,6,8,10,12
20 FOR n=1 TO 6
30 READ d
40 PRINT d
50 NEXT n
```

Observati ca DATA poate fi pusa oriunde in program (inainte sau dupa instructiunea READ). Lansati programul in executie.

O instructiune DATA poate sa contina si date sir. De exemplu:

```
10 FOR a=1 TO 9
20 READ n$
30 PRINT n$
40 DATA "Mercur","Venus","Pamintul","Marte","Jupiter",
   "Saturn","Uranus","Neptun","Pluton"
50 NEXT a
```

Nu este necesar ca instructiunile READ si DATA sa fie in ordine. Aceasta este posibil utilizind instructiunea RESTORE, urmata de un numar de linie. Prin aceasta se specifica instructiunii READ din care instructiune DATA (din linia cu numarul specificat) sa-si ia datele.

Introduceti si rulati urmatorul program:

```
10 DATA 1,2,3,4,5
20 DATA 6,7,8,9
30 GO SUB 110
40 GO SUB 110
50 GO SUB 110
60 RESTORE 20
70 GO SUB 110
80 RESTORE
90 GO SUB 110
100 STOP
110 READ a,b,c
120 PRINT a'b'c'
130 RETURN
```

Daca instructiunea RESTORE nu este urmata de un numar de linie, atunci prima citire se va face din prima instructiune DATA prezenta in program.

Observati efectul instructiunii RESTORE. Stergeti linia 60 si rulati din nou programul, observind ce se intimpla.

4.7 Expresii

Din sumar:

Operatiile +, -, *, /
Expresii, notatii stiintifice
Nume de variabile

Ati observat deja citeva moduri in care Tims-S Plus opereaza cu numere. Executa cele patru operatii aritmetice: adunarea (+), scaderea (-), inmultirea (*) si impartirea (/) si gaseste valoarea unei variabile, fiind dat numele acesteia.

Exemplul:

```
LET taxa=suma*15/100+report
```

ilustreaza combinarea operatiilor. O astfel de combinatie cum este **suma*15/100+report**, se numeste expresie. Cind calculatorul evalueaza (calculeaza) expresia, efectueaza operatiile pe care le contine expresia, deci, in cazul nostru, o inmultire a valorii variabilei **suma** cu 15, apoi impartire cu 100 si apoi adunare cu valoarea variabilei **report**, valoarea finala obtinuta atribuind-o variabilei **taxa**. Ordinea de prioritate in cadrul celor patru operatii este: *, /, +, -. Inmultirea si impartirea au aceeasi prioritate (la fel si adunarea si scaderea) si sint evaluate in ordinea aparitiei in expresie (de la stinga la dreapta). Deci in expresia 8-12/4+2*2, prima operatie tratata va fi impartirea 12/4 care este 3, deci expresia va deveni 8-3+2*2. In continuare se va executa inmultirea 2*2 si deci expresia va fi 8-3+4. Urmeaza scaderea 8-3, dupa care expresia va fi 5+4 si in final se efectueaza adunarea, rezultatul fiind 9.

Incercati urmatoarea instructiune:

```
PRINT 8-12/4+2*2
```

O lista completa a prioritatilor pentru operatiile matematice (si logice) se **gaseste** in partea 31 a acestui capitol. Puteti schimba ordinea de prioritate, utilizind paranteze; in acest caz se evalueaza mai intii ce se afla intre paranteze. De exemplu:

```
PRINT 8-12/(4+2)*2
```

Rezultatul va fi in acest caz 4 in loc de 9.

Expresiile pot contine si siruri sau variabile sir, ca in exemplul urmator:

```
10 LET a$ = "Phobos "
20 LET b$ = "si Deimos "
30 LET c$ = a$ + b$ + "sint sateliti naturali ai planetei
   Marte"
40 PRINT c$.
```

Sa analizam acum numele variabilelor. Numele unei variabile sir trebuie sa fie format dintr-o litera urmata de semnul $. Numele variabilei de control dintr-o bucla FOR...NEXT trebuie sa fie format dintr-o singura litera. Numele variabilelor numerice sint mai putin restrictive, pot contine si litere si cifre, obligatoriu fiind doar ca prima sa fie litera. Se poate utiliza si caracterul spatiu, pentru o recunoastere mai usoara a variabilei, dar acesta nu va fi luat in considerare ca facind parte din nume. De asemenea nu conteaza daca literele din nume sint mici

sau mari. Sint restrictii asupra numelor variabilelor, ca sa nu contina cuvinte cheie ale limbajului BASIC (cu spatii intre ele).

Iata citeva exemple de nume de variabile care sint permise:

```
X
orice lucru nou
v22
acest nume nu este bun deoarece este prea lung
varead
litere
LiteRE
```

Cele doue nume`litere si LiteRE sint considerate aceleasi si se refera la o singura variabila.

Urmatoarele nume nu sint permise:

pi	(Pi este cuvint rezervat);
to be or not to be	(contine To, OR si NOT care sint cuvinte rezervate);
42tone	(incepe cu cifra);
645	(contine numai cifre);
A*B*C	(contine asterisc (*) care nu este nici litera, nici cifra);
X-Y	(contine semnul - care nu este nici litera, nici cifra).

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

Incercati urmatoarele instructiuni si observati ce se intimpla:

```
PRINT 2.34e0
PRINT 2.34e1
PRINT 2.34e2
```

si asa mai departe pina la:

```
PRINT 2.34e15
```

Instructiunea PRINT afiseaza numai opt cifre dintr-un numar. Incercati urmatoarele:

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

si veti observa ca Tim-S Plus memoreaza numarul 4294967295 si poate executa operatii cu el, dar de afisat, afiseaza numai 7 cifre.

Calculatorul utilizeaza calculul in virgula flotanta, adica se separa cifrele numarului (mantisa) si pozitia virgulei (exponent). Introduceti:

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

Numerele se memoreaza ca precizie pe aproximativ noua cifre si jumatate, deci 1e10 este prea mare pentru a fi memorat cu precizie. Inprecizia este mai mare de 1, deci numerele 1e10 si 1e10+1 vor parea egale pentru calculator.

Cel mai mare numar intreg care poate fi memorat complet in memorie este 4294967294.

Sirul "" fara nici un caracter, se numeste sirul nul. Un sir care contine numai spatii nu este identic cu sirul nul.

Incercati urmatoarele:

```
PRINT "Unicul satelit natural al Pamintului este "Luna"?"
```

Cind apasati ENTER, veti observa ca exista eroare si calculatorul nu va accepta eroarea. Atunci cind calculatorul gaseste ghilimelele de la "Luna" le interpreteaza ca un sfirsit pentru sirul "Unicul satelit natural al Pamintului este " si nu isi poate defini sirul "Luna". In acest caz, modificati comanda in:

```
PRINT "Unicul satelit natural al Pamintului este ""Luna""?"
```

Din ce se va afisa pe ecran, fiecare din ghilimelele duble este de fapt una, dar trebuie tastate de doua ori pentru a-l ajuta pe calculator sa le interpreteze corect.

4.8 Siruri

Din sumar:

Formarea de subsiruri utilizind TO

Un sir consta dintr-o succesiune de caractere. Un subsir consta dintr-o succesiune de caractere luate in ordine dintr-un sir.

Forma generala a unui sir este urmatoarea:

```
expresie sir (start TO final)
```

De exemplu:

```
"abcdef"(2 TO 5)
```

este subsirul **bcde.**
Daca lipseste prima valoare de start, **se considera** implicit 1, iar daca lipseste valoarea finala, se considera ultimul caracter al sirului de referinta.

Astfel:

```
"abcdef"(  TO 5) inseamna abcde;
"abcdef"(2 TO  ) inseamna bcdef;
"abcdef"(  TO  ) inseamna abcdef.
```

Ultima reprezentare este echivalenta cu **"abcdef"( ).** Se utilizeaza si forma fara TO, cu un singur numar in paranteze, ceea ce reprezinta un singur caracter. De exemplu **"abcdef" (3)** este echivalent cu **"abcdef" (3 TO 3)** si inseamna c.

Ambele valori - de start si finala - trebuie sa se refere la lungimea reala a sirului. Daca valoarea de start este mai mare decit cea finala, va rezulta sirul nul. Expresia:

```
"abcdef" (5 TO 7)
```

va genera mesaj de eroare "3 Subscript wrong" pentru ca sirul contine doar 6 caractere. Expresiile:

```
"abcdef" (8 TO 7)
```

cit si

```
"abcdef" (1 TO 0)
```

vor genera sirul nul si deci sint permise. Daca valoarea de start sau finala sint negative, se va genera mesajul de eroare: "B integer OUT of range". Urmatorul program ilustreaza citeva din aceste reguli:

```
10 LET a$="abcdef"
20 FOR n=1 TO 6
30 PRINT a$(n TO 6)
40 NEXT n
```

Rulati programul si apoi tastati NEW, dupa care introduceti urmatorul program:

```
10 LET a$="1234567890"
20 FOR n=1 TO 10
30 PRINT a$ (n TO 10), a$((11-n)TO 10)
40 NEXT n
```

In cazul variabilelor sir pe linga operatiile de extragere (prelevare) subsiruri, se mai pot face si atribuiri. De exemplu, tastati:

```
LET a$="Sextantul"
```

si apoi

```
LET a$(6 TO 9)="nte******"
```

si

```
PRINT a$.
```

Intrucit subsirul a$(6 TO 9) are doar 4 caractere, numai primele patru caractere din "nte******" vor fi retinute, restul fiind ignorate. Aceasta este o caracteristica pentru atribuirea subsirurilor: subsirul trebuie sa aiba aceeasi lungime dupa atribuire ca si inainte. Pentru aceasta, calculatorul executa o atribuire de tip Procust, in sensul ca, daca sint caractere in plus (ca in exemplul anterior), acestea sint ignorate, iar daca sint prea putine, subsirul este completat cu spatii pina la lungimea sa.

Expresiile cu siruri complicate necesita paranteze pentru evitarea aparitiei erorilor. De exemplu:

```
"abc"+"def" (1 TO 2) inseamna "abcde";
("abc"+"def")(1 TO 2) inseamna "ab".
```

4.9 Functii speciale

Din sumar:

LEN, STR$, VAL, VAL$, SGN, ABS, INT, SQR, DEF FN

Sintetic putem face urmatoarea reprezentare:

Argument --> Functie --> Rezultat

Functia actioneaza asupra unei valori numita argument, o prelucreaza in mod specific fiecarei functii si in urma prelucrarii se obtine o noua valoare, care constituie rezultatul functiei.

Pentru argumente diferite se vor obtine rezultate diferite.

Pentru fiecare functie exista un domeniu de valori in care trebuie sa fie cuprins argumentul. In caz contrar, se va genera mesaj de eroare.

Apelarea (utilizarea) unei functii se face prin tastarea numelui ei (cuvintului BASIC rezervat) urmat de argument.

Functia LEN

Calculeaza lungimea unui sir de caractere. Argumentul este sirul de caractere caruia doriti sa i se gaseasca lungimea (numarul de caractere).

Tastati urmatoarele:

```
PRINT LEN "Indraznetii intodeauna inving !"
```

si veti primi raspunsul 31 (numarul de caractere al sirului inclusiv spatiile).

Daca intr-o expresie apar si functii si operatii matematice, atunci functiile se vor evalua prima data si apoi se efectueaza operatiile in ordinea prioritatilor. Aceasta regula poate fi incalcata prin utilizarea parantezelor. Va prezentam acum doua exemple care difera prin paranteze si modul in care sint evaluate:

```
LEN "Calea" + LEN " Lactee"
5 + LEN " Lactee"
5 + 7
12
```

si

```
LEN ("Calea" + " Lactee")
LEN ("Calea Lactee")
LEN "Calea Lactee"
12.
```

Functia STR$

Converteste numere in siruri. Deci argumentul este numar, iar rezultatul este sirul format din cifrele numarului. De exemplu:

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

are acelasi efect cu:

```
LET a$="100"
```

Tastati:

```
PRINT LEN STR$ 100.0000
```

si veti obtine raspunsul 3, intrucit STR$ 100.0000 este egal cu 100, a carui lungime este 3.

Functia VAL

Este functia inversa a lui STR$, deci converteste siruri in numere. De exemplu:

```
VAL "3.5"
```

va da numarul 3.5.

Daca luati un numar, si ii aplicati VAL, apoi STR$, nu obtineti intotdeauna sirul initial. Functia VAL poate avea ca argument si expresii numerice, ca de exemplu:

```
VAL "2*3"
```

da valoarea 6. Dar si

```
VAL ("2" + "*3")
```

este egal tot cu 6. Cum se face evaluarea in al doile exemplu? Mai intii expresia dintre paranteze este evaluata la sirul "2*3", apoi se inlatura ghilimelele si se evalueaza ca un numar, rezultind valoarea 6.

Functia VAL$

Este similara cu VAL. Argumentul este sir, iar rezultatul este tot sir. Pentru obtinerea rezultatului, se evalueaza argumentul la un sir, apoi se inlatura ghilimelele si ce ramine este evaluat tot ca un sir.

Astfel:

```
VAL$ ""Michelangelo""
```

este egal cu **Michelangelo**.
Tastati:

```
LET a$="99"
```

si afisati cu instructiunea PRINT urmatoarele: VAL a$, VAL "a$", VAL ""a$"", VAL$ a$, VAL$ "a$", VAL$ ""a$"". Unele expresii vor fi evaluate, altele nu; cautati sa va explicati de ce.

Functia SGN

Este functia semn. Si argumentul si rezultatul sint numere. Rezultatul este 1, daca numarul e pozitiv, 0 daca argumentul este zero, si -1, daca numarul e negativ.

Functia ABS

Argumentul si rezultatul sint numere. Converteste argumentul intr-un numar pozitiv (care este rezultatul), ignorindu-i semnul. De exemplu:

```
ABS -3.2
```

este egal cu:

```
ABS 3.2
```

si de fapt este 3.2.

Functia INT

Si argumentul si rezultatul sint numere. Functia converteste un numar cu zecimale (argumentul) intr-un numar intreg (rezultatul) ignorind zecimalele. Deci INT 3.9 este egal cu 3.

Rotunjirea se face intotdeauna in jos, deci atentie la numerele negative. Astfel:

```
INT -3.1
```

este egal cu -4.

Functia SQR

Argumentul si rezultatul sint numere. Calculeaza radacina patrata a argumentului (deci rezultatul inmultit cu el insusi, da argumentul). De exemplu:

```
SQR 4
```

este egal cu 2 (2*2=4)

```
SQR 0.25
```

este egal cu 0.5.

Argumentul functiei SQR (radical) trebuie sa fie pozitiv, altfel se genereaza mesaj de eroare "A Invalid Argument".

DEF FN si FN

Puteti să va definiti si functiile dvs. proprii. Nume posibile pentru aceste functii sint FN urmate de o litera (daca rezultatul este un numar) sau o litera urmata de semnul $, daca rezultatul este un sir. La aceste functii, argumentul trebuie inchis intre paranteze.

Functiile proprii se definesc cu ajutorul instructiunii DEF. De exemplu, definim o functie FN p a carei rezultat este patratul argumentului:

```
10 DEF FN p(x)=x*x
```

p care urmeaza dupa DEF FN este numele functiei. x dintre paranteze este numele sub care se apeleaza argumentul functiei. Puteti utiliza pentru argument o litera sau o litera urmata de semnul $. Dupa semnul = urmeaza formula de definire a functiei. Aceasta poate sa fie orice expresie si poate referi argumentul utilizindu-i numele (in cazul nostru x) ca la o variabila simpla.

Odata introdusa aceasta linie, **functia poate** fi apelata ca una din functiile standard ale limbajului BASIC pe Tim-S Plus, tastind FN p urmata de argument, intre paranteze. Incercati urmatoarele:

```
PRINT FN p(2)
PRINT FN p(3+4)
PRINT 1 + INT FN p(LEN "cratita"/2 + 3).
```

Odata definita o functie cu DEF FN, se poate apela la fel ca cele standard.

Functia INT rotunjeste intotdeauna in jos. Pentru a obtine primul intreg mai mare se aduna 0.5 si va puteti defini propria functie:

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

Veti obtine de exemplu urmatoarele:

```
FN r(2.9)  este egal cu 3
FN r(2.4)  este egal cu 2
FN r(-2.9) este egal cu -3
FN r(-2.4) este egal cu -2.
```

Comparati aceste rezultate cu cele pe care le obtineti utilizind **INT** in loc de **FN r**. Introduceti si executati urmatorul program:

```
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()
```

Sint citeva lucruri importante de observat in acest program. In primul rind, se observa ca argumentul unei functii, definita cu **DEF FN**, nu este impus la 1. De fapt poate avea maxim 26 argumente numerice si - in acelasi timp - pina la 26 argumente sir.

In al doilea rind nu conteaza unde in program este pusa linia de definire a functiei. Dupa ce calculatorul a executat linia 10, "sare" peste liniile 20 si 30 si executa linia cu numarul 40. Linia de definire a functiei trebuie pusa undeva in program si nu poate fi o comanda.

In al treilea rind, x si y sint atit nume de variabile in program cit si argumente pentru functia FN p. Functia FN p "uita" temporar de variabilele cu numele x si y, dar din moment ce pe a nu il are argument, tine cont de variabila a. Astfel, cind se evalueaza FN p(2,3), x ia valoarea 2, y ia valoarea 3, iar a are valoarea 10. Rezultatul va fi deci 10+2*3, care este egal cu 16. Atunci cind se evalueaza FN q(), deoarece functia nu are argumente, x, y si a se vor lua variabile din program si vor avea valorile 0, 0 si respectiv 10. Rezultatul evaluarii functiei FN q() va fi de data aceasta 10 + 0 * 0, deci 10.

Schimbati acum linia 20 in:

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

De data aceasta FN p(2,3) va avea valoarea 10, deoarece FN q() tine cont de variabilele x si y. Utilizati functia FN p(x)=x*x pentru a testa functia SQR. Veti gasi ca:

```
FN p(SQR x)
```

este egala cu x pentru numere pozitive si

```
SQR FN p(x)
```

este egala cu **ABS** x pentru x pozitiv sau negativ.

4.10 Functii matematice

Din sumar:

ridicare la putere ^
PI, EXP, LN
SIN, COS, TAN, ASN, ACS, ATN

Functiile ^ si EXP

Ridicarea la putere inseamna inmultirea unui numar cu el insusi, de atitea ori cit este puterea lui. De exemplu, puterile

lui doi sint:

```
2^1 este egal cu 2
2^2 esta egal cu 2*2=4
2^3 este egal cu 2*2*2=8
2^4 este egal cu 2*2*2*2=16
```

si asa mai departe.

Deci **a^b** (a la puterea **b**) inseamna **a** inmultit cu el insusi de **b** ori. Se respecta regula:

a^(b+c)=a^b*a^c.

Ridicarea la putere are prioritate mai mare fata de inmultire si impartire, deci in calculul expresiilor se va evalua mai intii ridicarea la putere.
Retineti:

a^0 este egal cu **1**;
a^(-b) este egal cu **1/a^b**;
a^(1/b) este egal cu radical indice **b** din **a**, adica numarul care inmultit cu el insusi de **b** ori da **a**;
a^(b*c) este egal cu **(a^b)^c**.
Doua relatii mai des folosite:
a^(-1) este egal cu **1/a**

si

a^(1/2) este egal cu **SQR a**.

Incercati urmatorul program:

```
10 INPUT a, b, c
20 PRINT a^(b+c),a^b*a^c
30 GO TO 10
```

Veti observa bineinteles ca cele doua numere ce vor fi afisate sint egale.

Pentru +3, la ridicarea la putere (a^b) numarul **a** trebuia sa fie pozitiv.

Sa luam acum urmatorul exemplu: sa presupunem ca ati depus la CEC suma de 100 lei, iar dobinda anuala pe care o primiti este de 5%. In acest caz, dupa un an veti avea 100 lei plus dobinda de 5%, deci 105% fata de cit ati avut initial. Altfel spus, ati inmultit suma depusa initial cu 1.05. Dupa inca un an se va intimpla acelasi lucru, deci veti avea de 1.05*1.05 sau 1.05^2 sau de 1.1025 ori suma initiala de bani. In cazul general, dupa n ani veti avea de 1.05^n ori suma care ati depus-o initial.

Incercati urmatoarea comanda:

```
FOR n=0 TO 100: PRINT n,100*1.05^n:NEXT n
```

si veti vedea cum creste suma de 100 lei dupa n ani, cu dobinda de 5%. Aceasta crestere cu o cantitate constanta in timp se numeste crestere exponentiala si se calculeaza pin ridicarea unui numar fix la o putere dependenta de timp.

Sa presupunem ca ati introdus:

```
10 DEF FN a(x)=a^x
```

unde **a** este stabilit printr-o instructiune LET. Daca **a** are

valoarea matematica e, se obtine functia standard EXP implementata pe +3. Deci EXP este egal cu e^x. Daca doriti sa aflati valoarea lui e (limitata din cauza zecimalelor) tastati:

PRINT EXP 1

(intrucit e^1=e).

Functia LN

Functia logaritm este inversa functiei exponentiale. Logaritmul in baza a al unui numar, este puterea la care ar trebui ridicat a pentru a-l obtine pe x. Logaritmii pot avea orice baza; cei in baza e se numesc logaritmi naturali. Functia LN calculeaza logaritmul natural al argumentului. Pentru a obtine logaritmul in alta baza, de exemplu baza a folositi formula LNx/LNa.

Functia PI

Da numarul PI, care incepe ca valoare cu 3.1415927, valoare care o puteti afla cu

PRINT PI.

Se foloseste in calcule trigonometrice. De exemplu, perimetrul (circumferinta) unui cerc este egal cu 2*PI*r, unde r este raza cercului.

Functiile SIN, COS, TAN, ASN, ACS, ATN

Functia SIN da sinusul unui unghi exprimat in radiani. Functia COS calculeaza cosinusul unui unghi exprimat in radiani.

Retineti ca ambele functii au periodicitate de 2*PI, deci **SIN a** este egal cu SIN (a+2*PI) si COS a=COS(a+2*PI).

Functia TAN calculeaza tangenta, cu definitia **SIN a/COS a.** Uneori avem nevoie de valoarea unghiului, cunoscindu-i sinusul, cosinusul sau tangenta si atunci se folosesc functiile inverse: arcsinus (ASN), arccosinus (ACS) si arctangenta (ATN).

Functia cotangenta se calculeaza cu formula 1/TAN a. Nu uitati ca functiile trigonometrice SIN, COS etc. au nevoie de argument in radiani. Dam mai jos corespondenta dintre grade si radiani:

PI/6 - 30grade	3PI/4 - 135grade
PI/4 - 45grade	5PI/6 - 150grade
PI/3 - 60grade	PI - 180grade
PI/2 - 90grade	2PI - 360grade
4PI/6 - 120grade	

Pentru a transforma din grade in radiani, impartiti cu 180 grade si inmultiti cu PI, iar pentru a transforma din radiani in grade impartiti cu PI si inmultiti cu 180.

4.11 Numere aleatoare

Din sumar:
RANDOMIZE, RND

RND este si ea o functie (calculeaza si produce un rezultat) dar nu are argument. La fiecare apelare va genera un numar alea-

tor intre 0 si 1 (uneori poate fi 0, dar niciodata 1).
Incercati urmatoarele:

```
10 PRINT RND
20 GO TO 10
```

In realitate numerele generate cu RND nu sint aleatoare ci pseudoaleatoare, intrucit se genereaza cu aceeasi secventa fixa de 65536 numere.
RND genereaza numere aleatoare cuprinse in intervalul 0...1. Daca doriti numere aleatoare generate in alt interval, de exemplu a...b utilizati formula a+RND*b sau daca doriti numai numere intregi a+INT(RND*b).
Introduceti si executati urmatorul program:

```
10 REM generator zar
20 CLS
30 FOR n=1 TO 2
40 PRINT 1+INT(RND*6);" ";
50 NEXT n
60 INPUT a$: GO TO 20
```

Tastati ENTER de fiecare data cind doriti sa apara zarul.
Instructiunea RANDOMIZE se foloseste pentru a defini un punct de plecare pentru RND in secventa de generare a numerelor aleatoare. Incercati urmatorul program:

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

Dupa fiecare executie a liniei 10 (RANDOMIZE 1), secventa de generare numere aleatoare cu RND incepe cu 0.0022735596. Puteti utiliza si alte numere pentru RANDOMIZE, cuprinse intre 1 si 65536 si RND va incepe secventa de generare cu numere diferite.
Daca aveti un program in care utilizati RND si aveti greseli in program, atunci va va ajuta folosirea lui RANDOMIZE, intrucit programul se va comporta la fel de fiecare data.
RANDOMIZE fara argument (sau RANDOMIZE 0) utilizeaza timpul de cind +3 a fost pus in functiune. Deci secventa urmatoare:

```
10 RANDOMIZE
20 PRINT RND: GO TO 10
```

nu este "prea aleatoare". Obtineti rezultate mai bune daca inlocuiti GO TO 10 cu GO TO 20.
Testati urmatorul program:

```
10 LET fata=0 : LET stema=0
20 LET moneda=INT (RND*3)
30 IF moneda = 0 THEN LET fata = fata+1
40 IF moneda = 1 THEN LET stema = stema+1
50 PRINT fata; ",";stema,
60 IF stema <>0 THEN PRINT fata/stema;
70 PRINT : GO TO 20
```

Raportul fata/stema ar tebui sa fie aproximativ 1, daca lasati programul sa se execute un timp destul de lung.
Alegeti un numar intre 1 si 872 si apoi tastati:

```
RANDOMIZE n (n fiind numarul dvs.).
```

Verificati ca prima valoare pentru RND va fi (75*(n+1)-1)/65536.

4.12 Tablouri

Din sumar:
Siruri
DIM

Sa presupunem ca aveti o lista de numere, de exemplu notele a 10 copii dintr-o clasa, la o materie. Pentru a le memora in calculator ar trebui sa utilizati 10 variabile n1, n2, ..., n10, iar programul pentru initializarea acestor variabile ar fi lung si plictisitor:

```
10 LET n1=75
20 LET n2=80
30 LET n3=44
40 LET n4=38
50 LET n5=55
60 LET n6=64
70 LET n7=70
80 LET n8=64
90 LET n9=58
100 LET n10=60.
```

In loc de aceasta, se utilizeaza asa numitele tablouri, la care se specifica o variabila care (in loc sa aiba o singura valoare ca o variabila normala) poate contine un numar de elemente, fiecare cu alta valoare. Fiecare element se refera printr-un numar de index (indice) scris intre paranteze dupa numele variabilei. Pentru exemplul de mai jos tabloul variabilei n ar putea fi n (numele unui tablou de variabile trebuie sa fie o singura litera), iar cele 10 variabile vor fi referite cu n(1), n(2),..., n(10). Elementele unui tablou se numesc variabile indexate si sint diferite de variabilele simple.

Inainte de utilizarea unui tablou trebuie sa-i rezervati spatiu in memoria calculatorului si aceasta se face cu instructiunea DIM (declararea dimensiunii). Astfel:

```
DIM n(10)
```

rezerva spatiu in memorie pentru un tablou cu numele n, de dimensiune 10 (sint 10 variabile). Instructiunea DIM initializeaza cele 10 elemente ale tabloului cu zero. De asemenea sterge orice alt tablou cu numele n, care a existat anterior (nu sterge variabila simpla n, daca ea a existat; o variabila tablou poate coexista in acelasi timp cu o variabila simpla cu acelasi nume).

Indicele elementelor din tablou poate fi repetat de orice expresie numerica din care rezulta un numar de indice valid. Astfel, programul anterior poate fi scris in urmatoarea varianta:

```
10 DIM n(10)
20 FOR i=1 TO 10
30 READ n(i)
40 NEXT i
50 DATA 75,44,90,38,55,64,70,12,75,60
```

in care valorile se citesc din lista din instructiunea DATA, sau:

```
10 DIM n(1)
20 FOR i=1 TO 10
30 INPUT n(i)
40 NEXT i
```

unde valorile se introduc de la tastatura.

Instructiunea DIM, de declarare a unui tablou, poate sa apara oriunde in program, dar neaparat inainte de referirea (utilizarea) tabloului respectiv.

Valorile elementelor unui tablou se pot afla cu:

```
10 FOR i=1 TO 10
20 PRINT n(i)
30 NEXT i
```

sau pe rind

```
PRINT n(1)
PRINT n(2)
PRINT n(3)
```

s.a.m.d.

Se pot utiliza si tablouri cu mai mult de o dimensiune. Pentru un tablou cu doua dimensiuni aveti nevoie de doua numere pentru a referi un element al tabloului (ca de exemplu linia si coloana unui caracter afisat pe ecran). Daca va imaginati numar de linie si coloana pentru doua dimensiuni, dintr-o pagina de exemplu, a treia dimensiune ar putea fi reprezentata de numarul paginii. Deci un asemenea tablou ar avea ca indici (numar pagina, numar linie, numar coloana).

Pentru declararea unui tablou bidimensional, de dimesiuni 3 si 6, veti scrie:

```
DIM c(3,6)
```

Tabloul va avea 18 variabile indexate:

```
c(1,1) c(1,2) ... c(1,6)
c(2,1) c(2,2) ... c(2,6)
c(3,1) c(3,2) ... c(3,6)
```

Acelasi principiu se respecta pentru oricite dimensiuni. Asa cum am precizat, pot exista in acelasi program o variabila simpla si o variabila tablou, dar nu pot coexista doua tablouri cu acelasi nume, dar de dimensiuni diferite.

Se pot utiliza si tablouri de siruri de caractere. Sirurile din tablou difera de sirurile simple prin aceea ca sint de lungime fixa, iar asignarea lor este intotdeauna de tip Procust (se taie din caractere sau se adauga spatii).

Numele unui tablou sir este format dintr-o litera urmata de semnul $. Ca diferenta fata de tablourile numerice, un tablou sir nu poate coexista in acelasi program cu un sir simplu cu acelasi nume.

Sa presupunem ca doriti un tablou sir de cinci siruri. Trebuie sa va hotariti ce lungime trebuie sa aiba aceste cinci siruri; sa presupunem ca 10 caractere pentru fiecare sir este suficient. Veti declara:

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

Prin aceasta se rezerva spatiu pentru un sir de 5*10 caractere, dar se poate privi si astfel: fiecare rind ca un sir.

a$(1) este egal cu a$(1,1),a$(1,2),...,a$(1,10)
a$(2) este egal cu a$(2,1),a$(2,2),...,a$(2,10)
a$(3) este egal cu a$(3,1),a$(3,2),...,a$(3,10)
a$(4) este egal cu a$(4,1),a$(4,2),...,a$(4,10)
a$(5) este egal cu a$(5,1),a$(5,2),...,a$(5,10).

Daca la referire utilizati doi indici veti obtine un singur caracter, iar daca utilizati un singur numar, obtineti un sir de lungime fixa.
Astfel tastati:

```
LET a$(2)="1234567890"
```

si apoi:

```
PRINT a$(2),a$(2,7)
```

Veti obtine:

```
1234567890          7
```

Pentru ultimul indice (care poate lipsi) se poate utiliza si forma:

a$(2,4 TO 8) este egal cu a$(2) (4 TO 8) si este egal cu "45678".

Indicele unei variabile tablou poate fi o variabila, o expresie numerica sau un numar, dar trebuie sa se incadreze in dimensiunea declarata in instructiunea DIM.

Pentru DIM a$(10), a$ se comporta ca o variabila sir simpla, dar care are intotdeauna dimensiunea 10, iar asignarea lui este de tip Procust.

Utilizati instructiunile READ si DATA pentru a initializa un tablou sir 1$ de 12 siruri in care fiecare 1$(i) este numele unei luni din an. Instructiunea DIM in acest caz va fi DIM 1$(12,9). Scrieti dvs. programul de initializare si afisare a celor 12 siruri.

4.13 Conditii

Din sumar:
AND, OR, NOT

Am vazut ca forma generala a instructiunii IF este:

IF conditie THEN

Conditiile prezentate pina acum au fost constituite din expresii relationale (cu operatorii relationali =, <, >, <=, >=, <>, care compara doua numere sau siruri). Conditiile pot fi constituite si din expresii logice, care utilizeaza operatorii logici **AND, OR, NOT.**

O expresie AND alta expresie, este adevarata daca ambele expresii sint adevarate, ca de exemplu:

```
IF a$="DA" AND x>0 THEN PRINT "rezultat"
```

in care **rezultat** va fi afisat numai daca a$ este egal cu DA si daca x>0.

O expresie OR alta expresie, este adevarata atunci cind cel putin una din expresii este adevarata (deci daca sau numai una sau amindoua expresiile sint adevarate).

Operatorul NOT neaga expresia. **NOT expresie,** este adevarata atunci cind expresia este falsa, si este falsa atunci cind expresia este adevarata.

Expresiile logice pot utiliza combinatii cu operatii AND, OR si NOT, la fel cum expresiile numerice pot utiliza operatiile aritmetice +, -, *, /. Daca este necesar, se pot utiliza si paranteze. Operatiile logice au prioritati la fel ca si operatiile aritmetice. NOT are cea mai mare prioritate, apoi AND si OR.

NOT este ca o functie cu un argument si un rezultat, dar fata de celelalte functii are prioritate mai mica. Argumentul ei nu necesita paranteze, numai in cazul in care contine operatori AND sau OR (sau amindoi).

NOT a=b inseamna NOT (a=b) (si are acelasi rezultat cu a<>b).

<> este negatia lui =, cu sensul ca este adevarat doar daca = este fals. Cu alte cuvinte:

a<>b este acelasi cu NOT a=b

si de asemenea

NOT a<>b este acelasi cu a=b

Va puteti convinge si dvs. ca >= si <= sint negatiile pentru < si respectiv >. Astfel puteti elimina NOT punind operatorul relational complementat in loc.

De asemenea:

NOT (expresia 1 AND expresia 2)

este echivalent cu:

NOT (expresia 1) OR NOT (expresia 2)

iar expresia:

NOT (expresia 1 OR expresia 2)

este echivalenta cu:

NOT (expresia 1) AND NOT (expresia 2)

Logic vorbind NOT nu este necesar, dar uneori ajuta la claritatea programelor.

Incercati:

```
PRINT 1=2,1<>2
```

si v-ati astepta la eroare de sintaxa.

Nu se intimpla insa asa, deoarece calculatorul pentru valori logice utilizeaza numere simple. In asemenea cazuri sint respectate urmatoarele reguli:

1. =, <, >, <=, >=, <> dau rezultate numerice: 1 pentru adevarat si 0 pentru fals. Astfel comanda PRINT de mai sus va afisa 0 pentru 1=2 (expresie falsa) si 1 pentru 1<>2 (expresie adevarata).

2. In instructiunea :

```
IF conditie THEN
```

conditia poate fi o expresie numerica. Daca valoarea ei este 0, atunci va fi luata in considerare ca falsa si oricare alta valoare (inclusiv 1) va fi luata in considerare ca adevarata. Deci instructiunea IF de mai sus inseamna acelasi lucru ca:

```
IF conditie <>0 THEN
```

3. AND, OR, NOT sint de asemenea operatii care sint echivalente cu numere.

x AND y are valoarea: x, daca y este adevarat (diferit de zero);
0 (fals), daca y este fals (zero).

x OR y are valoarea: 1(adevarat), daca x este adevarat (diferit de zero);
x, daca y este fals (zero).

NOT x are valoarea: 0(fals), daca x este adevarat (diferit de zero);
1(adevarat), daca x este fals (zero).

Retineti ca "adevarat" inseamna diferit de zero cind se testeaza o valoare dar inseamna 1 cind se genereaza una noua.

Incercati acest program:

```
10 INPUT a
20 INPUT b
30 PRINT (a AND a>=b)+(b AND a<b)
40 GO TO 10
```

De fiecare data va tipari cel mai mare numar dintre a si b.

O expresie care utilizeaza OR poate arata cam asa:

```
LET total=pret*(1.05 OR v$="rata zero")
```

Se pot construi si expresii cu siruri care utilizeaza operatorii AND, OR, NOT.

x$ AND y are valoarea: x$ daca y este diferit de zero "" (sirul nul), daca y este zero.

Incercati urmatorul program care introduce doua siruri si le pune in ordine alfabetica:

```
10 INPUT "Introduceti doua siruri",a$,b$
20 IF a$>b$ THEN LET c$=a$: LET a$=b$: LET b$=c$
30 PRINT a$;" ";("<"AND a$<b$)+("=" AND a$=b$);" ";b$
40 GO TO 10
```

4.14 Setul de caractere

Din sumar:
CODE, CHR$
POKE, PEEK
USR, BIN

Literele, cifrele, semnele de punctuatie s.a.m.d. care pot apare in siruri sint denumite caractere si formeaza setul de caractere al calculatorului. Multe din acestea sint simboluri, dar mai sint si altele, denumite "simboluri compuse", care reprezinta cuvinte cheie ale limbajului BASIC, ca: PRINT, STOP, <>, etc.

Sint 256 caractere si fiecare are un cod cuprins intre 0 si 255 (sint listate complet in sectiunea 4.29 a acestui capitol). Pentru a face conversii intre coduri si caractere sint disponibile doua functii si anume CODE si CHR$.

Functia CODE se aplica unui sir si da codul primului caracter din sir (sau 0 daca sirul este nul).

Functia CHR$ se aplica unui numar si da caracterul al carui cod este numarul respectiv.

Urmatorul program afiseaza intregul set de caractere:

```
10 FOR a=32 to 255 : PRINT CHR$ a : NEXT a
```

Dupa cum vedeti, setul de caractere consta dintr-un spatiu, 15 simboluri si semne de punctuatie, cele 10 cifre, inca 7 simboluri, literele mari, inca 6 simboluri, literele mici si inca 5 simboluri. Acestea sint toate (exceptie fac ` si @) luate din setul de caractere standard ASCII (American Standard Code for Information Interchange). ASCII asigneaza si coduri numerice acestor caractere, coduri utilizate si de +3.

Restul caracterelor nu fac parte din codul ASCII, dar sint dedicate familiei de calculatoare TIM. La inceput ele cuprind un spatiu si 15 combinatii de alb si negru. Acestea sint denumite caractere semigrafice si se utilizeaza la construirea figurilor grafice de pe ecran. Aceste caractere semigrafice se pot introduce de la tastatura, utilizind modul grafic "G". In modul grafic se intra tastind GRAPH. Apoi tastind cifrele 1,2,3,4,5,6,7 si 8 obtinem simbolurile semigrafice. Tot in modul grafic tastind CAPS SHIFT si una din tastele de la 1 la 8 se obtin versiunile inversate ale simbolurilor semigrafice anterioare (negrul devine alb si albul devine negru).

Apasind tasta 9 se revine la normal (se apasa din nou GRAPH). Tasta 0 sterge caracterul din stinga cursorului.

Dupa caracterele semigrafice urmeaza o copie a alfabetului de la A la S. Aceste caractere pot fi redefinite de catre utilizator (cind se pune in functiune calculatorul ele sint initializate ca litere) si sint numite - caractere grafice definite de utilizator (u.d.g.-de la "user defined graphics"). Le puteti introduce de la tastatura, din modul grafic, tastind litere de la A la S.

Pentru a defini un nou caracter grafic, urmati urmatorul exemplu, care defineste caracterul PI:

1. Fiecare caracter are 8*8 puncte, fiecare punct putind fi scris sau nu. Pe matricea de mai jos se deseneaza caracterul PI (forma pentru caracterul PI) astfel (unde este 1 se considera un punct scris si unde este 0, se considera un punct nescris):

```
Linia 1     00000000
```

2	00000000
3	00000010
4	00111100
5	01010100
6	00010100
7	00010100
8	00000000

Un punct scris (1) inseamna ca are culoarea cernelii (INK), iar un punct nescris (0) inseamna ca are culoarea hirtiei (PAPER). Lucrul cu culorile va fi explicat in sectiunea 4.16 a acestui capitol.

2. Alegeti caracterul din alfabet pentru care doriti sa se afiseze caracterul PI ...Sa zicem P (deci atunci cind tastati P, va apare semnul PI).

3. Memorati forma aleasa. Fiecare caracter definit de utilizator este memorat ca 8 numere, cite unul pentru fiecare rind. Puteti scrie aceste numere in program BIN urmat de 8 cifre (pentru fiecare numar) de 0 sau 1.

Instructiunile pentru caracterul PI vor fi:

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

Aceste valori sint de fapt reprezentarea in binar a cite unui numar zecimal.

Aceste 8 numere se memoreaza in 8 locatii de memorie succesive. Fiecare locatie are cite o adresa. Adresa primului octet (primei locatii) este USR "P" (deoarece la punctul 2 am ales caracterul P). Adresa urmatorului octet este USR "P"+1 s.a.m.d. pentru fiecare din cei 8 octeti, ultimul avind adresa USR "P"+7.

USR aici este o functie care converteste un argument sir in adresa primului octet din caracterul definit. Sirul argument trebuie sa fie un singur caracter, care poate fi el insusi definit de utilizator sau litera corespunzatoare. Exista si o utilizare a functiei USR pentru argument numeric, dar acest aspect il vom discuta mai tirziu.

Urmatorul program defineste caracterul:

```
10 FOR n=0 TO 7
20 READ rind:POKE USR "P"+n,rind
30 NEXT n
40 DATA BIN 00000000
50 DATA BIN 00000000
60 DATA BIN 00000010
70 DATA BIN 00111100
80 DATA BIN 01010100
90 DATA BIN 00010100
100 DATA BIN 00010100
110 DATA BIN 00000000
```

Instructiunea POKE memoreaza un numar direct in locatia de memorie specificata prin adresa. Instructiunea inversa pentru POKE este PEEK, care permite vizualizarea locatiilor de memorie fara a le modifica continutul. Instructiunile PEEK si POKE sint

descrise mai amanuntit in sectiunea 4.24.

Dupa caracterele grafice, definite de utilizator, in setul de caractere urmeaza simbolurile compuse (cuvintele rezervate ale limbajului BASIC).

Ati observat in primul program din acest capitol, ca nu am afisat primele 32 caractere (de la 0 la 31); acestea sint caractere de control si sint utilizate pentru controlul afisarii unor functii din +3. Daca incercati sa afisati un caracter de control, calculatorul va va raspunde cu semnul intrebarii "?" pentru a arata ca nu intelege. Caracterele de control sint descrise mai pe larg in sectiunea 4.29.

Cele trei caractere de control pe care le utilizeaza afisarea pe ecran sint 6, 8 si 13.

CHR$ 6 este interpretata drept tiparire spatii (blancuri) in acelasi mod in care este interpretata virgula pusa intr-o instructiune PRINT; de exemplu:

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

face acelasi lucru ca si:

```
PRINT 1,2
```

O alta metoda de utilizare este:

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

CHR$ 8 este "spatiu inapoi" - muta pozitia de afisare (tiparire) cu o pozitie inapoi. Incercati:

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

care va tipari:

1235

CHR$ 13 este "NEW LINE" - muta pozitia de afisare la inceputul liniei urmatoare.

Caracterul cu codul 16 este explicat in sectiunea 4.29. Caracterul cu codul 23 reprezinta control de tabulare (caracterul de control TAB; argumentul lui reprezinta numarul de spatii care se vor tipari pe linie).

Incercati sa priviti setul de caractere ca un alfabet care in loc de 26 litere are 256 caractere in aceeasi ordine in care sint codurile lor. Ordonarea alfabetica la +3 se refera la "alfabetul" mare, la setul de caractere. De exemplu, urmatoarele siruri sint in ordine alfabetica pentru +3 (observati ca literele mici vin dupa literele mari, astfel a vine dupa Z; observati si ca spatiile sint semnificative):

```
CHR$ 3+"GRADINA ZOOLOGICA"
CHR$ 8+"INAINTE"
" ABCDE"
"(Indraznetii intodeauna inving !)"
"300"
"300.000 km/s"
"AA BB CC"
"Aa Bb Cc"
"Leaganul vietii in sistemul solar este planeta Pamint"
"PRINT"
"Zoo"
```

```
"[Observatorul din Puerto Rico, Arecibo]"
"acasa"
"asteptare"
"calculator"
"fizic"
"fizica"
```

Regula pentru a determina ordinea in care sint comparate doua siruri este urmatoarea: se incepe cu compararea primelor doua caractere. Daca sint diferite, atunci unul are codul mai mic (vine inainte) decit celalalt. Daca primele doua caractere sint la fel, atunci se continua compararea urmatoarelor doua caractere. Daca in acest proces, un sir se termina inaintea celuilalt, atunci acesta vine inainte.

Relatiile =, <, >, <=, >=, <> sint utilizate pentru siruri la fel ca pentru numere; < inseamna "vine inainte", > inseamna "vine dupa", astfel relatiile:

```
"AA BB "<"AABBCCDD"
"AABBCCDD">"AA BB "
```

sint ambele adevarate.

<= si >= sint evaluate ca si la numere, astfel ca:

```
"Acelasi sir"<="Acelasi sir"
```

este adevarata, dar:

```
"Acelasi sir"<"Acelasi sir"
```

este falsa.

Faceti testele pe urmatorul program, care introduce doua siruri si le pune in ordine.

```
10 INPUT "Introduceti doua siruri:",a$,b$
20 IF a$>b$ THEN LET c$=a$ : LET a$=b$ : LET b$=c$
30 PRINT a$;" ";
40 IF a$<b$ THEN PRINT "<";:GO TO 60
50 PRINT "="
60 PRINT " ";b$
70 GO TO 10
```

Observati ca in linia 20 am introdus un nou sir c$ cind am schimbat sirurile a$ si b$ intre ele. Explicati de ce nu se poate folosi simplu:

```
LET a$=b$ : LET b$=a$
```

Urmatorul program defineste caracterele grafice astfel incit urmatoarele taste sa afiseze piese de sah:

```
C - pentru cal
K - pentru rege
N - pentru nebun
P - pentru pion
R - pentru regina
T - pentru tura
```

```
  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 8 octeti pentru fiecare piesa
30 READ a : POKE USR p$+f,a
40 NEXT f
50 NEXT n
100 REM nebun
110 DATA "n",0,d,BIN 00101000,BIN 01000100
120 DATA BIN 01101100,c,b,0
130 REM rege
140 DATA "k",0,d,c,d
150 DATA c,BIN 01000100,c,0
160 REM tura
170 DATA "t",0,BIN 01010100,b,c
180 DATA c,b,b,0
190 REM regina
200 DATA "r",0,BIN 01010100, BIN 00101000,d
210 DATA BIN 01101100,b,b,o
220 REM pion
230 DATA "p",0,0,d,c
240 DATA c,d,b,0
250 REM cal
260 DATA "c",0,d,c,BIN 01111000
270 DATA BIN 00011000,c,b,0
```

Observati ca in instructiunea DATA am pus simplu 0 in loc de BIN 00000000. Cind executati programul, puteti vedea piesele, tastind GRAPH si apoi tastele C,K,N,P,R,T.

Executati urmatorul program:

```
10 INPUT c
20 PRINT CHR$ c;
30 GO TO 10
```

Veti observa ca functia **CHR$ c** rotunjeste la cel mai apropiat numar intreg; daca c nu este cuprins intre 0 si 255, atunci programul se opreste cu eroare **B integer out of range.**

4.15 Mai mult despre PRINT si INPUT

Din sumar:

CLS
TAB, AT
LINE
SCREEN$

Ati mai intilnit instructiunea PRINT si pina acum si stiti cum se utilizeaza. Expresiile care apar in instuctiunea PRINT pot sa fie despartite prin virgula, punct si virgula sau apostrof, simboluri care aici au rol separatori.

In instructiune se mai pot folosi si cuvintele cheie AT si TAB. Sa luam ca exemplu urmatoarea instructiune:

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

care va afisa un asterisc la mijlocul ecranului. Aceasta deoarece dupa AT se pune numarul liniei, numarul coloanei, unde vreti sa se faca afisarea.

Ecranul are 22 linii (de la 0 la 21, incepind de sus) si 32 coloane (de la 0 la 31, de la stinga la dreapta).

Functia SCREEN$ este inversa functiei PRINT AT si va citi caracterul care se afla la o anumita pozitie pe ecran. Utilizeaza

numere de linie si coloana la fel ca si AT. De exemplu, instrucțiunea:

```
20 PRINT AT 0,0; SCREEN$ (11,16)
```

va citi asteriscul afisat la mijlocul ecranului si il va afisa din nou in coltul din stinga sus al ecranului.

Caracterele din simbolurile compuse sint citite normal (ca si caractere simple) si spatiile sint citite ca spatii. La incercarea de a citi caractere grafice definite de utilizator sau linii trasate cu DRAW, PLOT sau CIRCLE va rezulta sirul nul. La fel si daca s-a utilizat OVER pentru a crea un caracter compus. (Cuvintele cheie PLOT, DRAW, CIRCLE si OVER vor fi descrise mai tirziu).

Functia **TAB coloana** afiseaza suficiente spatii pentru a muta pozitia de PRINT in coloana specificata. Afisarea se face pe aceeasi linie sau daca sint necesare spatii inapoi trece la linia urmatoare. Calculatorul reduce numarul de coloana modulo 32 (imparte cu 32 si retine restul) astfel ca TAB 33 inseamna TAB 1.

Sa luam urmatorul exemplu:

```
PRINT TAB 30;1;TAB12;"cuprins"; AT 3,1;"capitol"; TAB 24;
"Pagina"
```

Acesta ilustreaza cum se construieste un cap de tabel.

Incercati urmatorul program:

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

Aici puteti vedea ce inseamna modulo 32.

Retineti urmatoarele:

1. Dupa TAB si expresiile din instructiunea PRINT este cel mai bine sa se puna punct si virgula, asa cum am facut mai sus. Puteti utiliza virgula sau nimic, dar aceasta inseamna ca dupa ce ati definit o pozitie pentru PRINT, o schimbati imediat.

2. Ecranul are de fapt 24 linii, de la 0 la 23, dar pe ultimele doua (22 si 23) nu puteti afisa intrucit ele sint rezervate pentru comenzi, date pentru INPUT, mesaje de eroare, etc. Ultima linie a ecranului inseamna linia 21.

3. Puteti utiliza AT pentru a localiza pozitia pentru PRINT, chiar daca in acel loc exista deja afisat ceva, noul PRINT va scrie peste cel vechi.

Alta instructiune legata de PRINT este CLS. CLS curata (sterge) ecranul.

Cind afisarea ajunge la ultima linie de jos, ecranul face "scroll" in sus (ca la masina de scris). Alegeti din domeniul de editare optiunea Screen si apoi introduceti:

```
CLS: FOR n=1 TO 30: PRINT n : NEXT n
```

Cind s-a umplut ecranul, Tim-S Plus afiseaza mesajul 'scroll?' in partea de jos a ecranului. Daca tastati Y se va afisa un nou ecran cu numere. Acelasi efect il are orice tasta cu exceptia urmatoarelor N, BREAK sau spatiu care opresc actiunea de "scroll" cu mesajul **D BREAK-CONT repeats.**

Ati vazut deja instructiunea INPUT de forma:

```
INPUT "Citi ani aveti?",virsta
```

la care calculatorul va afisa "Citi ani aveti" in partea de jos a ecranului si apoi asteapta sa introduceti valoarea pentru variabila **virsta**. De fapt in instructiunea INPUT pot apare mai multe expresii aproximativ la fel ca la PRINT. Se ia in considerare urmatoarea regula: daca o expresie (un articol) din INPUT incepe cu o litera, atunci ea trebuie sa fie o variabila a carei valoare se introduce. S-ar parea deci ca intr-un mesaj din INPUT nu puteti afisa valoarea unei variabile. Aceasta este posibil insa punind paranteze. Orice expresie care incepe cu o litera, trebuie pusa intre paranteze, daca se doreste sa fie afisata ca parte din mesajul de INPUT.

Orice expresie din PRINT care nu este afectata de aceste reguli, poate constitui expresie si pentru INPUT. Dam un exemplu pentru a ilustra cele de mai sus:

```
LET eu = INT (RND*100):INPUT ("EU am";eu;"ani");"Citi ani
aveti? "; virsta
```

variabila **eu** este intre paranteze deci valoarea ei se va afisa, iar **virsta**, intrucit nu este intre paranteze, trebuie sa i se introduca valoarea.

Tot ce se scrie la o instructiune INPUT, se afiseaza in partea de jos a ecranului, care este intr-un fel independenta de partea de sus a ecranului. Practic liniile din partea de jos sint numarate relativ la liniile din partea de sus, chiar daca partea de jos a ecranului s-a extins si peste liniile partii de sus (care au fost necesare daca s-au tiparit date in INPUT). Cind programul se opreste si incepeti editarea, partea de jos a ecranului revine ca dimensiune la cele doua linii din partea de jos a ecranului. Pentru a vedea cum actioneaza AT in instructiunea INPUT, incercati urmatorul exemplu:

```
10 INPUT "Aceasta este linia 1",a$:AT 0,0;"Aceasta este
linia 0",a$;AT 2,0;"Aceasta este linia 2",a$;AT 1,0;
"Aceasta este tot linia 1",a$.
```

Executati programul (tastati ENTER cind se opreste).

Incercati acum urmatorul exemplu:

```
10 FOR n=0 TO 19: PRINT AT n,0;n;: NEXT n
20 INPUT AT 0,0;a$;AT 1,0;a$;AT 2,0;a$;AT 3,0;a$;4,0;a$;AT
   5,0;a$;
```

Alta functie care poate fi prezenta in instructiunea INPUT este LINE si este folosita la introducerea variabilelor sir. Daca utilizati LINE inainte de numele unei variabile sir ce trebuie introdusa, ca de exemplu:

```
INPUT LINE a$
```

atunci calculatorul nu va mai afisa ghilimelele, asa cum se intimpla in mod normal pentru o variabila sir. Deci, daca introduceti:

```
Perseu
```

variabila a$ va avea valoarea **Perseu**. Retineti ca LINE se utilizeaza numai cu variabile sir.

Incercati sa tastati la INPUT cuvinte cheie. Executati acest program, daca sinteti interesat:

```
10 INPUT numar
20 PRINT numar
30 GO TO 10
```

si la o introducere, tastati EXTENDED MODE urmat de tasta M. Va apare cuvintul PI, iar daca apasati ENTER se va afisa valoarea 3.1415927. Daca introduceti separat cele doua litere P si I, calculatorul se va opri cu mesajul de eroare **2 Variable not found, 10:1**. Aceasta se poate intimpla cind apasati combinatii de date in INPUT.

Caracterele de control CHR$ 22 si CHR$ 23 au efect ca AT si TAB. Cind i se spune calculatorului sa tipareasca unul dintre ele, caracterul trebuie sa fie urmat de inca doua caractere care nu au semnificatia lor normala, dar sint tratate ca numere (codurile lor) pentru a specifica linia si coloana pentru AT, sau pozitia TAB. Aproape intotdeauna este mai usor sa utilizati AT si TAB in mod normal decit sa utilizati caracterele de control; ele pot fi mai utile in anumite situatii speciale. AT este CHR$ 22. Primul caracter dupa el specifica linia, iar al doilea coloana, astfel ca:

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

are acelasi efect cu:

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

TAB este caracterul CHR$ 23 si cele doua caractere de dupa el se combina pentru a forma un numar intre 0 si 65535. Instructiunea:

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

are acelasi efect cu:

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

Puteti opri calculatorul sa va intrebe "scroll?" dupa afisarea unui ecran de 22 linii prin:

```
POKE 23692,255
```

Dupa aceasta, calculatorul va face scroll de 255 ori inainte de a afisa mesajul "scroll?".
Ca exemplu, incercati:

```
10 FOR n=0 TO 1000
20 PRINT n : POKE 23692,255
30 NEXT n
```

Mai incercati - ca exercitiu - si urmatorul program, pentru a testa la copii tabla inmultirii:

```
10 LET m$=""
20 LET a=INT (RND * 12)+1: LET b=INT (RND * 12)+1
30 INPUT (m$)' '"cit face ";(a);" * ";(b);"? ";c
100 IF c=a*b THEN LET m$="Corect.": GO TO 20
110 LET m$="Gresit. Mai insista." : GO TO 30
```

Puteti sa-l si pacaliti pe calculator. Astfel, de exemplu, la intrebarea: **cit face 2*3**, puteti introduce **2*3**. Aveti grija insa, fiindca asemenea satisfactii le sint favorabile numai celor

initiati.

4.16 Culori

Din sumar:

INK, PAPER, FLASH, BRIGHT
INVERSE, OVER, BORDER

Executati urmatorul program:

```
10 FOR m=0 TO 1 : BRIGHT m
20 FOR n=1 TO 10
30 FOR c=0 TO 7
40 PAPER c : PRINT "    ";:REM 4 spatii colorate
50 NEXT c : NEXT n : NEXT m
60 FOR m=0 TO 1 : BRIGHT m : 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 : NEXT m
130 PAPER 7 : INK 0 : BRIGHT 0
```

Cu acest program puteti vedea cele 8 culori si cele doua nivele de stralucire pe care Tim-S Plus le poate produce pentru o culoare. (Daca televizorul dvs. este alb/negru, atunci veti vedea diferite nuante de gri). O metoda rapida de a obtine acelasi rezultat, este de a apasa butonul de RESET, urmat de tasta 3 (incarcare soft de +3), dupa care se apasa tasta BREAK (la cel mult 5 s dupa eliberarea tastei 3). Dam jos o lista cu numerele ce reprezinta culorile:

0 - negru	4 - verde
1 - albastru	5 - azuriu
2 - rosu	6 - galben
3 - violet	7 - alb

Pe un televizor alb/negru, aceste numere sint in ordinea stralucirii. Pentru a utiliza culorile este necesar sa intelegeti cum este aranjat ecranul.

Ecranul are 768 pozitii (24 linii de cite 32 coloane) unde pot fi afisate caracterele. Fiecare caracter consta dintr-o matrice de 8*8 puncte. (Aceasta ar trebui sa va aminteasca de caracterele grafice definite , unde am definit caracterul PI si am reprezentat punctele scrise cu 1 si cele nescrise cu 0, vezi paragraful 4.14).

Un caracter (o matrice de 8*8 puncte) are asociate doua culori, una pentru fond (paper = hirtie) si una pentru scris (ink = cerneala). La punerea in functiune a calculatorului culorile implicite sint alb pentru hirtie si negru pentru cerneala, deci scrisul apare negru pe alb.

Un caracter poate fi normal sau stralucitor. De asemenea poate fi static sau clipitor. Clipirea se face prin schimbarea culorilor pentru hirtie si cerneala intre ele.

Deci un caracter este caracterizat prin:

1. O matrice de 8*8 puncte 0 si 1 care definesc forma caracterului, cu 0 pentru fond si 1 pentru cerneala.
2. Culori pentru hirtie si cerneala, fiecare fiind un numar cuprins intre 0 si 7.

3. Stralucire - 0 pentru normal si 1 pentru stralucitor.
4. Clipire - 0 pentru static si 1 pentru clipitor.

Deoarece fondul si cerneala acopera un caracter, inseamna ca pentru un caracter nu puteti avea mai mult de doua culori. La fel si stralucirea si clipirea caracterizeaza nu un punct ci un caracter care are 64 de puncte (8*8).

Culoarea, stralucirea si clipirea se numesc atributele unui caracter.

Asa cum am mai mentionat anterior, la punerea in functiune a calculatorului culorile implicite sint alb pentru fond, negru pentru cerneala, fara stralucire si fara clipire. Aceste atribute se pot schimba utilizind instructiunile INK, PAPER, BRIGHT si FLASH.

Din meniul de editare alegeti optiunea **Screen**, care va pozitiona cursorul in partea de jos a ecranului si incercati:

```
PAPER 5
```

apoi utilizind PRINT, afisati ceva pe ecran. Ce afisati, apare pe culoarea azuriu.

Forma instructiunilor pentru atribute este:

```
PAPER (numar intreg intre 0 si 7)
INK (numar intreg intre 0 si 7)
BRIGHT (0 sau 1)
FLASH (0 sau 1)
```

Mai sint citeva numere care pot fi utilizate si care au un efect mai putin direct. Astfel 8 poate fi utilizat in toate cele patru instructiuni si inseamna "transparent" in sensul ca vechiul atribut se pastreaza. Sa presupunem ca introduceti:

```
PAPER 8
```

Nici un caracter nu-si va schimba culoarea de fond la 8 intrucit acesta nu exista, dar ce se intimpla este ca, atunci cind se afiseaza un caracter, culoarea hirtiei este pastrata. Instructiunile INK 8, BRIGHT 8 si FLASH 8, fac acelasi lucru cu atributul corespunzator.

9 se utilizeaza numai cu PAPER si INK si inseamna "contrast". Culoarea (cernelei sau fondului) pe care o utilizati este aleasa incit sa faca contrast cu cealalta. Astfel va fi alba daca cealalta culoare este inchisa si se alege negru, daca cealalta culoare este deschisa.

Se considera culori inchise culorile: negru, albastru, rosu si violet si culori deschise culorile: verde, azuriu, galben si alb.

Incercati urmatoarele:

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

Apoi introduceti:

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

Aici culoarea cernelii este aleasa totdeauna ca sa faca contrast cu vechea culoare a hirtiei, pentru fiecare matrice de 8 x 8 puncte.

Culorile la televizor se bazeaza pe faptul ca ochiul uman vede doar cele trei culori: rosu, verde si albastru in diferite

combinatii si intensitati astfel ca rezulta toate culorile.

Tim-S Plus afiseaza culorile utilizind amestecuri de rosu, verde si albastru. de exemplu galben se obtine din rosu si verde - de aceea codul ei este 6, suma codurilor celor doua culori.

Incercati urmatorul program pentru a vedea cum se combina culorile (spatiile de cerneala se obtin prin intrarea in modul grafic tastind GRAPH si apoi cu tasta CAPS SHIFT apasata tastati 8. Pentru a iesi din modul grafic tastati 9):

```
10 BORDER 0 : PAPER 0 : INK 7 : CLS
20 FOR a=1 TO 6
30 PRINT TAB 6; INK 1;"   ... " : REM 18 spatii de cerneala
40 NEXT a
50 LET lindata=200
60 GOSUB 1000
70 LET lindata=210
80 GOSUB 1000
90 STOP
200 DATA 2,3,7,5,4
210 DATA 2,2,6,4,4
1000 FOR a=1 TO 6
1010 RESTORE lindata
1020 FOR b=1 TO 5
1030  READ c :  PRINT INK c;  "      ";:  REM 6 spatii  de
cerneala
1040 NEXT b : PRINT : NEXT a
1050 RETURN
```

Utilizind functia ATTR aflati ce atribute sint la o anumita pozitie pe ecran; ea va fi explicata la sfirsitul capitolului.

Mai exista doua instructiuni INVERSE si OVER, care afecteaza forma caracterului care se afiseaza. Utilizeaza numerele 0 pentru inactiv si 1 activ. Daca utilizati INVERSE 1, atunci fiecare caracter va fi afisat invers fata de forma normala (culoarea fondului devine culoarea cernelii si culoarea cernelii devine culoarea fondului). In mod normal cind scrieti un caracter pe ecran, ce a fost inainte pe acea pozitie, este sters. Utilizind OVER 1, noul caracter se suprapune peste cel vechi. Aceasta se poate utiliza atunci cind doriti sa scrieti caractere compuse ca de exemplu litere subliniate ca in urmatorul program. Resetati calculatorul si selectati +3 BASIC. Semnul de subliniere se obtine tastind SYMBOL SHIFT si 0 simultan.

```
10 OVER 1
20 PRINT "w"; CHR$ 8;"_";
```

Inainte de suprapunere am utilizat caracterul de control CHR$ 8 (spatiu inapoi).

Instructiunile INK, PAPER etc. pot fi utilizate si in instructiunea PRINT, urmate de semnul punct si virgula si fac acelasi lucru pe care l-ar fi facut daca ar fi fost instructiuni separate, doar ca efectul lor este temporar valabil, doar pentru instructiunea PRINT in care au fost specificate. Afisarile ce se fac dupa aceasta instructiune PRINT iau in considerare vechile atribute. Astfel daca introduceti:

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

numai x se va afisa pe fond galben.

Instructiunile PAPER, INK etc. cind sint utilizate ca instructiuni, nu afecteaza culorile din partea de jos a ecranului

(unde se afiseaza mesajele, se introduc datele pentru INPUT). Aceste linii au culoarea hirtiei, culoarea bordurii (marginii) ecranului si cod 9 (culoare contrast) pentru culoarea cernelii, nu au clipire (FLASH 0) si nici stralucire (BRIGHT 0). Puteti schimba culoarea bordurii la una din cele 8 culori (fara 8 sau 9) cu instructiunea:

BORDER numarul culorii.

Cind introduceti date in INPUT acestea vor fi scrise cu cerneala contrast pe fond de culoarea bordurii. Acest lucru poate fi schimbat punind in instructiunea INPUT, culorile pe care le doriti, la fel ca la PRINT, cu PAPER, INK, etc.

Efectul lor dureaza pina la sfirsitul instructiunii sau pina cind se introduc din nou date. Incercati:

INPUT FLASH 1; INK 4;"Introduceti un numar"; n

Editorul utilizeaza intotdeauna cerneala neagra pe fond alb, indiferent de culorile pe care le utilizati dvs.

Mai exista un mod de a schimba culorile utilizind caracterele de control:

CHR$ 16 corespunde la INK
CHR$ 17 corespunde la PAPER
CHR$ 18 corespunde la FLASH
CHR$ 19 corespunde la BRIGHT
CHR$ 20 corespunde la INVERSE
CHR$ 21 corespunde la OVER.

Fiecare este urmat de un caracter care specifica culoarea, ca de exemplu:

PRINT CHR$ 16 + CHR$ 9;"Iuri Gagarin"

are acelasi efect cu:

PRINT INK 9; "Iuri Gagarin"

Normal este sa utilizati instructiunile PAPER, INK etc. si nu caracterele de control, intrucit nu exista nici un avantaj in aceasta.

Functia ATTR are forma:

ATTR (linie, coloana).

Rezultatul este un numar ce contine codificate atributele pozitiei de pe ecran specificate. Puteti utiliza ATTR in expresii la fel ca orice alta functie a calculatorului.

Rezultatul functiei ATTR este suma a patru numere, dupa cum urmeaza:

128 - daca caracterul clipeste; 0 daca nu;
64 - daca caracterul are stralucire; 0 daca este normal;
8 - inmultit cu codul culorii pentru fond;
1 - inmultit cu codul culorii pentru cerneala.

De exemplu, daca caracterul are clipire, este normal ca stralucire, are culoarea fondului galben si culoarea cernelii albastru, atunci numerele ce trebuiesc adunate sint: 128, 0, 8*6=48 si 1 rezultind un total de 177. Testati urmatoarele in-

structiuni:

```
PRINT AT 0,0; FLASH 1; PAPER 6; INK 1;" "; ATTR (0,0)
```

Incercati urmatorul program:

```
10 POKE 22527 + RND * 704, RND * 127
20 GO TO 10
```

Mai incercati:

```
PRINT "B"; CHR$ 8; OVER 1;"/";
```

si apoi:

```
PRINT CHR$ 8; OVER 1;"/"
```

si veti obtine litera B. Explicatia acestui ultim exemplu este ca instructiunea OVER 1 face un sau exclusiv intre caracterul deja existent si caracterul nou care se scrie.

4.17 Grafica

Din sumar:

PLOT, DRAW, CIRCLE
"Pixel" (punct). POINT

Programele exemplu din acest capitol le introduceti si executati cu RUN, utilizind optiunea Screen din meniul de editare.

Veti vedea in continuare cum se deseneaza figuri cu calculatorul Tim-S Plus in modul de lucru Spectrum. Partea de ecran pe care o puteti utiliza are 22 linii si 32 coloane deci, 704 pozitii de caractere. Fiecare caracter este format din 64 puncte (pixeli) (8*8).

Un pixel este adresabil cu doua numere, care sint coordonatele lui. Coordonata x reprezinta pozitia pe orizontala, pornind de la coloana cea mai din stinga. Coordonata y reprezinta pozitia pe verticala, pornind de la linia cea mai de jos a ecranului. Aceste coordonate se scriu perechi si se pun in paranteze: (x,y). Astfel (0,0), (255,0), (0,175), (255,175) reprezinta colturile din stinga jos, dreapta jos, stinga sus si dreapta sus ale ecranului.

Instructiunea:

```
PLOT x,y
```

scrie cu cerneala punctul de coordonate x si y, deci programul

```
10 PLOT INT (RND*256), INT (RND*176):INPUT a$: GO TO 10
```

marcheaza un punct pentru fiecare apasare a tastei ENTER.

Urmatorul program traseaza graficul functiei SIN (sinus) pentru valori intre 0 si 2*PI:

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

Urmatorul program traseaza graficul functiei SQR (radical indice doi) intre 0 si 4.

```
10 FOR n=0 TO 255
20 PLOT n, 80*SQR (n/64)
30 NEXT n
```

Coordonatele x, y ale unui punct de pe ecran sint diferite de linia si coloana din AT. Liniile si coloanele se refera la caractere (matrici de 8*8 puncte) si au valori intre 0 si 21 si, respectiv, 0 si 31. Coordonatele x, y se refera la un singur punct si au valori cuprinse intre 0 si 255 si, respectiv, 0 si 175.

Pentru trasarea liniilor se utilizeaza instructiunea DRAW, iar pentru a se trasa cercuri, se utilizeaza instructiunea CIRCLE.

Instructiunea DRAW (pentru trasare linii drepte) are forma:

```
DRAW x,y.
```

Daca ultima pozitie dupa o instructiune PLOT, DRAW sau CIRCLE este x0, y0 (RUN, CLEAR, CLS si NEW pun aceasta pozitie implicit coltul din stinga jos, de coordonate 0,0), atunci o noua instructiune DRAW x,y va trasa o linie din punctul (x0, y0) pina in punctul (x0+x , y0+y) (este o instructiune DRAW incrementala). Deci instructiunea DRAW determina lungimea si directia liniei, dar nu si punctul initial.

Incercati urmatoarele:

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

Observati ca in instructiunea DRAW pot apare si numere negative, dar nu si in instructiunea PLOT.

Se pot utiliza la PLOT si DRAW si instructiuni de stabilire a culorilor, dar nu uitati ca o culoare acopera intotdeauna o matrice de 8*8 puncte si deci nu poate fi specificata pentru fiecare punct.

Incercati spre exemplificare urmatorul program:

```
10 BORDER 0 : PAPER 0 : INK 7 : CLS : REM ecran negru
20 LET x1=0 : LET y1=0 : REM inceputul liniei
30 LET c=1 : REM culoarea cernelii albastra
40 LET x2=INT (RND * 256) : LET y2=INT (RND * 176) : REM
   linie
50 DRAW INK c; x2-x1,y2-y1
60 LET x1=x2 : LET y1=y2 : REM urmatoarea linie incepe unde
   s-a terminat precedenta
70 LET c=c+1 : IF c=8 THEN LET c=1 : REM culoarea noua
80 GO TO 40
```

Puteti utiliza PAPER, INK, FLASH, BRIGHT, INVERSE sau OVER in instructiunile PLOT sau DRAW la fel ca in PRINT sau INPUT si vor fi urmate de semnul punct si virgula.

Cu instructiunea DRAW se pot trasa nu numai linii drepte ci si linii curbe (parti de cerc), utilizind un al treilea parametru in instructiune:

```
DRAW x,y,a
```

unde x,y au aceleasi interpretari ca mai inainte, iar a este numarul in radiani cu care trebuie sa se intoarca linia. Daca a este pozitiv, atunci linia se intoarce spre stinga, iar daca a este negativ, linia se intoarce spre dreapta. Alt punct de vedere

de a privi parametrul a este ca fractiune de cerc complet ce va fi trasat: (un cerc complet are 2*PI) deci daca a este PI va fi trasat un semicerc, daca a este 0.5*PI, atunci va fi trasat un sfert de cerc s.a.m.d.

Sa presupunem de exmplu ca a=PI. Atunci orice valori vor avea x si y, se va trasa un semicerc.

Incercati:

```
10 PLOT 100,100 : DRAW 50,50,PI
```

Semicercul incepe in punctul (100, 100) si va merge pina in punctul (150, 150).

Executati de mai multe ori programul, dar cu alte valori in locul lui PI, ca de exemplu: -PI, PI/2, 3*PI/2, PI/4, 1, 0 etc.

Instructiunea CIRCLE traseaza un cerc intreg. Forma generala a instructiunii este:

```
CIRCLE x,y,r
```

unde x,y sint coordonatele centrului cercului, iar r este raza. La fel ca la PLOT sau DRAW si in instructiunea CIRCLE puteti utiliza culori.

Functia POINT specifica daca un punct este scris sau nu (daca are culoarea cernelii sau a hirtiei). Are doua argumente, coordonatele puntului (care trebuie puse intre paranteze) iar rezultatul este 0 daca punctul are culoarea hirtiei sau 1 daca are culoarea cernelii.

Incercati:

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

Introduceti:

```
PAPER 7 : INK 0
```

si testati ce efect au INVERSE si OVER intr-o instructiune PLOT. Dam mai jos o lista a posibilitatilor:

```
PLOT;
```

-aceasta este forma cea mai uzuala; scrie un punct cu cerneala;

```
PLOT INVERSE 1;
```

-pune un punct care "sterge", deci pune un punct de culoarea hirtiei;

```
PLOT OVER 1;
```

-schimba culoarea punctului; daca a fost de culoarea cernelii, il face de culoarea hirtiei si invers;

```
PLOT INVERSE 1; OVER 1;
```

-lasa punctul neschimbat, dar schimba pozitia de PLOT.

Un alt exemplu de utilizare a instructiunii OVER este urmatorul. Umpleti ecranul cu caractere scrise negru pe alb si apoi introduceti:

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

Se va trasa o linie,cu goluri in locurile unde aceasta linie atinge scrisul. Acum tastati din nou aceeasi comanda si linia va dispare.

Daca utilizati INVERSE 1, atunci dupa:

```
PLOT 0,0 : DRAW 255,175
```

si

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

atunci se va sterge si din scris.

Acum incercati:

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

si incercati sa o stergeti cu:

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

si nu veti reusi. Linia trebuie stearsa in aceeasi directie in care a fost desenata.

Incercati acum urmatorul program:

```
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
1030 REM acum tastati GRAPH si apoi A
```

si va rezulta un caracter grafic definit in tabela de sah.

Incercati instructiunea PLOT si numarul 8 pentru PAPER, INK, BRIGHT, FLASH.

Incercati urmatorul program si observati diferenta dintre trasarea unui cerc cu functiile SIN si COS si cu instructiunea CIRCLE.

```
10 FOR n=0 TO 2*PI; STEP PI/180
20 PLOT 100+80*COS n, 87+80*SIN n
30 NEXT n
40 CIRCLE 150, 87, 80.
```

Incercati urmatorul program:

```
CIRCLE 100,87,80 : DRAW 50,50
```

Veti observa ca instructiunea CIRCLE lasa pozitia de PLOT intr-un punct nedeterminat. Dupa o instructiune CIRCLE este necesar sa puneti o instructiune PLOT.

4.18 Temporizare

Din sumar:

PAUSE, PEEK, INKEY$

Se intimpla uneori ca doriti ca un program sa aiba o anumita durata de executie (lungime in timp). Pentru a realiza acest lucru puteti folosi instructiunea PAUSE. Forma ei generala este:

```
PAUSE n
```

Efect: opreste executia si afiseaza ecranul pentru n cadre TV

(50 in Europa si 60 in SUA). valoarea lui n poate fi pina la 65535, care da o pauza de aproximativ 22 minute. Pentru PAUSE 0, programul se opreste "nedefinit".

O pauza poate fi intotdeauna scurtata prin apasarea unei taste.

Urmatorul program deseneaza un ceas si miscarea secundarului.

```
10 REM desenarea ceasului
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 REM acum 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 deseneaza
    secundarul
210 PAUSE 42
220 PLOT 128,88 : DRAW OVER 1; sx,sy : REM sterge
    secundarul
400 NEXT t
```

Ceasul va merge aproximativ 55.5 ore, datorita liniei 60, dar poate fi facut sa mearga mai mult. Temporizarea este controlata in linia 210. V-ati astepta ca PAUSE 50 sa-l faca sa se miste o data pe secunda, dar si calculul ia din timp. Cel mai bun test este compararea cu un ceas real si ajustarea liniei 210 dupa acesta.

Se poate insa folosi si alta metoda de temporizare, care utilizeaza continutul anumitor locatii de memorie. (Vezi sectiunea 4.25 din acest capitol).

Introduceti expresia:

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

Aceasta afiseaza numarul de secunde de cind calculatorul a fost pus in functiune (pina la aproximativ 3 zile si 21 ore cind revine la 0).

Urmeaza o varianta a programului anterior:

```
10 REM desenarea ceasului
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 FN t()=INT((65536 * PEEK 23674 + 256 * PEEK 23673
   +PEEK 23672)/50): REM numarul de secunde de la start
100 REM porneste ceasul
110 LET t1=FN t()
120 LET a=t1/30 * PI : REM a este unghiul secundarului
130 LET sx=72 * SIN a : LET sy=72 * COS a
140 PLOT 131,91 : DRAW OVER 1;sx,sy : REM deseneaza
    secundarul
200 LET t=FN t()
210 IF t<=t1 THEN GO TO 200 : REM asteapta pina la
    urmatoarea mutare
220 PLOT 131,91 : DRAW OVER 1; sx,sy : REM sterge secundarul
230 LET t1=t : GO TO 120
```

Acest ceas are o precizie de aproximativ 0.01% (aprox. 10

secunde pe zi) daca se excuta numai programul. Daca utilizati instructiunea BEEP sau daca se opereaza cu discul sau cu alte periferice conectate la Tim-S Plus, ceasul intern al calculatorului se opreste, pierzind timp.

Numerele PEEK 23674, PEEK 23673 si PEEK 23672 sint in memoria calculatorului si sint utilizate pentru a numara cite a 50-a parte dintr-o secunda. Fiecare numara de la 0 la 255 si dupa 255 revine la 0. Locatia de memorie al carei continut se incrementeaza cel mai des, este 23672 la fiecare a 1/50-a parte dintr-o secunda. Cind ajunge la 255, urmatoarea incrementare o aduce la 0 si se incrementeaza cu 1 locatia 23673 (la fiecare 256/50 secunde). Cind locatia 23673 ajunge la 255, la urmatorul pas va deveni 0 si va fi incrementata cu 1 locatia 23674.

Sa presupunem acum ca cele trei numere sint 0 (pentru PEEK 23674), 255 (pentru PEEK 23673) si 255 (pentru PEEK 23672). Aceasta inseamna ca sint aproximativ 2 minute de cind a fost pornit calculatorul. Expresia numerica este (65536 * 0 + 256 * 255 + 255)/50 care este egal cu 1310.7.

Se poate intimpla si urmatoarea posibilitate: la urmatoarea incrementare, numerele vor fi 1, 0, 0, dar daca inainte de incrementare se face evaluarea (in timp) este posibil sa obtineti rezultatul 0,0,0, care este eronat.

Pentru a evita aceasta posibilitate, se fac doua evaluari ale expresiei si se ia in considerare rezultatul mai mare.

Daca observati atent programul anterior, veti vedea ca aceasta se face implicit in program.

Incercati urmatorul exemplu. Definiti functiile:

```
10 DEF FN m(x,y)=(x+y+ABS(x-y))/2 : REM cel mai mare dintre
   x si y
20 DEF FN n()=(65536 * PEEK 23674 + 256 * PEEK 23673 + PEEK
   23672)/50 : REM timp (poate fi gresit)
30 DEF FN t()=FN m (FN n(), FN (n)) : REM timp corect
```

Puteti schimba cele trei numaratoare, astfel incit sa arate timpul real in locul timpului de cind a fost pus in functiune. De exemplu, daca doriti timpul la 10 dimineata utilizati: 10*60*60*50 ceea ce este egal cu: 1.800.000 * 1/50 dintr-o secunda.

Pentru a seta cele trei numaratoare la 27,119 si 64 tastati:

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

Functia INKEY$ citeste tastatura. Nu are argument. Daca apasati o tasta (sau CAPS SHIFT si o tasta) atunci rezultatul functiei este caracterul pe care il da tasta respectiva; daca nu se apasa nimic, rezultatul va fi sirul nul.

Incercati urmatorul program, care actioneaza ca 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
```

Observatie:

La INKEY$ nu trebuie sa apasati si ENTER ca la INPUT.

Stergeti linia 10 din programul de mai sus si observati ce se intimpla.

Alta varianta de utilizare a lui INKEY$ este in comun cu PAUSE, ca in exemplul urmator:

```
10 PAUSE 0
20 PRINT INKEY$;
30 GO TO 10.
```

4.19 Sunet

Din sumar:
BEEP, PLAY

Calculatorul echipat in varianta cu circuit specializat pentru sunet poate produce o varietate de sunete, la mufa corespunzatoare pentru casetofon si muzica. De la mufa poate fi culeș semnal mono sau stereo si amplificat spre boxe de sunet de orice capacitate.

Comanda folosita pentru a genera sunete complexe este PLAY. Sa incercam comanda:

```
PLAY "ga"
```

(E important sa se scrie cu litere mici si mari, deoarece "ga" e diferit de "Ga" si diferit de "gA" si de "GA").

In executie se vor genera doua sunete muzicale (SOL, LA) la diferenta de un ton muzical.

Comanda

```
PLAY "g$a"
```

va genera 2 note asemanatoare ca in exemplul anterior, dar cu nota a doua mai apropiata de prima, la un semiton.

Comanda

```
PLAY "gD"
```

va genera o diferenta intre note numita cincime, iar comanda

```
PLAY "gG"
```

va produce 2 note la o diferenta de o octava (notele vor suna cumva similar).

Instructiunea care va produce sunet in difuzorul calculatorului, chiar daca nu este echipat cu circuit specializat de sunet, este:

```
BEEP durata, inaltime
```

unde durata notei este data dupa tabela:

```
.
.          ...
.
-1         SI              (B)
 0         DO mediu        (C)
 1         DO# sau REb
 2         RE              (D)
 3         RE# sau MIb
 4         MI              (E)
 5         FA              (F)
 6         FA# sau SOLb
 7         SOL             (G)
 8         SOL# sau LAb
```

```
 9        LA            (A)
10        LA# sau SIb
11        SI            (B)
12        DO            (C)
 .
 .        ...
 .        etc.
```

Numerele negative sint pentru notele de sub DO (C) mediu.
De exemplu pentru a genera o jumatate de secunda un sunet LA (A) imediat urmator lui Do mediu, vom tasta comanda:

```
BEEP 0.5,9.
```

Pentru a executa gama Do major vom introduce si executa programul:

```
5 LET scala=0
10 FOR f=1 TO 8
20 READ nota
30 BEEP 0.5,nota+scala
40 NEXT f
50 DATA 0,2,4,5,7,9,11,12
```

Pentru a alcatui gama urmatoare se va inlocui linia 5 cu:

```
5 LET scala=12.
```

Pentru a alcatui **gama anterioara** se va inlocui linia 5 cu:

```
5 LET scala=-12.
```

Comanda BEEP este usor de programat si genereaza sunete simple. Comanda PLAY (care necesita circuit specializat de sunet) genereaza sunete mai complexe, de calitate, pe trei voci simultan si e mai usor de programat. De exemplu pentru a genera nota La din gama Do major vom da comanda:

```
PLAY "a"
```

iar pentru a genera toata gama Do major:

```
PLAY "cdefgabC".
```

Nota DO notata cu C (litera mare) in exemplul de mai sus indica un Do la o diferenta de o octava de DO mediu. O gama este deci un set de note cuprinse intr-o diferenta de tonuri de o octava. Exemplul anterior este numit gama Do major. Un alt set e gama Do minor care poate fi auzita cu:

```
PLAY "cd$efg$a$bC".
```

O nota avind inainte simbolul $ o coboara cu un semiton (o atenueaza). O nota cu simbolul # inainte o ridica cu un semiton (o ascute).
Comanda PLAY genereaza 9 octave. Octava se alege cu litera O urmata de un numar intre 0 si 8, apoi urmind sirul de note de interpretat.
Ca exemplu introducem programul:

```
10 LET O$="O5"
```

```
20 LET n$="DECcg"
30 LET a$=O$+n$
40 PLAY  a$.
```

Se observa utilizarea unei variabile sir ca argument al instructiunii PLAY. Instructiunea PLAY poate avea ca argument o variabila sir lunga de citeva mii de note, care poate fi construita din fragmente mai mici, asa ca in exemplul de mai sus. In linia 10 se poate pune si alta octava prin editarea liniei. Daca nu este specificata octava, +3BASIC ia implicit octava 5 (care incepe cu Do de jos din gama Do major).

Cele trei octave mai normale sint: octava 5 care incepe cu DO mediu (DO de jos din gama Do major), octava 4, la care, ca Do de sus este Do mediu, si, octava 3, care tine pina la Do mediu (exclusiv). O octava se reprezinta prin literele: c,d,e,f,g,a,b ,C,D,E,F,G,A,B (DO, RE, MI, FA, SOL, LA, SI, DO, RE, MI, FA, SOL, LA, SI).

Sirul "O3D" va genera sunet identic cu "O4d". Citeva note din octavele 0 si 1 nu sint generate corect.

Instructiunea PLAY poate genera diverse lungimi de note. Daca in exemplul anterior inlocuim:

```
10 LET O$="2"
```

si dam comanda RUN, apoi punem pe rind

```
10 LET O$="1"
```

si dam RUN, apoi

```
10 LET O$="9"
```

si dam RUN, vom observa schimbarea lungimilor notelor.

Lungimea notelor se poate schimba specificind oriunde in sir un numar intre 1 si 9, si notele ce urmeaza vor fi de lungimea impusa. Vom avea tabele de lungimi:

1 - saisprezecime;
2 - saisprezecime cu punct;
3 - optime;
4 - optime cu punct;
5 - patrime;
6 - patrime cu punct;
7 - nota intreaga;
8 - nota intreaga cu punct;
9 - nota dubla.

Comanda PLAY poate interpreta triplete (3 note executate pe timp de 2 note). Numerele referitoare la triplet se vor referi la urmatoarele 3 note si sint:

10 - triplet saisprezecime;
11 - triplet optime;
12 - triplet patrime.

Simbolul & este folosit pentru a genera tacere (pauza) de lungime egala cu lungimea notei curente. De exemplu daca in programul precedent inlocuim:

```
10 LET O$="O3"
20 LET n$="DEC&cg"
```

vom avea pauza intre notele DO.

Doua note interpretate una dupa alta fara intrerupere se numesc note legate si se obtin folosind simbolul subliniere "_". De exemplu un DO patrime si un DO intreg, interpretate unul dupa altul cu

```
PLAY "5_7c"
```

Pentru evitarea ambiguitatilor exista si nota falsa N folosita ca delimitator. Daca de exemplu avem sirul "O62" prin care vrem sa indicam octava 6 si lungimea 2 pentru note vom genera eroare deoarece calculatorul va interpreta sirul ca octava 62 care genereaza mesajul de eroare **"Out of range"**. Corect vom scrie "O6N2", unde N are rol doar de delimitare a textului.

Volumul sunetului poate fi programat intre 0 (minim) si 15 (maxim) folosind litera V urmata de numarul volumului.

Pentru a interpreta pe mai mult de 2 canale simultan (din 3) vom utiliza PLAY cu argumente pentru fiecare canal separate prin virgula:

```
10 LET a$="O4cCcCgGgG"
20 LET b$="O6CaCe$bd$bD"
30 PLAY a$,b$.
```

Orice text muzical pate fi interpretat pe oricare din cele 3 canale, dar impunerea tempo-ului se va face doar pe canalul A (primul argument al lui PLAY). Tempo-ul se programeaza cu litera T urmata de un numar intre 60 si 240. Valoarea normala este de 120 adica 2 patrimi pe secunda.

Daca vom modifica programul:

```
5 LET t$="T120"
10 LET a$=t$+"O4cCcCgGgG"
20 LET b$="O6CaCe$bd$bD"
30 PLAY a$,b$
```

prin modificarea liniei 5 vom putea auzi diferite tempo-uri de interpretare.

Repetitia in interpretare a unui grup de note se poate face incluzind grupul respectiv intre paranteze normale (). In exemplul anterior putem rescrie linia 10:

```
10 LET a$=t$+"O4(cC)(gG)"
```

cu acelasi rezultat. Daca se termina textul muzical cu), atunci textul va fi repetat continuu. De exemplu:

```
PLAY "O4N2cdefgfed)"
```

si,

```
PLAY "O4N2cd(efgf)ed)"
```

se vor interpreta pina se tasteaza BREAK. Aceasta facilitate poate fi utila la programarea fondului de bas.

Pentru a stopa orice interpretare se va plasa litera H in locul unde se vrea terminarea melodiei inclusiv a fondului bas, care pote fi in repetitie indefinita. De exemplu:

```
10 LET a$="cegbdfaC"
```

```
20 LET b$="O4cC)"
30 PLAY a$,b$
```

va executa melodia din a$ cu fondul **bas repetat** indefinit, b$. Daca dorim terminarea generarii sunetului la terminarea melodiei principale, vom modifica linia:

```
10 LET a$="cegbdfaCH".
```

Instructiunea PLAY poate programa si variatia (anvelopa) in timp a volumului in executia unei note. De exemplu o nota poate incepe cu valoarea maxima si poate termina cu valoarea minima. Pentru programarea aceastor efecte se foloseste litera W, urmata de un numar intre 0 si 7, cu care se alege forma de unda folosibila si litera U, care permite producerea acestor efecte pe acel canal. Canalele programate cu V nu vor functiona la folosirea lui U.

Pentru forma undei (variatia volumului notei in timp) avem tabelul:

0 - descreste de la maxim la minim si ramine minim
1 - creste de la minim la maxim apoi scade minim
2 - descreste de la maxim la minim apoi creste brusc la maxim
3 - creste de la minim la maxim, apoi ramine maxim
4 - dinti de ferastrau: maxim la minim, apoi creste brusc la maxim ... etc.
5 - dinti de ferastrau: minim la maxim, apoi scade la minim ... etc.
6 - minim la maxim, apoi maxim la minim ... etc.
7 - maxim la minim, apoi minim la maxim ... etc.

Ca exemplu, programul urmator va genera aceeasi nota cu fiecare din cele 8 forme de unde pentru DO:

```
10 LET a$="UX1000W0C&W1C&W2C&W3C&W4C&W5C&W6CV&W7C"
20 PLAY a$.
```

Comanda "X1000" adica X urmat de un numar intre 0 si 65535, programeaza cit de lung va fi efectul de volum. Daca nu se specifica o comanda X, +3BASIC va lua valoarea cea mai mare ca initiala.

Formele de unde 0 la 3 suna bine cu X aproximativ 1000, pe cind formele 4 la 7 (repetitivele) suna bine cu X mai scurt de circa 300.

Comanda PLAY poate genera si zgomot - alb pe oricare din cele 3 canale, mixat sau fara alte note muzicale. Pentru a selecta mixaj de note cu zgomot se foloseste litera M urmata de un numar intre 1 si 63. Se foloseste tabelul:

1 - A = canal cu sunet;
2 - B = canal cu sunet;
4 - C = canal cu sunet;
8 - A = canal cu zgomot;
16 - B = canal cu zgomot;
32 - C = canal cu zgomot.

De exemplu, daca se doreste folosirea canalului A pentru zgomot, B pentru sunet si C pentru mixaj, vom aduna 8+2+4+32=46 si acest numar va urma lui M. Efecte superioare se obtin pe canalul A in interpretare.

Daca dorim sa includem comentarii in sirul text al frazei muzicale, vom marca inceputul si sfirsitul textului de comentariu cu !. De exemplu:

```
10 LET z$="CDcE3Ge4_f! linia 3! egeA"
```

Aceste comentarii nu sint considerate la interpretare.

Cu instructiunea PLAY se poate comanda si un sintetizator extern de muzica (conectat corespunzator) cu pina la 16 canale (1...16). Folosind Y se specifica numarul canalului. Folosind Z urmat de cod (in zecimal) se poate programa sintetizatorul in cod.

De exemplu, pentru a interpreta pe un sintetizator cu 4 canale avem programul:

```
10 LET a$="Y1T10002(((1CCg$b))(($E$E$b$D))((FFC$E))((GGDF))))"
20 LET b$="Y205N&&&&C$bfG)"
30 LET c$="Y304((3C&)C&1CCDD(3$E&($E&1$E$EEE(3F&)F&1FF$G$G(3G&)
          G&1GG4EC))
40 LET d$="Y4N9&&&&&&&&(9EGF7b5CD))"
50 PLAY a$, b$, c$, d$.
```

Sumar de programare PLAY:

```
a...g
A...G - specifica nota din octava curenta;
    $ - nota ce urmeaza este atenuata (bemol);
    # - nota ce urmeaza este ascutita (diez);
    O - specifica octava curenta (0 la 8);
1...12 - specifica lungimea notei;
    & - specifica tacere de o lungime curenta;
    _ - specifica note legate;
    N - separator;
    V - specifica volumul (0...150);
    W - specifica efecte de volum (0...7);
    U - permite efecte de volum;
    X - specifica durata efectelor de volum (0...65535);
    T - specifica tempo-ul (60...240);
  () - marcheaza fraza ce se repeta;
 ! ! - comentariu;
    H - oprirea executiei instructiunii PLAY;
    M - specifica modul de folosire a canalelor (1...63);
    Y - specifica numarul canalului pentru sintetizator (1
        la 16);
    Z - codul de programare pentru sintetizator.
```

4.20 Operatii cu fisiere

Din sumar: **Unitati de disc**
FORMAT, SAVE, LOAD
Nume de fisier
CAT, MERGE
ERASE, MOVE, COPY
RAM disc
Atribute de fisier
Operatii cu caseta
VERIFY, CAT

4.20.1 Unitatile de disc

Calculatorul dispune de 2 unitati de disc flexibil de 5.25" pe care utilizatorul le poate folosi ca memorie de masa pentru salvarea/incarcarea fisierelor continind programe/date proprii sau ale altor utilizatori scrise sub interpretorul +3 BASIC. Cele 2 unitati sint notate unitatea **A:** , cea din stinga si unitatea **B:** , cea din dreapta. Fiecare disc are o capacitate de 180 K din care 173 K disponibili pentru utilizatori (prima versiune).

Procesorul poate adresa doar 64 K de memorie fizica. Sistemul +3 organizeaza restul de memorie disponibila ca pe un disc numit RAM disc cu o capacitate de 58 K disponibili utilizatorului, identificat ca unitatea **M:** (memorie). Discul **M:** este mai rapid decit discurile magnetice, dar este volatil, la RESET sau la intreruperea alimentarii, continutul lui se pierde.

Exceptind comanda **FORMAT**, toate comenzile referitoare la discuri se aplica discului **M:**. Comanda **NEW** nu afecteaza continutul discului **M:**.

Daca dintr-un anumit motiv discul **B:** nu e functional calculatorul se poate folosi numai cu unitatea **A:**. Atunci cind se ajunge la operarea cu discul **B:** va apare mesajul:

Please put the disk for B: into
the drive then press any key

Se va pune discul care ar trebui sa fie pus in **B:** in unitatea **A:** apoi se va tasta. Calculatorul va trata discul **A:** ca si cum ar fi discul **B:**. Daca in continuare calculatorul va solicita sa lucreze cu discul **A:** se va genera mesajul :

Please put the disk for A: into
the drive then press any key

si va trebui pus in unitatea **A:** discul corespunzator.
Acest mod de lucru e util cind se foloseste comanda **COPY** pe disc.

Pentru un disc nou, nefolosit, prima operatie inainte de folosire va fi formatarea. Prin formatare orice informatie aflata anterior pe disc se pierde, iar discul e gata pentru lucrul in +3 BASIC. Se foloseste comanda:

FORMAT "a:"

Daca discul nu e in unitate sau clapeta nu e inchisa, la executia comenzii se va genera mesajul **"Drive not ready"**. Se va pune discul de formatat in unitatea **A:** si se repeta comanda.

Se poate intimpla ca in lucrul cu discul sa apara si mesajul **Drive A: not ready** urmat probabil de **Retry, Ignore or Cancel?**. Cauza este absenta discului in unitate.

De cite ori un mesaj de eroare e urmat de RIC (**Retry, Ignore or Cancel?**) sint posibile 3 actiuni.

Daca de exemplu, mesajul este **Drive not ready**, atunci se pune discul in unitate si se tasteaza R (Retry-Reincearca). Ca urmare sistemul va incerca sa execute ultima comanda data discului. Daca de exemplu la jumatatea copierii unui fisier mare, apare mesajul de eroare **Missing adress mark** aceasta semnifica aparitia unei alterari a continutului discului. Daca dupa mai multe incercari cu R (Retry) apare mereu acelasi mesaj inseamna ca discul e compromis. Dar daca totusi continutul nedistrus al fisierului e interesant se va incerca I (Ignore - Ignora), care va spune sistemului +3 sa ignore eroarea aparuta, insa nu mai e garantie pe veridicitatea datelor citite.

Daca de exemplu eroarea care apare nu mai poate fi mascata cu R sau I, se va abandona executia comenzii tastind C (Cancel - anuleaza). Ca urmare +3 va abandona executia si va genera un mesaj de eroare de obicei similar cu primul.

Pentru protectie, operatia de formatare asupra unui disc deja formatat va genera mesajul:

Disk is already formatted
A to abandon, other key continue

Tastind A se va abandona operatia de formatare, daca s-a luat alt disc din greseala.

Operatia de formatare dureaza circa jumatate de minut, dupa care apare mesajul **0 OK.** Dupa formatare discul e utilizabil de sistemul +3. Un disc vechi prin reformatare se poate sterge cu pierderea completa a informatiei de pe el.

Discul poate fi compromis prin praful care poate patrunde in anvelopa, lichide varsate pe el, cimpuri magnetice care se inchid pe el (telefon, televizor, sursa), prin scoaterea din unitati in timpul scrierii, sau prin lasarea in unitate in timpul intreruperii alimentarii. Un disc compromis poate da erori in **LOAD, SAVE.** Prin reformatare e posibil sa mai fie utilizat.

Daca trebuie recuperate fisiere de pe un disc compromis se indica copierea individuala a fisierelor pe un disc bun (prin **COPY**). Daca exista fisiere ce nu pot fi copiate, probabil ele sint irecuperabile. Se recomanda memorarea in mai multe copii a programelor si datelor de mare valoare. Si in lucrul obisnuit se recomanda utilizarea copiilor. Se pot face copii si pe caseta, aceasta fiind un suport magnetic mai ieftin si robust.

Comanda **FORMAT** apartine +3DOS ca orice alta comanda si ea nu afecteaza +3 BASIC sau programe din memorie.

Daca avem programul "drapel" :

```
10 BORDER 7 : CLS
20 FOR i = 0  TO  21
30 PRINT PAPER 2; "          ";
       PAPER 6; "          "; PAPER 1; "          "
40 NEXT i
50 PRINT #0; "Albert Einstein"
60 PAUSE 0
```

aceasta se salveaza pe disc prin tastarea comenzii:

```
SAVE "drapel"
```

Cuvintul "drapel" e un nume folosit la identificarea fisierului pe disc. Aceste nume de fisiere sint diferite de cele folosite la salvarea pe caseta.

4.20.2 Nume de fisier

Formatul numelui de fisier e acelasi ca si la sistemul de operare **CP/M.** Mai mult, fisierele generate cu **+3** sint compatibile **CP/M PLUS.** Aceasta caracteristica e utila pentru utilizatorii care vor sa translateze cod sau text ASCII de la **+3** la **CP/M PLUS** sau invers.

Numele fisierului e format din 4 cimpuri:

-numarul utilizatorului;

-litera unitatii de disc;
-numele fisierului pe disc;
-tipul fisierului.

Primele doua cimpuri si ultimul sint optionale. Numarul utilizatorului e intre 0 si 15 si poate fi omis deoarece se refera la echipamente compatibile CP/M cu foarte mare capacitate de memorie de masa. Pe +3 sint posibile maximum 64 de fisiere pe un disc asa ca nu se pune problema partitionarii memoriei de masa pe zone-utilizator. Numele fisierului va avea structura :

numar utilizator litera unitatii de disc : nume fisier

Litera unitatii de disc poate fi **A:**, **B:** sau **M:** functie de ce unitate se foloseste.
Da exemplu putem salva programul "drapel" pe discul **A:** in zona utilizator 3 prin comanda;

```
SAVE "3A: drapel"
```

Se recomanda precautie in folosirea zonelor utilizator deoarece comanda **CAT** va lista doar continutul in fisiere a unei zone din cele 16.
E mai directa salvarea cu comanda :

```
SAVE "a: drapel"
```

In cazul folosirii comenzii fara specificarea unitatii

```
SAVE  "drapel"
```

unitatea de disc folosita va fi ultima pe care s-a lucrat anterior. In initializarea sistemului unitatea de disc **A:** este cea pe care se lucreaza.
Pentru schimbarea unitatii de disc cu care se lucreaza de exemplu pe discul **M:** se poate folosi una din comenzile:

```
SAVE "M:" sau LOAD "M:"
```

care nu au alta actiune decit ca aleg **M:** ca unitate de disc pe care se lucreaza. Deci comenzile

```
 SAVE "B:"
SAVE "drapel"
```

au aceeasi actiune ca si comanda

```
SAVE "b: drapel"
```

Numele fisierului poate fi format din 1 pina la 8 caractere de pot fi : litere mari sau mici, cifre sau unul din simbolurile: ", #, $, ´, @, ^, _ , {, }, ~. Literele mari sau mici sint tratate la fel. Deci fisierul cu numele "MARE" poate fi la fel de bine si apelat cu numele "mare". Numele poate continua cu un cimp de tip (optional) care incepe cu un simbol punct si continua cu 3 simboluri. +3 BASIC nu aloca automat acest cimp, deci utilizatorul poate completa dupa voia lui acest cimp.
De exemplu pentru fisierele continind programe BASIC, cimpul poate fi **.BAS**, iar pentru fisiere continind cod Z80 cimpul poate fi **.COD** sau **.BIN.**
Programul nostru BASIC se poate salva cu:

```
SAVE "drapel.bas"
```

Citeva exemple de nume permise :

```
K
drapel
m:poza.bin
b:ioan
11a:fisier
OM:MARE
program.bas
test.pas
b:b.b
```

Citeva exemple de nume eronate

```
un test          - nu se admite spatiu
casa (2)         - nu se admit paranteze
<>/= &           - nu se admit aceste semne
locomotiva       - mai mult de 8 litere
.pas             - nu se admit puncte
7: dubios        - nu e specificata unitatea de disc
```

4.20.3 Catalogul discului

Pentru a afla ce contine un disc se va utiliza comanda **CAT**, care provoaca listarea pe ecran a continutului (prin nume de fisiere) discului. Lista numelor apare in ordine alfabetica iar la sfirsit se afiseaza si spatiul liber disponibil pe disc in Ko. Fiecare nume de fisier e urmat de lungimea lui in Ko (rotunjita in sus).

Comanda simpla **CAT** va da catalogul discului cu care se lucreaza. Daca se vrea catalogul unui anumit disc (de exemplu **m:**) acesta se specifica in comanda:

```
CAT "m:"
```

Comanda **CAT "m:"** nu modifica discul cu care se lucreaza (asa cum fac **LOAD** si **SAVE**).

4.20.4 Marcatori

Pentru un disc cu multe fisiere se pot selecta fisiere de interes folosind marcatorul *.

Daca de exemplu se vrea catalogul tuturor fisierelor de tip **.COD** se va tasta comanda :

```
CAT "*.COD"
```

Daca nu e nici un fisier de acest tip +3 va genera mesajul **"No files found"** urmat de spatiul liber pe disc in Ko. Daca se vrea catalogul tuturor fisierelor cu numele "PROGRAM" indiferent de tip se va da comanda :

```
CAT "PROGRAM.*"
```

Comanda CAT "*.*" e sinonima cu comanda **CAT.** Daca dupa executia ei apare mesajul **"No files found"** aceasta inseamna ca

discul e gol (173 K liberi pentru **A:** si **B:** si 58 K liberi pentru **M:**). Unele discuri pot fi protejate la **CAT.**

Daca dorim catalogul fisierelor de tip **.BAS** al caror nume incep cu X se va da comanda:

CAT "X*.bas"

Daca dorim catalogul fisierelor al caror nume incep cu S si al caror nume tip incepe cu T vom da comanda:

CAT "S*.T*"

Se observa ca marcatorul * e folosit singur sau dupa un simbol, cu semnificatia "nu conteaza ce text e pina la sfirsit" in nume sau tip.

Pentru a putea da o specificatie ca de exemplu "nu conteaza ce e pe pozitia 2" din numele de fisier, se foloseste marcatorul ?, ca in comanda:

CAT "a?a.bas"

Comanda va da catalogul fisierelor de tip **.bas** cu numele de 3 simboluri incepind si sfirsind cu **a.**

Iata citeva exemple de folosire corecta a marcatorilor:

```
?.bas          - (nume de o litera si tipul .bas);
??rapel.*      - (nume de 7 litere ultimele"rapel",orice tip);
pro*.b?s       - (nume ce incep cu "pro", tip ce incepe cu b
                 si sfirseste cu s);
????????.???   - (la fel ca *.*);
?????.*        - (toate numele de 5 semne, indiferent de tip);
*.?            - (toate cu tipul de o litera);
???.c          - (toate cu nume de 3 litere si tipul c);
?x*.*          - (orice tip, a doua litera din nume este x).
```

Daca imprimanta e conectata si functionala se poate lista catalogul folosind comanda :

CAT #3

sau variante, ca de exemplu:

CAT #3, "a:*.cod"

care va lista catalogul fisierelor de tipul **.cod** de pe discul din unitatea **a:**. Specificarea #3 se refera la faptul ca sirul 3 este cuplat cu canalul de imprimanta la initializare. Daca imprimanta nu e functionala, abandonarea comenzii se face tastind **BREAK.**

Orice forma a comenzii **CAT** poate sfirsi cu cuvintul **EXP** (de la expandare), ceea ce va determina furnizarea unui catalog extins, cuprinzind atributele fisierelor aflate pe disc (protectie, stare, etc.).

4.20.5 Lucrul cu discul

Ca memorie de masa, discul flexibil ofera o varianta sigura de a salva programele simplu si rapid utilizind comanda **SAVE.** De exemplu vom salva programul "drapel":

SAVE "a:drapel.bas" LINE 10

In urma comenzii se va genera pe discul de unitate **a:** fisierul cu nume "drapel" de tipul **.bas**, cuprinzind un program si variabilele lui in **+3 BASIC**, autolansabil din linia 10.

Dupa salvarea programului, apasarea butonului de **RESET** va sterge toata memoria RAM a sistemului **+3**, dar ceea ce am salvat pe discul flexibil ramine.

Daca s-ar utiliza comanda **NEW**, doar continutul memoriei folosite de +3BASIC va fi pierdut, continutul RAM discului **m:** va ramine neschimbat.

Dupa **RESET** se alege din meniul principal optiunea **+3 BASIC**. Tastind apoi comanda :

LOAD "drapel.bas"

se va citi de pe disc fisierul specificat care, deoarece a fost salvat cu autolansare, se va lansa in executie din linia 10. Comanda **LOAD** distruge tot ce a fost anterior ei, program sau variabile **BASIC** in memorie, daca isi desfasoara executia normal. Comenzile **SAVE, LOAD** cu discul necesita specificarea unui nume de cel putin o litera. Forma comenzii **LOAD ""** se refera doar la lucrul cu caseta semnificind "incarca urmatorul program **BASIC** de pe banda". La disc aceasta forma nu are sens si va genera mesaj de eroare. E bine ca notarea numelor fisierelor sa se faca mnemonic asa incit sa-l ajute pe utilizator la identificare.

In meniul principal optiunea **"Loader"** va actiona astfel: intii va cauta pe disc un program cu numele *, care de obicei e un program in cod masina (deoarece in **BASIC** nu se admite numele *). Daca il va gasi il va lansa in executie. Daca nu, va cauta un fisier numit **"Disk"**. Acest fisier poate fi un program, chiar in **BASIC** scris de utilizator si salvat anterior. Daca sistemul **+3** va gasi fisierul cu numele **"Disk"** pe disc, atunci il va incarca in memorie. Daca se vrea editarea programului dupa incarcare sa va tasta sageata cursor jos si apoi **ENTER**, ceea ce va provoca selectarea optiunii **+3 BASIC** din meniul principal.

Daca nu e un fisier cu numele **"Disk"** pe discheta sau nu e discheta in unitate, sistemul va incerca sa incarce programul de pe caseta aparind mesajul:

Insert tape and press PLAY
To cancel - press BREAK twice

Aceasta metoda e recomandata la incarcarea programelor scrise sub **+2, +3** sau **128 (Spectrum)** de pe caseta.

Dupa cum am vazut comanda **LOAD** in executie sterge programul si variabilele **BASIC** aflate anterior in memorie, incarcind programul nou. Exista insa comanda **MERGE**, care similar cu **LOAD**, incarca programul si variabilele, insa sterge doar acele variabile vechi, care coincid ca nume si tip cu variabilele nou incarcate, si sterge liniile **BASIC** vechi care coincid ca numar de linie cu cele nou incarcate.

Comanda **MERGE** se refera doar la programe **BASIC**.

Similar cu caseta, pentru comanda **SAVE** exista forma ce se refera la tablouri:

SAVE "b.mat" DATA B() - (salveaza matrice B() in fisierul b.mat)

sau

SAVE "TEXT.DOC" DATA T$() - (salveaza matrice T$() in

fisierul TEXT.DOC)

Daca ulterior vom avea comanda:

LOAD "b.mat" DATA X()

in executie sistemul va calcula daca e loc in memorie pentru ce contine fisierul b.mat, si apoi va sterge continutul variabilei X(), incarcind in ea continutul lui b.mat.

O comanda ulterioara de forma:

LOAD "TEXT.DOC" DATA A$()

va genera stergerea oricarei variabile tablou cu numele A$.

Se recomanda in lucrul cu blocuri tip **DATA** (matrici, texte, etc.) folosirea discului **M:** care ofera o viteza mare de operare.

Alta forma pentru comanda **SAVE** se refera la salvarea blocurilor de cod **(CODE)**:

SAVE "nume" CODE adin, lung

care salveaza codul Z80 aflat incepind de la adresa inceput **adin** pe lungime **lung** in fisierul cu numele **"nume"**.

La incarcare vom avea :

LOAD "nume" CODE adin , lung

care va incarca codul din fisierul **"nume"** incepind de la adresa **adin**, doar daca lungimea e mai mica decit **lung**. Daca lungimea reala a codului continut in fisierul **"nume"** e mai mare decit **lung** se va genera mesajul de eroare **Code lenght error** (se stie ca pentru caseta mesajul in situatia mentionata e **R Tape loading error**).

Daca se foloseste comanda :

LOAD "nume" CODE adin

sistemul va incarca tot continutul de cod al fisierului **"nume"** incepind de la adresa **adin**.

Comanda :

LOAD "nume" CODE

va incarca codul aflat in fisierul **"nume"** la aceeasi adresa de unde a fost salvat cu comanda **SAVE.**

Pentru salvarea imaginilor ecran exista comanda :

SAVE "nume" SCREEN$

sinonima cu

SAVE "nume" CODE 16384, 6912

In fel, pentru incarcarea imaginilor ecran se poate folosi comanda :

LOAD "nume" SCREEN$

sau

LOAD "nume" CODE 16384, 6912

sau eventual

```
LOAD "nume" CODE 16384
```

sau

```
LOAD "nume" CODE
```

Comanda **VERIFY** pentru lucrul cu discul nu se executa.

4.20.6 Salvarea protectiva

Daca utilizatorul salveaza pe disc un fisier al carui nume se nimereste identic cu un alt fisier deja prezent pe disc, sistemul va renumi cimpul tip al fisierului existent cu **.BAK** (de la backup-reface), si va salva pe dischetа fisierul cu numele dat in comanda. Daca exista un fisier cu acelasi nume si cimpul tip **.BAK** acest fisier va fi sters. Acest lucru e util cind utilizatorul salveaza variante succesive in lucru de program sub acelasi nume. El va dispune in orice moment de doua variante anterioare pe disc.

4.20.7 Stergerea si renumirea fisierelor

Comanda **ERASE** ofera posibilitatea stergerii fisierelor de pe disc. Comanda trebuie urmata de un nume de fisier la care se pot folosi si marcatorii * si ? in functie de actiunea de stergere dorita(la fel ca la comanda **CAT**). Pentru stergerea unui anumit fisier (de exemplu cu numele **TEST.COD** pe discul **B:**) se specifica numele explicit:

```
ERASE "B:TEST.COD"
```

Executia comenzii provoaca stergerea completa a fisierului respectiv.

Daca se vrea stergerea unui grup de fisiere de pe disc se vor folosi marcatorii * si ?. Inaintea executiei comenzii sistemul va cere confirmarea actiunii distructive de stergere. Tastind Y, vor fi sterse fisierele din grup. Tastind N comanda e abandonata. De exemplu pentru a sterge toate fisierele tip **.BAS** din discul **m:** avem :

```
ERASE "m:*.BAS"
```

Inainte de executie sistemul va cere confirmarea actiunii distructive:

```
ERASE m:*.BAS? (Y/N)
```

La tastare Y se vor sterge fisierele **.BAS.**

Daca pe disc nu sint fisiere specificate in nume se va genera mesajul **File not found.**

Atentie la forma comenzii **ERASE** urmata de o litera de unitate de disc (exemplu **ERASE "m:"**) deoarece va provoca stergerea completa a discului fara a cere confirmare. Daca in procesul stergerii se incearca stergerea unui fisier protejat la scris (sau un disc protejat la scriere) procesul e oprit si se genereaza mesaj de eroare.

Pentru redenumirea fisierelor se foloseste comanda **MOVE**, in felul urmator:

```
MOVE "nume vechi" TO "nume nou"
```

Marcatorii * si ? nu pot fi folositi in cadrul comenzii MOVE. Daca numele de fisier nu contine litera unitatii de disc atunci se ia discul cu care se lucreaza in mod curent. Nu e posibila redenumirea unor fisiere aflate pe unitati de disc diferite, ca de exemplu comanda:

```
MOVE "a:prog" TO "b:test"
```

in aceste cazuri fiind generat mesajul de eroare **"No rename between drives"**. Pentru operatii intre unitatile de discuri se recomanda comanda **COPY**.

Daca de exemplu se urmareste schimbarea numelui fisierului **"DATA"** aflat pe discul din unitatea a: in numele **"DATAOK"** vom avea comanda :

```
MOVE "a: DATA" TO "a: DATAOK"
```

Daca, de exemplu, schimbind variante succesive de program pe unitatea **m:** am constatat ca ultima varianta salvata cu acelasi nume ("demo"), nu e de valoare si vrem sa redenumim penultima varianta vom avea:

```
ERASE "demo"
MOVE "demo.bak" TO "demo"
```

4.20.8 Atributele fisierelor

Comanda **MOVE** are si functia acordarii unor atribute speciale diferitelor fisiere. Aceste atribute sint practic biti de informatie care contin semnificatii de valoare pentru sistem.

Sint 3 atribute. Cel mai puternic este protejarea la scriere a fisierelor. O data protejat la scriere fisierul nu mai poate fi sters, sau rescris pina nu e inlaturata protectia. Aceasta protectie soft (atribut) e diferita de protectia la scriere a intregului disc care se poate face prin modificari materiale asupra discului. Dar acest atribut nu ofera protectie la comanda FORMAT, pe cind protectia la scriere a intregului disc (prin astuparea orificiului patrat de pe muchia dischetei de 5 1/4 inch) ofera protectie si la formatare.

Pentru a proteja la scriere un fisier (de exemplu numit "trans.cod") vom da comanda:

```
MOVE "trans.cod" TO "+P"
```

Daca vom incerca sa stergem un fisier protejat la scriere sistemul va genera mesajul de eroare **File is read only.**

Deci comanda :

```
ERASE "trans.cod"
```

va genera mesajul de eroare mentionat.

Pentru indepartarea protectiei la scriere a unui fisier (vezi exemplul) se va da comanda:

```
MOVE "trans.cod" TO "-P"
```

dupa care fisierul poate fi scris sau sters. In comenzile MOVE simbolul + inseamna acordarea atributului iar semnul - inseamna anularea atributului.

Comanda MOVE accepta marcatorii * si ? la specificarea numelor fisierelor. De exemplu pentru a proteja la scriere discul m: vom da comanda:

```
MOVE "m:*.*" TO "+p"
```

Acordarea unui atribut pentru un fisier care are deja acordat acel atribut nu genereaza mesaj de eroare.

Al doilea atribut al fisierelor e atributul despre stare sistem. Acest atribut se comuta folosind comanda MOVE cu +S sau -S. Fisierele avind acest atribut acordat nu vor apare, ca urmare a unei comenzi simple CAT, in catalogul de fisiere. La folosirea comenzii CAT...EXP aceste fisiere vor apare in catalogul discului urmate de cuvintul SYS (fisierele protejate la scriere vor fi urmate de cuvintul PROT). Atentie la faptul ca nu pot fi pe un disc doua fisiere cu acelasi nume cu atribute diferite. Deci daca se va copia pe un disc un fisier cu numele de exemplu "gen" si pe acel disc exista deja un fisier cu numele "gen" cu atributul +S (deci invizibil la comanda CAT simplu) varianta veche "gen" va fi distrusa in favoarea celei noi.

Al treilea atribut se refera la fisierele cu destinatie speciala si e numit atribut arhiva. In sistemul +3 atributul arhiva nu are vreo insemnatate si a fost adoptat pentru compatibilizare cu CP/M. Atributul se comuta folosind comanda MOVE cu +a sau -a. In catalogul extins (CAT...EXP) fisierele avind acordat acest atribut sint marcate cu cuvintul ARC.

Iata citeva exemple de acordare a atributelor:

```
MOVE "*.*" TO "+P"   (protejarea la scriere a intregului
                      disc);

MOVE "*.bas" TO "-S" (toate formele .bas sint vizibile
                      in CAT);

MOVE "*.doc" TO "+a" (toate fisierele .doc au atributul
                      arhiva);

MOVE "m:?.?" TO "-P" (toate fisierele cu numele si tipul
                      dintr-o litera pot fi scrise).
```

Daca in loc de p, a si s se pun alte litere sau mai multe litere se va genera mesaj de eroare **Invalid attribute.**

4.20.9 Copierea fisierelor

In situatia in care apare necesitatea realizarii unei copii de program pe un alt disc (un disc de pastrare, sau un disc de manevra), sau a unei copii din discul m: (pe care s-a lucrat rapid, dar care e volatil) pe un disc magnetic (a: sau b:), comanda COPY ofera utilizatorului un spectru de posibilitati de la copierea individuala a fisierului pina la copierea integrala a dischetei. O forma simpla a comenzii COPY poate fi:

```
COPY "m:test" TO "a:"
```

care realizeaza o copie a fisierului "test" aflata in RAM disc pe

discul a: sub numele "test".

Primul nume din comanda COPY specifica fisierul sursa, iar al doilea nume fisierul destinatie.

Comanda:

```
COPY "m:test" TO "a:testand"
```

va copia fisierul din RAM-disc numit "test" pe discul a: sub numele "testand".

Un alt exemplu, comanda :

```
COPY "p1" TO "prog1.bas"
```

va realiza pe acelasi disc o copie a fisierului "p1" care va avea numele "prog1.bas".

Atentie, daca numele fisierelor sursa si destinatie sint identice (deci sint pe aceiasi unitate de disc), se va genera mesaj de eroare **File already exist** sau **File already in use.**

Daca numele fisierelor sursa contin marcatorii * si ? atunci numele fisierului destinatie va putea fi doar o litera de unitate de disc (a:, b: sau m:).

De exemplu:

```
COPY "a:*.c" TO "m:"
```

va transfera toate fisierele de pe a:, care au tipul .C, in RAM-disc.

Dar, de exemplu, comanda:

```
COPY "m:*.cod" TO "a:*.bin"
```

nu are sens si va genera mesajul de eroare **Destination cannot be wilde.**

Comanda COPY nu copiaza si atributele fisierelor. Daca sint necesare atribute (protectii etc.) atunci ele trebuiesc acordate dupa copierea pe noul disc.

In executie comanda COPY afiseaza pe ecran fisierele cu care lucreaza oferind posibilitatea verificarii actiunilor. La incheierea executiei comenzii se afiseaza un mesaj despre numarul de fisiere copiate.

Pentru realizarea unei copii "sector la sector" dupa un disc se va utiliza comanda :

```
COPY "a:" TO "b:"
```

unde discul sursa este in a: iar discul copie formatat anterior va fi in b:

Pentru realizarea unei copii dupa un disc este indicat sa se utilizeze comanda:

```
COPY "a:*.*" TO "b:"
```

unde discul original este in a: iar discul copie va fi in b:. Se va realiza o copie optima.

Chiar in cazul lucrului cu o singura unitate de disc din cauza nefunctionarii celeilalte unitati se va putea folosi comanda COPY. De exemplu :

```
COPY "a:*.cod" TO "b:"
```

va continua cu mesajul :

```
Please put in the disk for B: into
the drive then press any key
```

apoi dupa scrierea pe disc, va continua cu:

```
Please put in the disk for A: into
the drive then press any key
```

Va continua asa pina la copierea tuturor fisierelor tip **.cod** de pe discul original pe discul copie.

Inaintea utilizarii comenzii **COPY** pentru realizarea unor copieri masive, e indicat sa se elibereze discul M:, deoarece spatiul de memorie liber e folosit de comanda **COPY** iar un spatiu mai mare inseamna rapiditate in executia comenzii.

Forma comenzii:

```
COPY "nume" TO SCREEN$
```

va realiza listarea pe ecran a continutului ASCII a fisierului **"nume"**, cu filtrarea codurilor de comanda, fiind indicata pentru vizualizarea continutului fisierelor cu texte.

Forma comenzii:

```
COPY "nume" TO LPRINT
```

va transmite spre interfata conectata (paralela sau seriala) a continutului fisierului **"nume"** inclusiv codurile de control. Daca anterior s-a folosit comanda :

```
FORMAT LPRINT "R"; "U"
```

prin care s-a ales interfata seriala **RS232** ca periferic, si modul neexpandat pentru codurile transmise, comanda **COPY** va transmite fisierul (cod sau text) prin legatura seriala **RS232** spre exterior (alte calculatoare, sau imprimante etc.).

Daca utilizatorul a luat un disc$formatat cu **+3** si a lucrat cu el pe o alta masina (calculator sau **CP/M**) generind pe el un fisier de cod **Z80** (de exemplu cu numele **"TIM.COM"**) si vrea acum sa citeasca acest fisier cu sistemul **+3**, va trebui sa tasteze comanda :

```
COPY "Tim.COM" TO SPECTRUM FORMAT
```

prin care din fisierul **"Tim.COM"**, generat cu alta masina **(CP/M)**, se construieste fisierul **Tim.HED** (prin impunerea cimpului tip **.HED**) care are un antet **(HEADER)** de 128 octeti recunoscut de **+3** BASIC. Aceasta comanda e valabila doar pentru fisiere continind cod, ele fiind considerate de **+3** de tipul **CODE**, de lungime corecta atit cit a fost fisierul original, dar cu adresa de incarcare in memorie necunoscuta. De aceea la incarcarea cu **LOAD** trebuie specificate adresa de incarcare identica cu adresa de la care codul a fost asamblat pe masina care a generat codul. Deci daca in exemplul dat fisierul **TIM.COM** a fost asamblat si e lansabil in executie de la **#7000 (28672)** atunci comanda de incarcare este:

```
LOAD "Tim.HED" CODE 28672
```

deoarece fisierele continind imaginea ecran **(SCREEN$)** sint tot de tipul **CODE** ele pot fi generate pe un alt calculator si

transportate in modul sugerat mai sus pe +3.

4.20.10 RAM - discul

Unitatea de disc m: , sau RAM-discul ofera o memorie de masa mai rapida decit unitatile magnetice A: si B:, la o capacitate mai mica de numai 58 Ko.

RAM-discul poate fi folosit pentru stocarea unor parti de program BASIC, care pot fi aduse printr-o programare judicioasa in memoria BASIC folosind comanda:

```
MERGE "M:nume"
```

in secvente. Astfel pot fi realizate programe BASIC de pina la 90 Ko, avind grija la realizarea structurii programului.

Pentru uzul utilizatorului poate fi creata pe discheta o biblioteca de rutine specifice care la inceputul executiei unui program (care le foloseste) sa fie aduse folosind comanda COPY in RAM-disc, de unde se incarca in memoria BASIC mai rapid decit daca ar fi pe discurile magnetice. Dar deoarece capacitatea discurilor magnetice e mai mare, e indicata folosirea de la caz la caz si combinata, o proiectare buna a programului rasplatind utilizatorul prin viteza mai mare in executie si performante mai bune.

Ca divertisment, e posibila proiectarea animatiei folosind RAM-discul, prin generarea unor imagini statice cu ajutorul programelor BASIC lente, memorarea imaginilor pe discul M: si apoi redarea rapida succesiva a imaginilor de animatie prin preluarea lor direct de pe discul M:, ca in programul:

```
 10  INK 7: PAPER 2 : BORDR 2 : CLS
 20  FOR f=1 TO 10
 30  CIRCLE f*20, 150, f
 40  SAVE "m:pong" +STR$(f) CODE 16384, 2048
 50  CLS
 60  NEXT f
 70  FOR f=1 TO 10
 80  LOAD "m:pong" +STR$(f) CODE
 90  NEXT f
100  BEEP 0.01, 0
110  FOR f=9  TO 2  STEP-1
120  LOAD "m:pong" +STR$(1) CODE
130  NEXT f
140  BEEP .01, 22
150  GO TO 70
```

Inainte de rularea programului se va sterge discul m: cu comanda:

```
ERASE "m:*.*"
```

prin confirmarea stergerii tastind Y, apoi se executa comanda RUN.

De remarcat salvarea in linia 40 doar a treimii superioare de ecran, pentru marirea vitezei.

4.20.11 Lucrul cu caseta

Sistemul +3 BASIC permite folosirea casetofoanelor obisnuite

echipate cu casete audio obisnuite ca memorie de masa pentru stocarea tuturor categoriilor de informatie posibile in sistem. Aceasta memorie de masa este lenta avind insa avantajul unor capacitati mari de memorare (aproximativ 450 KO pe o fata a unei casete 2*30 min.).

Comenzile discurilor care nu se aplica de loc in lucrul cu caseta sint: **FORMAT, COPY, MOVE** si **ERASE.** Comenzile **LOAD,** MERGE si **SAVE** se aplica in acelasi mod si la lucrul cu caseta. Se aplica si o forma speciala a comenzii **CAT.**

La initializarea sistemului **+3** acesta are ca memorie de masa pentru lucru discul **A:**, asa ca toate operatiile referitoare la memoria de masa vor lucra cu discul **A:**. Am vazut ca pentru a schimba discul de lucru se va folosi una din comenzile:

LOAD "litera unitatii:"

sau

SAVE "litera unitatii:"

unde litera unitatii de disc poate fi **A:**, **B:** sau **M:**.

Daca se va folosi ca litera a unitatii **T:** atunci memoria de masa pentru lucrul curent va fi caseta. Deci daca dorim sa folosim in **LOAD** sau **MERGE** caseta vom da comanda:

LOAD "t:"

si toate operatiile **LOAD, MERGE** ulterioare vor apela caseta.

Similar daca e necesar sa folosim **SAVE** cu caseta vom da comanda :

SAVE "t:"

si operatiile de **SAVE** ulterioare se vor desfasura pe caseta.

De remarcat ca celelalte comenzi cu discul (**MOVE, COPY, CAT** si **ERASE**) vor lucra cu unitatea de disc definita anterior ca disc de lucru. Pentru a reveni la lucrul cu discurile dupa o comanda **LOAD "t:"**, e necesar sa folosim **LOAD** cu litera unitatii de disc (de exemplu **LOAD"a:"**). Similar pentru a reveni la lucrul cu discul dupa o comanda **SAVE "t:"** e necesara folosirea lui **SAVE** cu litera unitatii de disc (de exemplu **SAVE "m:"**).

Pentru a salva un program pe caseta vom da secventa de comenzi :

SAVE "t:"
SAVE "nume program"

In lucrul cu caseta numele programului poate avea maxim 10 caractere si poate contine oricare simbol folosit de calculator inclusiv spatiu. Pentru lucru sint recomandate casetele tip LOW NOISE.

Ca urmare a comenzilor date pentru salvare **+3** BASIC va da mesajul:

Press REC&PLAY, then any key

Tastind se va declansa procesul de salvare al informatiei pe caseta. E necesar sa se stabileasca prin incercari nivelul optim de salvare pentru un anumit tip de casetofon.

La inregistrarea obtinuta prin **SAVE** informatia e generata in 2 blocuri: primul e antetul, al doilea blocul de date. Fiecare

bloc incepe cu o succesiune de impulsuri de sincronizare ce vor genera dungi rosii-bleu pe bordura imaginii TV pe o perioada de circa 5 secunde. Blocul antet contine 17 octeti de informatie despre numele fisierului, lungime, adresa de incarcare si tipul. Octetii de informatie provoaca atit la blocul antet cit si la blocul de date pentru bordura imaginii dungi galben-albastre. Intre cele 2 blocuri e o pauza de 2 secunde.

Un exemplu de salvare a informatiei e urmatorul:

- pozitionati caseta
- dati comenzile :

```
SAVE "t:"
SAVE "nume"
```

- va apare mesajul :

```
Press REC&PLAY, then any key
```

- puneti casetofonul pe inregistrare, apoi apasati orice tasta
- cind sistemul pune mesajul 0 OK atunci a terminat salvarea si se poate opri casetofonul.

Pentru verificarea corectitudinii copiei pe caseta se va folosi comanda VERIFY. Pentru exemplificare vom avea urmatoarea succesiune de actiuni:

- pozitioneaza caseta la inregistrarea ce se verfica
- da comanda:

```
VERIFY "nume" sau VERIFY ""
```

- porneste casetofonul in redare

Pina va gasi antetul inregistrarii bordura va oscila intre culorile rosu si bleu. Cind va gasi inregistrarea bordura se va comporta ca la salvare cu **SAVE**. Se va afisa pe ecran tipul si numele inregistrarii (**PROGRAM, Bytes, Numerical array** sau **Character array**). Daca inregistrarea e valida va apare mesajul **OK**. Comanda **VERIFY** se refera doar la lucrul cu caseta, in lucrul cu discul tratarea informatiei nu necesita aceasta comanda.

La preluarea inregistrarii cu VERIFY sau cu LOAD daca inregistrarea e compromisa fizic (demagnetizari, sifonare de banda, nivel necorespunzator), va apare mesajul de eroare:

```
R Tape loading error
```

Informatia continuta de o inregistrare compromisa e in general pierduta, (eventual e recuperabila pina in locul aparitiei defectului pe banda).

Se indica sa se foloseasca casete normale LOW NOISE dar in buna stare a benzii si casetofoane care nu strica banda.

Daca verificarea a constatat ca avem o inregistrare valida pe caseta, vom putea folosi oricind inregistrarea, preluind-o de pe caseta cu comanda:

```
LOAD "nume"
```

Inceputul executiei comenzii **LOAD** va provoca stergerea oricarui program si variabile **BASIC** aflate anterior in memorie. La terminarea cu succes a incarcarii sistemul va da mesajul "0 OK".

Folosirea comenzii **LOAD""** va provoca incarcarea in memorie a primei inregistrari de pe caseta cuprinzind program **BASIC**. Se va intimpla la fel daca sistemul nu are disc in unitatea a: (sau are disc insa discul nu contine fisierul cu numele * sau cu numele **Disk**) si se alege optiunea Loader din meniul principal. E mai

rapida aceasta manevra pentru a incarca prima inregistrare cu program **BASIC** de pe caseta.

Deoarece multe programe existente pentru calculatoarele compatibile cu sistemul nostru sint memorate pe caseta (+, 128, +2, +3) e indicat sa se specifice o procedura posibila pentru trecerea programelor de pe caseta pe disc.

Vom utiliza lucrul in **LOAD** cu caseta si in **SAVE** cu discul a:

```
LOAD "t:"
SAVE "a:"
```

Apoi pentru fiecare program **BASIC** vom da comanda :

```
LOAD""
```

Dupa incarcarea in memorie se va salva programul pe disc a: cu comanda :

```
SAVE "nume"
```

Atentie la numele fisierelor pe disc!

Daca programul de pe caseta e autolansabil (salvat cu **LINE**) e indicat sa se incarce cu comanda:

```
MERGE ""
```

(Bineinteles pot exista protectii si pentru acest mod de incarcare).

Tablourile pot fi transferate incarcindu-le intii in memorie cu comanda **LOAD** corespunzatoare pentru tipul si dimensiunea lor, apoi salvindu-le pe disc.

Pentru inregistrarile de tip **CODE** si **SCREEN$** e necesar insa sa se cunoasca despre ele adresa de unde au fost salvate din memorie initial pe caseta si lungimea lor. Doar asa pot fi trecute pe disc. Aceste informatii se pot obtine usor folosind comanda **CAT "t:"**.

4.20.12 Catalogul continutului casetei

In urma comenzii:

```
CAT "t:"
```

+3 BASIC va astepta pornirea casetofonului avind caseta al carui continut il inspectam. Pe masura ce inregistrarile sint citite (mai precis, antetele inregistrarilor) vor apare pe ecran toate informatiile despre inregistrari. Abandonarea comenzii se face tastind **BREAK**.

Un exemplu de catalog e urmatorul :

```
"program      " (BASIC)
"test         " LINE 999 (BASIC)
"matrice a    " DATA  a()
"Text         " DATA X$()
"rutina       " CODE 40000, 5000
"ecranul      " CODE 16384, 6912
```

Observam cele 4 tipuri de informatie normala din **+3** BASIC : programe **BASIC** (cu sau fara autolansare), matrici numerice, matrici alfanumerice si blocuri de cod. Ultima inregistrare depis-

tata este un ecran (**SCREEN$**) deoarece are adresa 16384 si lungimea 6912.

Avind aceste informatii despre inregistrari, transferul lor pe disc se poate incerca. Dar aceasta metoda nu e infailibila, din diverse motive cum ar fi:

- existenta inregistrarilor fara header, greu de manevrat, invizibile pentru **CAT "t:"** si neincarcabile cu **LOAD**;
- posibilitatea utilizarii unor variabile ale sistemului +3 din domeniul 23296 - 23755;
- la fragmentele lungi exista posibilitatea alterarii stivei masina a sistemului.

Sa retinem si o forma a comenzii care ne va furniza o copie la imprimanta a catalogului casetei:

CAT #3, "t:"

Pentru abandonarea comenzii se va tasta **BREAK**.

4.21 Folosirea imprimantei

Din sumar:

Imprimante paralele
Imprimante seriale
LPRINT, LLIST
FORMAT
COPY

Sistemul dispune de o iesire paralela Centronix si o legatura seriala RS232 la care pot fi cuplate periferice corespunzatoare (adica interfata Centronix sau interfata seriala RS232). Daca se vrea listarea imaginilor grafice la imprimanta, aceasta trebuie sa fie de tip grafic. (Vezi si capitolul de functionare).

La cuplarea oricaror imprimante e necesar sa se cupleze cablul corespunzator in mod corect.

Initial sistemul lucreaza cu o imprimanta paralela, care daca e pornita, la comanda:

PRINT "Galilei"

va tipari textul la imprimanta, validind buna functionare a imprimantei.

Pentru folosirea legaturii seriale RS232 este necesar sa se initializeze in calculator lucrul pe transmisia seriala prin comanda:

FORMAT LPRINT "R"

si rata transmisiei seriale prin comanda:

FORMAT LINE rata

unde **rata** poate avea valori standard pentru transmisie seriala. Oricum, valoarea initiala pentru rata este 9600.

Aceste comenzi servesc la alegerea protocolului transmisiei in calculator. In imprimanta, alegerea protocolului se face diferit de la tip la tip, de obicei prin pozitionarea unor microintrerupatoare incorporate imprimantei (vezi manualul imprimantei).

Protocolul serial fiind stabilit corect, cablul fiind mon-

tat, la comanda:

```
LPRINT "Legatura seriala"
```

se va imprima textul pe hirtie.

Revenirea la lucrul cu interfata paralela Centronix se face simplu prin comanda:

```
FORMAT LPRINT "C"
```

Instructiunile de lucru cu imprimanta sint LPRINT si LLIST, care fac acelasi lucru la imprimanta ca si PRINT si LIST pe ecran. Optiunea "PRINT" din meniul +3BASIC are acelasi efect ca LLIST.

De Exemplu programul:

```
10 LPRINT "Programul:"
20 LLIST 40
30 LPRINT "listeaza setul de caractere:"
40 FOR n=32 TO 255
50 LPRINT CHR$n;
60 NEXT n
70 LPRINT
```

Comenzile LPRINT si LLIST nu transmit spre imprimanta majoritatea simbolurilor de control (cod 0 la 31) pe care le mascheaza (exceptind CR si LF). Daca se intentioneaza transmiterea spre imprimanta a unui cod de control de la +3BASIC este posibil sa se foloseasca comanda

```
FORMAT LPRINT "U"
```

inainte de lucrul cu imprimanta. Comanda va face sa se transmita la imprimanta codul (octetul corespunzator numarului), nu se mai mascheaza primele 32 se coduri, iar codurile peste 165, asa numitele simboluri BASIC compuse (token), nu vor mai fi expandate (adica, pentru simbolul de cod 255 va fi transmis la imprimanta octetul #FF si nu cele 4 coduri ale cuvintului BASIC , COPY).

Folosind comanda:

```
FORMAT LPRINT "E"
```

se va reveni in modul expandat de lucru cu imprimantele in care codurile sint interpretate prin expandare la simboluri compuse (token) folosite de BASIC si coduri de control folosite de sistem. Initial sistemul este in modul expandat.

De exemplu, daca se vrea transmiterea spre imprimanta a codului ESC si "r", apoi sa se listeze programul, vom avea:

```
10 FORMAT LPRINT "U"
20 LPRINT CHR$(27);"r"
30 FORMAT LPRINT "E"
40 LLIST
```

Comanda COPY realizeaza o copie a imaginii de pe ecran. Selectind optiunea "SCREEN" din meniul de editare se tasteaza comanda:

```
FOR i=0 TO 20:PRINT i,: NEXT i
```

apoi

COPY

si se va produce copia ecranului pe imprimanta. In cazul unor comportari anormale acestea se vor datora necompatibilitatii imprimatei cu sistemul. Deci comanda COPY va realiza pe hirtia imprimantei un punct negru pentru oricare punct scris, indiferent de culoarea cernelei, din fisierul de afisare.

Comanda **COPY EXP** va realiza copia in nuante de gri, discriminind si stralucirea (BRIGHT) zonelor de pe ecran. De exemplu:

```
10 FOR b=0 TO 1
20 BRIGHT b
30 FOR i=0 TO 6
40 FOR c=0 TO 31
50 PRINT INK i;i;
60 NEXT c
70 NEXT i
80 NEXT b
```

Executind programul si apoi comandind

```
COPY EXP
```

se va imprima copia in gri a ecranului. Daca se vrea copia negativa a imaginii de pe ecran se va utiliza comanda:

```
COPY EXP INVERSE
```

Comenzile **COPY EXP** si **COPY EXP INVERSE** produc o copie de dimensiune A4 la imprimanta.

Daca se utilizeaza comenzi de imprimanta cu imprimanta necuplata, calculatorul se va bloca. Pentru deblocare se va tasta BREAK de 2 ori consecutiv.

De exemplu programul:

```
10 FOR n=31 TO 0 STEP -1
20 PRINT AT 31-n,n; CHR$(CODE"0"+n);
30 NEXT n
```

in executie va genera simboluri in diagonala pina se umple ecranul si apare mesajul "scroll?". Modificind AT 31-n in linia 20 cu TAB n, se va intimpla la fel.

Daca se schimba Print din linia 2 cu LPRINT, imprimanta va lista numere in diagonale, toate 32. Schimbind la loc TAB n cu AT 31-n,n in linia 20, programul va lista la imprimanta numerele unul dupa altul. Iesirea prin LPRINT se face prin intermediul unei memorii tampon, care se transmite la imprimare atunci cind:

- zona este plina;
- dupa LPRINT, care nu sfirseste cu ";" sau ",";
- daca ",", "'" sau TAB necesita o noua linie;
- la sfirsitul programului, daca a ramas ceva in tampon;
- eventual la deconectarea imprimantei.

4.22 Siruri spre canale

Din sumar:

Sistemul de siruri (fluxuri de date)

Canale
FORMAT, OPEN, CLOSE

Interfata de comunicatie om-calculator se realizeaza prin intermediul perifericelor. Folosind instructiunile INPUT si INKEY$, se pot citi date de la tastatura. Folosind instructiunile PRINT si LPRINT, se pot transmite date inteligibile prin intermediul ecranului TV sau al imprimantei.

Intern, calculatorul comunica cu perifericele folosind canalele (notate de obicei cu litere). De exemplu, in comunicatie, interpretorul BASIC foloseste cu ecranul canalul S si K; datele de la tastatura se transmit pe canalul K, iar datele spre imprimanta se transmit pe canalul P. Pentru a putea fi accesate canalele trebuiesc deschise. Prin deschiderea unui canal , acesta devine apt sa comunice date. Deschiderea unui canal se face prin conectarea la un sir (sau flux de date). De exemplu comanda:

```
OPEN#7, "p"
```

deschide canalul de imprimanta prin conectarea sirului (fluxului) #7 spre canalul "p". Calculatorul permite 16 siruri din care 4 sint folosite de interpretorul BASIC (sirurile #0, #1 conectate spre canalul "K", sirul #2 spre canalul "S", sirul #3 spre canalul "P"); -sirul #FF-este folosit intern (canal "R"). Aceste 4 siruri spre canale sint implicit deschise.

Comenzile BASIC PRINT, LPRINT transmit date nu inspre imprimanta direct, ci spre cite unul din canale. Canalul, daca este deschis, are in zona de informatii despre canale informatia necesara, despre canale, pentru tratarea intr-un anumit mod specific a datelor (spre ecran sau spre imprimanta).

Deci comenzile:

```
PRINT #3; "Scriu pe imprimanta"
LPRINT #2; "Scriu pe TV"
```

vor avea acelasi rezultat ca si comenzile:

```
PRINT "Scriu pe TV"
LPRINT "Scriu pe imprimanta"
```

Acest mod de adresare prin intermediul sirurilor (fluxurilor) spre canale, permite comunicatia cu diferite periferice folosind aceleasi comenzi (posibilitatea reconfigurarii periferiei).

De exemplu:

```
PRINT #1; "Scriu in josul ecranului"
```

va scrie in zona de editare BASIC.

```
PRINT #3; "Merge imprimanta"
```

va scrie pe imprimanta.

```
PRINT "Acesta este ecran"
```

va scrie pe ecran.

```
LPRINT #2; "Iarasi ecran"
```

va scrie iar pe ecràn.

Se pot scrie programe ce pot iesi cu rezultatele optional pe

unul din periferice. Pentru exemplu:

```
10 REM CALCUL
20 INPUT "Doriti text la imprimanta? D/N"; x$
30 LET SIR=2
40 IF x$="D" OR x$="d" THEN LET SIR=3
50 FOR i=1 TO 100
60 PRINT #SIR; i, SQR i
70 NEXT i
```

Dar, pentru +3 se pot si citi date de la periferice, spre exemplu:

```
10 REM DATE de la RS232
20 FORMAT LINE 9600
30 FORMAT LPRINT "r"
40 OPEN #4,"P"
50 PRINT INKEY$#4;
60 GO TO 50.
```

Daca la programul de mai sus se scriu liniile:

```
45 LET adr=40000
50 POKE adr, CODE (INKEY$#4)
55 LET adr=adr+1
```

atunci datele citite de la RS232 vor fi depuse in memorie incepind de la 40.000.

Se pot directiona iesirile BASIC implicite, de exemplu:

```
10 CLOSE #2
20 OPEN #2, "P"
     .
     .
     .
100$PRINT "Sfirsit"
110 LIST
120 STOP
```

Programul va redirectiona iesirile dinspre ecran (PRINT, LIST) spre imprimanta.

4.23 Lucrul cu porturile

Din sumar:

IN
OUT

Microprocesorul Z80 poate adresa un spatiu fizic de memorie de 65536 de locatii pe 8 biti. Din BASIC orice locatie (octet) poate fi citit cu functia PEEK si orice octet din memoria RAM neprotejata poate fi modificat utilizind instructiunea POKE. Analog, microprocesorul poate adresa un spatiu de porturi pe 8 biti de intrare sau de iesire, in numar de 65536 de porturi. Aceste porturi sint utilizate de microprocesor in tranzitul informatiei cu perifericele (de exemplu cu tastatura sau imprimanta) sau pentru controlul paginilor de memorie si a generarii sunetului. Din BASIC aceste porturi pot fi citite (daca exista) folosind functia IN, iar unele pot fi modificate constructiv folosind instructiunea OUT.

Functia IN este analoaga functiei PEEK; are forma:

IN adresa

unde argumentul reprezinta adresa portului in domeniul 0-65536, rezultatul functiei fiind valoarea zecimala a octetului citit de la acel port.

Instructiunea OUT este asemanatoare instructiunii POKE si are forma:

OUT adresa, valoare

avind ca rezultat scrierea la adresa portului specificata de primul argument in domeniul 0 la 65535 a valorii reprezentate de al doilea argument in domeniul 0 la 255.

Nu toate porturile posibile sint utilizate de catre Tim-S Plus, ci doar un numar restrins. Multe numere de porturi pot avea acelasi rezultat in utilizare, depinzind de modul cum e construit Tim-S Plus.

Pentru reprezentarea adresei porturilor sint utilizati 2 octeti, adica 16 biti. Notam cel mai semnificativ bit cu A15. Notam bitii octetului scris sau citit la un anumit port cu D7, D6... D0; D7 cel mai semnificativ.

Porturile folosite la citirea tastaturii vor fi generate de formula:

```
254+256*(255-2^n)
```

cu n de la 0 la 7.

Vor fi opt semilinii de cite 5 taste (este explicata aici configuratia tastelor la calculatoatele Spectrum originale):

```
IN 65278 (#FEFE)        CS la V
IN 65022 (#FDFE)        A  la G
IN 64510 (#FBFE)        Q  la T
IN 63486 (#F7FE)        1  la 5 (JOY 2)
IN 61438 (#EFFE)        0  la 6 (JOY 1)
IN 57342 (#DFFE)        P  la Y
IN 49150 (#BFFE)        ENTER la H
IN 32766 (#7FFE)        SPACE la B
```

In octetul citit de la porturile de tastatura D0 la D4 corespund tastelor din semilinia data, D0 fiind pentru tasta exterioara, D4 pentru tasta din mijloc. Daca bitul corespunzator tastei este 0, atunci tasta respectiva a fost apasata. Daca este 1, tasta nu a fost apasata. Bitul D6 este folosit la citirea de la casetofon. Daca nu e cuplat casetofonul, el poate fi 0 sau 1.

Cele 2 manete de joc (JOYSTICK - JOY) au urmatoarea interpretare a bitilor: bit 0 - foc, bit 1 - sus, bit 2 - jos, bit 3 - dreapta, bit 4 - stinga.

Portul de iesire 254 (#00FE) impune culoarea bordurii imaginii ecran pe bitii D0, D1 si D2, pe bitul D3 salveaza informatia spre casetofon, iar pe bitul D4 genereaza sunet la difuzor (clasic).

Informatii suplimentare si exhaustive despre porturile calculatorului Tim-S Plus veti gasi in cap.8.

4.24 Memoria

Din sursar:
PEEK
POKE
CLEAR

Toata informatia folosita in sistem se afla codificata pe unitati de 8 biti, numite octeti, in memoria calculatorului. Octetii reprezinta numere in gama 0 la 255 (adica hexazecimal #00 la #FF).

O locatie de memorie poate memora un octet la aceasta masina de calcul pe 8 biti. Microprocesorul Z80 poate adresa un spatiu fix de memorie de la 0 la 65535 (adica #0000 la #FFFF). Spatiul real de memorie al calculatorului este insa mult mai mare. Structurarea si restructurarea memoriei se face folosind un mecanism de paginare a memoriei. Unitatea de memorie cu care lucreaza mecanismul de paginare este pagina de 16K.

Harta memoriei pentru utilizarea calculatorului Tim-S Plus in modul de lucru +3BASIC este urmatoarea:

0-16383 (#0000-#3FFF)-RAM (blocul BR2) continind sistem de operare:

```
        ROM0  p8  Editor
sau     ROM1  p9  Analizor sintactic
sau     ROM2  p10 DOS
sau     ROM3  p11 48K BASIC
```

16384-32767 (#4000-#7FFF)-RAM 5 continind pagina video
32768-49151 (#8000-#BFFF)-RAM 2 continind soft de utilizator
49152-65535 (#C000-#FFFF)-pagina (blocul) de RAM comutabil:

```
      RAM0 p0
sau   RAM1 p1
sau   RAM2 p2
sau   RAM3 p3
sau   RAM4 p4
sau   RAM5 p5
sau   RAM6 p6
sau   RAM7 p7
```

Pe linga cele 4 pagini ROM si cele 8 pagini RAM masina contine si 16K (optional 64K) de memorie RAM video, necesara automatului de afisare. Memoria video poate fi folosita si de procesor paginind corespunzator. Totalul de memorie este de 4*16+8*16=192K la care se aduna 16K (sau optional 64K) memoria video.

Mecanismul de paginare a memoriei actioneaza transparent pentru utilizator, dar daca utilizatorul doreste poate pagina memoria. De exemplu in modul de lucru 48K BASIC utilizatorul vede doar 16K de interpretor BASIC si 48K de memorie pentru utilizator ca la un Tim-S obisnuit.

Locatiile memoriei pot fi citite folosind functia PEEK. De exemplu citim primele 21 locatii de memorie.

```
10 FOR i=0 TO 20
20 PRINT i,PEEK i
30 NEXT i
```

Argumentul functiei PEEK este adresa locatiei pe care vrem s-o citim (in domeniul 0 la 65535).

Comanda POKE m,n permite modificarea locatiei de adresa m la noua valoare n , cu 0<=m<=65535 si 0<=n<=255.

Deci, daca scriem:

POKE 37000,137

octetul de la locatia 37000 va contine 137, si

PRINT PEEK 37000

va da valoarea 137.

Folosirea functiei POKE se va face in cunostinta de cauza, deoarece modificarea anumitor locatii de memorie folosite de sistem poate bloca calculatorul.

Modul de lucru al interpretorului BASIC cu memoria este optim, orice stergere sau adaugare provocind deplasari de continut de zone de memorie. Daca de exemplu se adauga o linie BASIC, toate zonele de memorie de adresa mai mare vor fi deplasate corespunzator spre #FFFF.

Memoria alocata fisierului de afisare contine forma binara a imaginii TV si atribute de culoare. Ceea ce e scris (cu INK) pe imaginea TV in memorie are valoarea 1, iar ceea ce e nescris, 0. Ecranul alfanumeric este format din 24 de linii a cite 32 de caractere. Coltul stinga sus este linia 0, coloana 0. Un caracter este infatisat ca o forma binara de 8*8 pixeli (punctul minim reprezentabil pe ecran), deci pe 8 octeti. Memorarea formei binare a unui ecran se face memorind succesiv zonele liniilor alfanumerice 0...7 apoi 8...15, apoi zona 16...23. Fiecare zona este memorata astfel: intii linia de pixeli 0 (a liniei alfanumerice 0, apoi a liniei 1, ...liniei 7),..., apoi linia de pixeli 1(a liniei 0, liniei 1,...,liniei 7)... apoi linia de pixeli 7(a liniei 0, liniei 1,..., liniei 7). Linia de pixel 0 e prima linie de sus a primului rind alfanumeric (de caractere).

Modificarea pozitiilor alfanumerice se face cu instructiunea **PRINT AT**, iar citirea pozitiilor alfanumerice se face cu functia **SCREEN$**.

Ecranul grafic adresabil BASIC este format din 176 de linii de 256 pixeli cu pixelul de coordonate 0,0 in stinga jos. Pentru modificarea unui pixel se foloseste instructiunea **PLOT**, iar pentru citirea unei pozitii-pixel, functia **POINT**. Aceeasi zona de memorie se poate modifica la nivel de 8 pixeli consecutiv, folosind **POKE**. Ecranul grafic complet este constituit din 192 de linii de pixeli, in componenta lui intrind si liniile alfanumerice din ecranul inferior, folosite la editare.

Zona de atribute de culoare se refera la fiecare din cele 768 de pozitii alfanumerice ale ecranului memorat la rind, incepind cu linia 0, coloana 0. Atributele pot fi citite cu functia ATTR.

Structura memoriei este urmatoarea:

16384 - 22527 ...Fisierul de afisare
(#4000-#57FF)

22528 - 23295 ...Atribute de culoare
(#5800-#5AFF)

23296 - 23551 ...Variabile de sistem extinse
(#5B00-#5BFF) (la 48K BASIC - zona tampon pentru imprimanta)

23552 - 23733 ...Variabile de sistem
(#5C00-#5C35)

CHANS -PROG

23734 - (#5CB6-	...Informatii despre canale (si octet #80)
PROG - VARS	...Textul programului BASIC
VARS - ELINE	...Variabilele programului BASIC (si octet #80)
ELINE - WORKSP	...Spatiul de lucru la editare (si octet #80)
WORKSP - STKBOT	...Spatiul temporar de lucru
STKBOT - STKEND	...Stiva calculator
STKEND - SP	...Spatiul de rezerva (SP=STACK POINTER)
SP - RAMTOP	...Stiva masina si stiva GOSUB
RAMTOP	...Octet #3E
UDG - PRAMT	...Simboluri grafice definite de utilizator

Se observa folosirea zonei de memorie tampon pentru imprimanta de la 48K BASIC, ca zona de variabile de sistem extinse pentru +3 si +2.

Variabilele de sistem contin informatii necesare la mentinerea starii functionale a sistemului si la mentinerea structurii de memorie aflata la un moment dat in executie (PROG, VARS, ELINE, etc.). Denumirea lor este mnemonica.

Informatiile despre canale se refera la starea canalului daca este deschis si spre ce e deschis, asa incit fluxul (sirul) de date e directionat spre rutine interne specifice lucrului cu acel periferic care a fost selectionat.

Categoriile editabile cu care lucreaza interpretorul BASIC sint: linie BASIC, constanta BASIC, variabila BASIC (cu nume de o litera, de mai multe litere, variabila FOR, variabila sir, variabila tablou numeric, variabila tablou alfanumeric).

O linie BASIC are structura:

```
2 oct ... Numarul liniei BASIC (in ordinea scrierii curente )
2 oct ... Lungimea textului ce urmeaza (x+1)
x oct ... Textul liniei de x octeti
1 oct ... #0D - ENTER.
```

O constanta BASIC are structura:

```
1 oct ... #0E (CHR$14) marcator de constanta numerica
1 oct ... Exponentul
4 oct ... Mantisa
```

O variabila BASIC cu nume de o litera are structura:

```
1 oct ... 011xxxxx - unde xxxxx e codul literei nume minus #60
1 oct ... Exponent
4 oct ... Mantisa
```

O variabila BASIC cu nume mai lung are structura:

```
1 oct ... 101xxxxx - unde xxxxx e codul primei litere minus #60
1 oct ... Codul literei a doua
1 oct ... Codul ultimei litere cu bit 7 pe 1
1 oct ... Exponent
4 oct ... Mantisa.
```

O variabila tablou numeric are structura:

```
1 oct ... 100xxxxx - unde xxxxx e codul literei nume minus #60
2 oct ... Lungimea totala a textului ce urmeaza
1 oct ... Numarul de dimensiuni
2 oct ... Prima dimensiune
2 oct ... Ultima dimensiune
5 oct ... Primul element de tablou
5 oct ... Ultimul element de tablou
```

Ca exemplu, pentru tabloul C(3,6) ordinea de memorare a elementelor este: C(1,1), C(1,2),..., C(1,6), C(2,1),... C(2,6), C(3,1), ... C(3,6) (adica pe linii).

Variabila de control a buclei FOR are structura:

```
1 oct ... 111xxxxx - Unde xxxxx este codul literei nume minus #60
5 oct ... Valoarea curenta
5 oct ... Limita
5 oct ... Pasul
2 oct ... Numarul liniei BASIC in bucla (in ordinea scrierii
          normale)
1 oct ... Numarul instructiunii din linie.
```

Variabila sir alfanumeric are structura:

```
1 oct ... 010xxxxx - Unde xxxxx e codul literei nume minus #60
2 oct ... Numarul de caractere din sir
x oct ... Textul sirului.
```

Variabila tablou alfanumeric are structura:

```
1 oct ... 110xxxxx - Unde xxxxx este codul literei nume minus #60
2 oct ... Lungimea textului ce urmeaza
1 oct ... Numar de dimensiuni
2 oct ... Prima dimensiune
   ...
2 oct ... Ultima dimensiune
x oct ... Elementele tabloului.
```

Stiva calculator foloseste la realizarea calculelor aritmetice pentru memorare de operanzi.

Stiva masina ete folosita de microprocesorul Z80 ca stiva procesor. Stiva **GOSUB** este stiva folosita de interpretor atunci cind foloseste instructiunile **GOSUB - RETURN.**

Adresa memorata in variabila de sistem **RAMTOP** contine limita superioara a memoriei folosite de interpretorul BASIC. Comanda **NEW** sterge memoria doar pina la limita data de **RAMTOP.**

Valoarea din **RAMTOP** se poate schimba cu instructiunea:

CLEAR n

unde **n** este noua limita definita pentru sistemul BASIC. Se vor executa operatiile:

- stergerea tuturor variabilelor BASIC;
- stergerea ecranului (ca la CLS);
- pozitia de PLOT devine 0,0;
- indicatorul DATA este pozitionat ca dupa executia comenzii RESTORE;
- sterge stiva GOSUB si o pune la noul RAMTOP, daca fizic aceasta este posibil.

Instructiunea **CLEAR** fara argument executa doar primele 4 actiuni. Comanda RUN executa si **CLEAR.**

Folosind comanda **CLEAR** se poate rezerva spatiu mai mult pentru BASIC in detrimentul simbolurilor grafice definite de utilizator. Daca se asigura o stiva masina sub (#BFE0) se poate apela +3DOS.

Folosind comanda CLEAR 49151 se muta BASIC-ul sub pagina (blocul) de RAM comutabil si prin POKE 23388,16+n; cu n=0...7 se poate aduce una din cele 8 pagini RAM comutabile. Paginile comutabile sint folosite de sistem pentru discul **M:** si pentru operatiile editorului.

4.25 Variabilele de sistem

Din sumár:
PEEK, POKE

Zona de memorie de la #5B00 (23296) la #5CB6 (23734) este folosita aparte de calculator, continind citeva subrutine (de paginare) si locatii numite variabilele de sistem,care se pot citi pentru a afla informatii despre starea sistemului sau se pot modifica (unele) pentru a redefini optiunile de lucru ale sistemului.

In modul 48KBASIC, variabilele din zona #5B00 (23296) la #5BFF (23551) nu exista, zona fiind tampon pentru imprimanta. Programele de 48K care utilizeaza aceasta zona nu se vor executa pe +3BASIC. Daca exista sansa ca un anumit program de 48KBASIC sa adreseze porturi ale +3, prin OUT 32765,48 vom pune pe 1 bitul 5 al portului #7FFD pentru a nu mai permite comutari RAM ulterioare.

Numele variabilelor de sistem este mnemonic , neavind insemnatate pentru sistem. Variabilele notate cu X nu trebuie sa fie modificate folosind POKE. Variabilele notate cu N pot fi modificate prin POKE fara a bloca sistemul. Cele notate cu R nu sint variabile ci doar puncte de lansare a rutinelor de paginare. Se specifica si numarul de octeti ai variabilei, precum si adresa in hexazecimal si in decimal.

Pentru a modifica o variabila de sistem de 2 octeti se va folosi comanda:

```
POKE n,v-256*INT(v/256)
POKE n+1,INT(v/256).
```

Pentru a vizualiza valoarea unei variabile de sistem de 2 octeti, folosim comanda:

```
PRINT PEEK n+256*PEEK(n+1)
```

unde **n** e adresa variabilei in zecimal, iar v este valoarea de va atribuit variabilei.

Adresa	Nume	Comentariu

R16	#5B00 (23296)	SWAP	Rutina de paginare
R17	#5B10 (23312)	ST00	Rutina de paginare
R9	#5B21 (23329)	YOUNGER	Rutina de paginare
R16	#5B2A (23338)	REGNUOY	Rutina de paginare
R24	#5B3A (23354)	ONERR	Rutina de paginare
X2	#5B52 (23378)	OLDHL	Memorare registru HL in comutare
X2	#5B54 (23380)	OLDBC	Memorare registru BC in comutare
x2	#5B56 (23382)	OLDAF	Folosit la comutare ROM
N2	#5B58 (23384)	TARGET	Adresa rutinei in ROM3
X2	#5B5A (23386)	RETADDR	Adresa de revenire in ROM1
X1	#5B5C (23388)	BANKM	Copia continutului portului #7FFD
X1	#5B5D (23389)	RAMRST	Instructiune RST 8. Folosita de ROM1 pentru mesaje de eroare la ROM3
N1	#5B5E (23390)	RAMERR	Numarul erorii pasata intre ROM1 si ROM3. Folosit de SAVE/LOAD ca locatie de memorare temporara
2	#5B5F (23391)	BAUD	Viteza la RS232 in stari T/26 (folosit la FORMAT LINE
N2	#5B61 (23393)	SERFL	Folosite la receptie de date
N1	#5B63 (23395)	COL	Coloana curenta (1...latime)
1	#5B64 (23396)	WIDTH	Latimea hirtiei in caractere (initial 80)
1	#5B65 (23397)	TVPARS	Numarul de parametri asteptati de RS232
1	#5B66 (23398)	FLAG33	Diversi indicatori: Bit 2 - token expandat la PRINT Bit 3 - iesire pe RS Bit 4 - interfata disc prezenta Bit 5 - unitatea B: prezenta
X1	#5B67 (23399)	BANK678	Ultimul octet dat la portul #1FFD
N1	#5B68 (23400)	XLOC	Coordonata X la comanda COPY normala
N1	#5B69 (23401)	YLOC	Coordonata Y la comanda COPY normala
X2	#5B6A (23402)	OLDSP	Adresa stivei vechi (cind se foloseste TSTACK)
X2	#5B6C (23404)	SYNRET	Adresa de revenire pentru ONERR
5	#5B6E (23406)	LASTV	Ultima valoare in PRINT
2	#5B73 (23411)	RCLINE	Linia curenta la renumerotare
2	#5B75 (23413)	RCSTART	Linia de start la renumerotare (initial 10)
2	#5B77 (23415)	RCSTEP	Incrementul la renumerotare (initial 10)
1	#5B79 (23417)	LODDRV	Contine T daca LOAD/VERIFY/MERGE sint cu caseta, altfel A,B sau M
1	#5B7A (23418)	SAVDRV	Contine T daca SAVE e spre caseta altfel A,B sau M
1	#5B7B (23419)	DUMPLF	Folosit la COPY EXP (initial contine 9)
N8	#5B7C (23420)	STRIP1	Folosit la COPY
N8	#5B84 (23428)	STRIP2	Folosit la COPY
X115	#5BFF (23551)	TSSTACK	Stiva temporara folosita cind RAM 7 e paginat (descreste spre STRIP2), garantind 115 octeti cind BASIC cheama +3DOS
N8	#5C00 (23552)	KSTATE	FOlositi la citirea tastaturii
N1	#5C08 (23560)	LASTK	Memoreaza ultima tasta apasata
1	#5C09 (23561)	REPDEL	Initial 35. Timpul (in 1/50 sec)

			cit se tine tasta pentru a fi repetata
1	#5C0A (23562)	REPPER	Initial 5. Intervalul (in 1/50s)
			intre doua repetari a tastei
N2	#5C0B (23563)	DEFADD	Adresa argumentelor functiei
			definite (normal 0)
N1	#5C0D (23565)	KDATA	Memoreaza al doilea octet de
			control tastat al culorii
N2	#5C0E (23566)	TVDATA	Memoreaza octetii de culoare
			AT, TAB spre TV
X38	#5C10 (23568)	STRMS	Adresele canalelor atasate
			fluxurilor de date
2	#5C36 (23606)	CHARS	Adresa setului de caractere
			minus 256
1	#5C38 (23608)	RASP	Lungimea biziitului de avertizare
1	#5C39 (23609)	PIP	Durata sunetului la tastare
1	#5C3A (23610)	ERR NR	Codul de eroare minus 1
			(initial 255)
X1	#5C3B (23611)	FLAGS	Biti de control ai sistemului
			BASIC
X1	#5C3C (23612)	TVFLAG	Biti indicatori pentru lucrul cu
			ecranul
X2	#5C3F (23613)	ERRSP	Adresa adresei de revenire la
			eroare
N2	#5C3F (23615)	LISTSP	Adresa adresei de revenire din
			listare automata
N1	#5C41 (23617)	MODE	Cursorii K,L,C,E sau G
2	#5C42 (23618)	NEWPPC	Numarul liniei unde face GO TO
1	#5C44 (23620)	NSPPC	Numarul instructiei din linia
			unde face GO TO
2	#5C45 (23621)	PPC	Numarul liniei curente in
			executie
1	#5C47 (23623)	SUBPPC	Numarul instructiuni din linia
			curenta in executie
1	#5C48 (23624)	BORDER	Culoarea bordurii
			inmultita cu 8; contine
			atributele folosite in ecranul
			inferior
2	#5C49 (23625)	EPPC	Numarul liniei curente (care
			contine cursorul de linie)
X2	#5C4B (23627)	VARS	Adresa variabilelor BASIC
N2	#5C4D (23629)	DEST	Adresa variabilelor in atribuire
X2	#5C4F (23631)	CHANS	Adresa canalului de date
x2	#5C51 (23633)	CURCHL	Adresa informatiei curente
			folosita in lucrul cu periferia
X2	#5C53 (23635)	PROG	Adresa programului BASIC
X2	#5C55 (23637)	NXTLIN	Adresa urmatoarei linii BASIC
X2	#5C57 (23639)	DATADD	Adresa terminatorului ultimului
			articol DATA
X2	#5C59 (23641)	ELINE	Adresa comenzii editate
2	#5C5B (23643)	KCUR	Adresa cursorului
X2$	#5C5D (23645)	CHADD	Adresa urmatorului caracter de
			interpretat
2	#5C5F (23647)	XPTR	Adresa caracterelor dupa marcato-
			rul ? (la eroare)
X2	#5C61 (23649)	WORKSP	Adresa spatiului temporar de
			lucru
X2	#5C63 (23651)	STKBOT	Adresa stivei calculator
X2	#5C65 (23653)	STKEND	Adresa spatiului liber
N1	#5C67 (23655)	BREG	Registrul B
N2	#5C68 (23656)	MEM	Adresa memoriei folosite de

			"calculator"
1	#5C6A (23658)	FLAGS2	Indicatori de sistem
X1	#5C6B (23659)	DF SZ	Numarul de linii folosite de ecranul inferior
2	#5C6C (23660)	STOP	Numarul primei linii la listare automata
2	#5C6E (23662)	OLDPPC	Numarul liniei unde se face GO TO la CONTINUE
1	#5C70 (23664)	OSPCC	Numarul instructiunii din linie la care se face GO TO la CONTINUE
N1	#5C71 (23665)	FLAGX	Diversi indicatori de sistem
N2	#5C72 (23666)	STRLEN	Lungimea sirului aflat in atribuire
N2	#5C74 (23668)	TADDR	Adresa urmatorului articol in tabela sintactica
2	#5C76 (23670)	SEED	Initializarea pentru RND
3	#5C78 (23672)	FRAMES	Numarator de cadre TV
2	#5C7B (23675)	UDG	Adresa primului UDG (simbol grafic definit de utilizator)
1	#5C7D (23677)	COORDS	Coordonata X folosita la PLOT
1	#5C7E (23678)		Coordonata Y folosita la PLOT
1	#5C7F (23679)	P POSN	Folosit la imprimare
1	#5C80 (23680)	PR CC	Adresa urmatoarei pozitii folosita la LPRINT
1	#5C81 (23681)		Nefolosita
2	#5C82 (23682)	ECHO E	Folosit la introducere date
2	#5C84 (23684)	DF CC	Adresa pozitiei de PRINT in zona de afisare
2	#5C86 (23686)	DF CCL	Ca DFCC, dar in ecranul inferior
X1	#5C88 (23688)	SPOSNA	Numarul coloanei in PRINT
X1	#5C89 (23689)		Numarul liniei in PRINT
X2	#5C8A (23690)	SPOSNL	Ca SPOSN la ecranul inferior
1	#5C8C (23692)	SCR CT	Numarator pentru cite ecrane se afiseaza succesiv fara mesajul "scroll?"
1	#5C8D (23693)	ATTR P	Culorile permanente
1	#5C8E (23694)	MASK P	Folosit pentru transparente in mod permanent
N1	#5C8F (23685)	ATTR T	Culorile temporare
N1	#5C90 (23696)	MASK T	Ca MASK P dar temporar
1	#5C91 (23697)	P FLAG	Indicatori de sistem
N30	#5C92 (23698)	MEMBOT	Memoria folosita de "calculator"
2	#5CB0 (23728)	NMIADD	Adresa rutinei de NMI a utilizatorului
2	#5CB2 (23730)	RAMTOP	Adresa ultimului octet folosit de BASIC
2	#5CB4 (23732)	PRAMT	Adresa ultimului octet de RAM fizic (folosit la 48KBASIC)

4.26 Programarea in cod

4.26.1 USR n

Ne vom referi la citeva probleme privind programarea la nivel de microprocesor si de sistem a calculatorului. Microprocesorul Z80, care este inima calculatorului, e programabil in cod masina (limbaj de asamblare) si ofera pentru programatorul avansat o gama de instructii care permit realizarea unor rutine de utilizator dedicate unor aplicatii speciale. Aceste rutine necesita insa cunoasterea sistemului.

In anexe se poate urmari setul de instructiuni al microprocesorului Z80 ca limbaj de asamblare si cod masina in zecimal si in hexazecimal. Rutina pe care vrem s-o creem necesita codificarea ei intr-un sir de instructiuni. Trecerea de la limbajul de asamblare la codul masina se face folosind programe numite asambloare care se incarca de pe disc (de exemplu cuplul **MONS-GENS**, vezi sectiunea 4.35).

Sa luam de exemplu programul in limbaj de asamblare:

```
Ld bc, 99
ret
```

a carui actiune este de a incarca in registrul BC al microprocesorului valoarea 99 si apoi de a reveni. Aceasta secventa scurta de program de asamblare genereaza un cod - masina de 4 octeti: 1, 99, 0 (pentru **LD BC, 99**) si 201 (**RET**).

Avind codul masina, urmeaza sa-l introducem in memoria calculatorului. Daca am fi folosit un asamblor codul ar fi fost generat automat in memorie la adresa impusa. Deci vom alege locul unde sa punem rutina de cod. E indicat sa folosim in acest scop spatiul dintre limita superioara a sistemului **BASIC** (RAMTOP) si zona caracterelor grafice definite de utilizator (UDG).

Dind comanda

```
CLEAR 65267
```

vom limita sistemul **BASIC** pina la 65267 (**RAMTOP**) prin aceasta formind un spatiu de circa 100 octeti pina la zona UDG.

Pentru a introduce codul masina in memorie vom lansa in executie programul:

```
10 FOR a = 65268  TO  65271
20 READ n : POKE a, n
30 NEXT a
40 DATA 1, 99, 0, 201
```

Acum, pentru a lansa in executie codul masina introdus vom folosi functia **USR** cu argument numeric, adica adresa de lansare in executie a codului masina. Deci

```
PRINT  USR  65268
```

va lansa in executie codul, prin care se introduce in registrul BC valoarea 99, si ca rezultat al functiei **USR 65268** va fi 99. Deci instructiunea **PRINT** va afisa pe ecran valoarea 99.

Adresa de revenire in **BASIC** e pusa in stiva masina, asa ca modul de revenire e simplu prin instructiunea **RET**. Se recomanda sa nu fie folosite registrele IY si I ale microprocesorului in

rutine in cod masina, care sint folosite de catre mecanismul de intreruperi al sistemului. Pentru programe care se executa sub **48 KBASIC, 128 KBASIC** sau **+2** BASIC se indica sa nu se incarce registrul I al microprocesorului cu valori intre #40 si #7F chiar daca nu se foloseste modul de intreruperi IM2 al microprocesorului. Trebuie evitate si valorile in gama #C0 si #FF pentru registrul I daca RAM4 sau RAM7 sint paginate de la #C000.

Variabila de sistem de la #5CB0 (23728)(nefolosita la Tim-S) e folosita la sistemul PLUS ca vector de adresa la intrerupere nemascabila (NMI). Daca apare NMI, vectorul e nul si nu se petrece nimic, dar daca e nenul se va produce un salt la adresa continuta de aceasta variabila. Intreruperea nemascabila (NMI) nu trebuie sa se produca pe timpul cit unitatile de disc sint active.

Se recomanda atentie la mecanismul de paginare in cazul incercarilor de programare in cod, deoarece una din erorile de programare este de a pagina stiva masina in timpul executiei codului, ceea ce va determina o blocare ireversibila. Se recomanda atentie la plasarea in timp si spatiu de memorie a rutinelor de service in intreruperi pentru ca acestea sa fie la timpul dorit in pagina de memorie respectiva. Se recomanda pastrarea unor copii in pagini de memorie ale porturilor de paginare #7FFD si #1FFD care sint doar inscriptibile si nu sint citibile. Sistemul BASIC face aceste copii in variabilele de sistem BANKM si BANK678.

Daca se apeleaza in codul masina scris rutine ale +3DOS intreruperile trebuie permise la intrarea in aceste rutine, iar stiva masina trebuie sa fie sub #BFE0 (49120) si peste #4000 (16384) permitind minim 50 de adrese in stiva.

Revenind la programul din exemplul anterior, se poate salva codul masina folosind comanda :

```
SAVE "nume" CODE 65268, 4
```

Pentru a lansa acest cod vom construi urmatorul program BASIC incarcator:

```
10 LOAD "nume" CODE 65268, 4
20 PRINT USR 65268
```

pe care il vom salva ca program separat cu optiunea autolansabil (LINE 10) prin comanda:

```
SAVE "INCARC" LINE 10
```

Acum, de cite ori vrem sa executam codul masina creat vom incarca programul incarcator (autolansabil) prin comanda:

```
LOAD "INCARC"
```

care va incarca codul masina si va lansa in executie codul masina construit.

4.26.2 Utilizarea rutinelor de +3DOS

Cind interpretorul **BASIC** trateaza functia **USR** configuratia de memorie este: ROM3 (de la #0000), RAM5(de la #4000), RAM2 (de la #8000), RAM0 (de la #C000). Daca nu s-a folosit comanda **CLEAR** n pentru a reduce zona folosita de **BASIC** (RAMTOP), atunci stiva masina se afla in blocul RAM0. Dar, rutinele de +3DOS pot fi

apelate doar daca configuratia de memorie este : ROM2(de la #0000), RAM5 (de la #4000), RAM2 (de la #8000) si RAM7 (de la #C000) si stiva masina e obligatoriu undeva in zona #4000 la #BFE0.

Deci, pentru a apela una din rutinele de +3DOS prin tabela de salturi, e necesar sa se comute atit blocurile de RAM cit si blocuri de ROM, iar stiva masina trebuie mutata atit inainte cit si dupa apelul unei rutine de +3DOS. Daca a fost utilizata comanda **CLEAR** pentru a cobori stiva masina sub #BFE0 (49120) atunci nu mai e necesar a muta stiva.

Pentru utilizatorii avansati in programare dam ca exemplu un program de apel a rutinei DOS CATALOG din cod masina. Rutina in cod masina e apelata dintr-un program BASIC care va furniza un catalog simplu al discului:

```
10 LET sum = 0
20 FOR i = 28672 TO 28758
30 READ n
40 POKE i, n : LET sum= sum +n
50 NEXT i
60 IF sum <>9387 THEN PRINT "Eroarea la DATA" : STOP
70 LET X = USR 28672
80 IF INT (PEEK(28757)/2)=PEEK (28757)/2
   THEN PRINT "Eroare la disc"; PEEK (28758):
   STOP
90 IF X=1 THEN PRINT "Nici un fisier"; STOP
100 FOR i=0 TO X-2
110 FOR j=0 TO 10
120 PRINT CHR$ (PEEK(32781)+i*13+j));
130 NEXT i
140 PRINT
150 NEXT j
160 DATA 243, 237, 115, 0, 144, 1, 253
         127, 58, 92, 91, 203, 167, 246,
         7, 50, 92, 91, 237, 121, 49, 255, 159,
         251
170 DATA 33, 0, 128, 17, 1, 128, 1, 0,
         4, 54, 0, 237, 176, 6, 64, 14, 1, 17,
         0, 128, 33, 81, 112, 205, 30, 1, 245,
         225, 34, 85, 112, 72, 6, 0
180 DATA 243, 197, 1, 253, 127, 58, 92, 91,
         203, 231, 230, 248, 50, 92,
         91, 237, 121, 193, 237, 123, 0, 144, 201
190 DATA 42, 46, 42, 255, 0, 0
```

Se recomanda mentinerea rutinelor in cod si a variabilelor importante in cei 32K centrali ai memoriei (ca in exemplul dat). Codul masina introdus cu **DATA** poate fi vizualizat si analizat de catre utilizatorul avansat folosind un program dezasamblor (de exemplu MONS). Se va observa ca intreruperile sint permise la apelarea rutinei DOS. Dar inainte de a permite intreruperile e necesar ca sistemul sa fie informat de comutarea de ROM, deci utilizatorul trebuie sa modifice variabila de sistem BANK678 la ultima valoare de iesire pe portul #1FFD. Metoda de actualizare este prin copierea variabilei in registrul A al Z80 si modificarea bitilor (SET/RESET) corespunzatori, apoi realizarea comutarii prin instructia OUT. Intreruperile trebuiesc invalidate pe timpul cit variabila de sistem nu reflecta starea curenta a portului #1FFD. Portul #1FFD nu controleaza doar comutarile ROM asa ca e indicata folosirea instructiunilor de mascare AND/OR si a instructiunilor SET/RESET, in actualizarea variabilei de sistem.

Variabila de sistem BANKM memoreaza ultima valoare de la portul #7FFD utilizat la paginare RAM. Se recomanda si aici folosirea instructiunilor SET/RESET si a mastilor cu AND/OR pentru actualizarea variabilei.

Alte detalii despre sistem se refera la modul in care lucreaza optiunea "Loader" din meniul principal. Cind e aleasa aceasta optiune sistemul cauta un fiser cu numele "*" apoi (daca nu gaseste) un fisier cu numele "Disc" apoi (daca nu gaseste) incearca sa incarce primul fisier de la casetofon. De fapt sistemul face urmatoarele operatii: incearca sa incarce un sector autoincarcator de pe discul din A:, acesta fiind sectorul 1 de pe pista 0 si fata 0, doar daca suma de control modulo 256 pe sector este 3. Daca se modifica acest sector (512 octeti) astfel ca suma de control modul 256 sa fie 3, atunci sectorul e considerat incarcator (bootstrap loader). Optiunea "Loader" din meniul principal e similara cu comanda BASIC: LOAD "*", si va cauza incarcarea si lansarea sectorului autoincarcator. Sectorul autoincarcator se va incarca in blocul de RAM superior la #FE00 avind pe primii 16 octeti specificatiile de disc normale si de la #FE10 codul care se va executa propriu-zis.

4.27 Detalii despre +3DOS

4.27.1 Sistemul de operare

Continutul sistemului de operare este incarcat la initializarea sistemului de pe discul a:, de pe discheta sistem generata in mod corespunzator. Sistemul de operare ocupa 4 blocuri de 16 K (notate ROM - RAM cu Sistem de Operare). Doar unul din aceste blocuri e adresat la un moment dat in spatiul de memorie #0000 la #3FFF, printr-un mecanism de comutare ce foloseste porturile #1FFD si #7FFD (asa cum se arata la capitolul despre porturi).

Primul bloc comutat pe spatiul #0000 la #3FFF este ROM0, continind editorul care supervizeaza si genereaza meniurile de dialog cu utilizatorul si functiile de editare.

ROM1 contine analizorul de sintaxa +3 BASIC, toate instructiunile inclusiv partea BASIC a instructiunilor referitoare la lucrul cu discul.

ROM3 contine interpretorul de 48K BASIC (asemanator cu cel de la Tim-S).

Portiunea de cod executata la generarea intreruperii va decrementa o variabila contor necesara lucrului cu discul si cind aceasta variabila va fi zero motorul discului e oprit. Aceasta variabila si alte variabile ale editorului si ale +3DOS-ului sint continute in RAM 7. Pagina RAM7 va fi comutata doar daca bitul 4 al variabilei de sistem FLAGS e pe 1 (adica se executa +3BASIC). Daca prin comanda SPECTRUM (sau din meniu) se comuta in modul de lucru 48K BASIC, atunci acest bit e pus pe 0, si paginarea lui RAM 7 si decrementarea variabilei contor a discului e oprita. Daca bitul 4 al variabilei FLAGS e pus pe 1 din program si procesorul lucreaza in modul 1 cu intreruperile mascabile se va realiza paginarea si decrementarea.

+3 BASIC nu mai contine in ROM3 rutinele de la 128 si de la +2 referitoare la scanarea tastaturii.

Alta modificare din 48K BASIC se refera la intreruperea nemascabila(NMI). Daca se genereaza un semnal corespunzator pe pinul NMI al procesorului se va executa un salt la adresa #0066, unde se verifica variabila de sistem NMIADD. Daca aceasta e 0 se va executa RETN, altfel se va face un salt la rutina de la adresa continuta de NMIADD. In ROM2 la adresa #0066 se executa un RETN.

Blocul ROM3 face posibil modul de lucru 48K BASIC si produce o parte din comenzile de +3BASIC.

ROM2 contine sistemul de operare pe disc. Subrutinele existente aici pot fi folosite de utilizatorul avansat in scrierea unor aplicatii mai complexe. Apelul lor se poate face prin blocul de salt. Punctele de intrare in subrutine (ENTRY) sint continute in tabela de la #0100 (256) in ROM2.

Sistemul +3DOS ofera urmatoarele facilitati:

-2 unitati de discuri flexibile si un RAM-disc;
-compatibilitate cu fisierele CP/M Plus si CP/M 2.2;
-compatibilitate cu fisierele AMSTRAD CPC si PCW;
-pina la 16 fisiere deschise simultan;
-citire/scriere fisiere din/la orice pagina de memorie;
-acces la nivel de octet;
-stergere, redenumire fisiere si modificare atribute pentru fisiere;
-crearea de discuri sistem;
-acces la orice nivel la unitatile de disc;
-optional poate lucra cu 2 discuri logice pe o unitate de disc fizic.

4.27.2 Interfata cu +3DOS

Aceasta interfata e un nucleu de rutine accesibile prin blocul de salt. Rutinele sint de 3 categorii:

-rutine de sistem principale;
-rutine aditionale;
-rutine de baza pentru acces la disc (formatare, copiere, etc.).

Rutine principale de sistem: adresa in hexazecimal

DOS INITIALISE	100	- initializare +3DOS
DOS VERSION	103	- furnizeaza sursa si numarul versiunii
DOS OPEN	106	- creaza si/sau deschide un fisier
DOS CLOSE	109	- inchide un fisier
DOS ABANDON	10C	- abandoneaza un fisier
DOS REF HEAD	10F	- genereaza adresa antetului de fisier
DOS READ	112	- citeste octeti din memorie
DOS WRITE	115	- scrie octeti in memorie
DOS BYTE READ	118	- citeste octet
DOS BYTE WRITE	11B	- scrie octet
DOS CATALOG	11E	- furnizeaza catalogul de directori
DOS FREE SPACE	121	- spatiu liber pe disc
DOS DELETE	124	- sterge un fisier
DOS RENAME	127	- renumeste un fisier
DOS BOOT	12A	- program autoincarcator
DOS SET DRIVE	12D	- discul de lucru
DOS SET USE	123	- zona utilizator

Rutine aditionale de sistem:

DOS GET POSITION	133	- preia indicatorul de pozitionare al fisierului
DOS SET POSITION	136	- initializeaza indicatorul de pozitionare al fisierului
DOS GET EOF	139	- preia pozitia EOF
DOS GET 1346	13C	- impune structura memoriei 1-3-4-6

DOS SET 1346	13F	- relocare a memoriei 1-3-4-6
DOS FLUSH	142	- aduce disc in actualizare
DOS SET ACCESS	145	- schimba modul de acces al fisierului deschis
DOS SET ATRIBUTES	148	- schimba atributele fisierului
DOS OPEN DRIVE	14B	- deschide o unitate de disc cu un singur fisier
DOS SET MESSAGE	14E	- activeaza mesaje de eroare
DOS REF XDPB	151	- pozitioneaza la blocul extins de parametri ai fisierului
DOS MAP B	154	- pune disc logic B pe unitate fizica 0 sau 1

Rutine de lucru pentru acces la disc:

DD INTERFACE	157	- verifica prezenta interfetei cu discul;
DD INIT	15A	- initializeaza unitatile de disc;
DD SETUP	15D	- specifica parametrii unitatilor;
DD SET RETRY	160	- initializeaza numarul de incercari;
DD READ SECTOR	163	- citeste un sector;
DD WRITE SECTOR	166	- scrie un sector;
DD CHECK SECTOR	169	- verifica un sector;
DD FORMAT	16C	- formateaza o pista;
DD READ ID	16F	- citeste identificator de sector;
DD TEST UNSUITABLE	172	- testeaza daca discul e potrivit pentru a scrie pe el;
DD LOGIN	175	- fixeaza discul, initializeaza XDPB;
DD SEL FORMAT	178	- preinitializeaza XDPB pentru DD FORMAT;
DD ASK 1	17B	- verifica daca sint 2 unitati fizice;
DD DRIVE STATUS	17E	- preia stare unitate;
DD EQUIPMENT	181	- verifica tipul unitatii;
DD ENCODE	184	- pune rutina de interceptie pentru protectie la copiere;
DD L XDPB	187	- initializare XDPB din specificatiile discului;
DD L DPB	18A	- intializeaza DPB din specificatiile discului;
DD L SEEK	18D	- cauta unitate de disc;
DD L READ	190	- citeste din unitate;
DD L WRITE	193	- scrie in unitate;
DD L ON MOTOR	196	- porneste motor;
DD LT OFF MOTOR	199	- initializeaza variabila de oprire motor;
DD L OFF MOTOR	19C	- opreste motor;

4.27.3 Alte programe pe disc

+3DOS ofera urmatoarele facilitati pentru orice programe descrise in cod Z80:

- folosind DOS BOOT se poate incarca un singur sector de pe disc, numit sector incarcator (bootstrap loader) care preia lucrul cu toata masina calculator Tim-S Plus;

- folosind DOS SET 1346 se genereaza spatiu de memorie pentru utilizator ceea ce permite unui program non-BASIC sa preia functionarea calculatorului putind utiliza si facilitati ale +3DOS. Daca nu se foloseste +3DOS se cheama DD L OFF MOTOR si se pune pe 0 bitul 4 al variabilei FLAGS.

- o unitate de disc poate fi deschisa cu un singur fisier, ceea ce permite examinarea directorilor si a fisierelor fara a mai trece prin structura de fisiere normala.

4.27.4 +3DOS fara unitatile A: , B:

+3DOS poate fi utilizat si fara unitatile de disc flexibil, caz in care:

- e disponibil doar RAM-discul M:;
- discul initial de lucru e M: (nu A:);
- orice incercare de a folosi A: sau B: va genera eroare "22 - Drive not found";
- deoarece nu mai avem unitatile A:, B: sectorul "cache" nu mai e necesar si capacitatea discului M: creste la 64K (paginile 1,3,4,6). Vor fi 62K de date si 2K de directori (64 de intrari);
- prezenta interfetei cu unitatile de disc mecanic poate fi detectata folosind DD INTERFACE. Daca nu e prezenta toate rutinele de baza DD ... nu trebuie apelate deoarece efectul este nedefinit.

4.27.5 Atributele fisierelor

Bitii 7 ai codurilor ASCII ce formeaza numele (notati f1 ... f8) si al codurilor ASCII ce formeaza tipul (notati t1, t2, t3) contin atributele fisierului dupa cum urmeaza:

f1 ... f4 - disponibil utilizatorului;
f5 ... f8 - rezervati (de obicei 0);
t1 - 0
- 1 = fisier protejat la scriere;
t2 - 0
- 1 = fisier sistem;
t3 - 0
- 1 = fisier cu atribut - arhiva.

Fisierele cu t1=1 nu pot fi scrise, renumite sau sterse. Daca discul e formatat se pierd.

Fisierele cu t2 = 1, optional, pot fi omise din catalogul de fisiere.

Bitul cu t3 nu are insemnatate pentru +3.

Fisierele nou create au toate atributele 0.

Atributele fisierelor pot fi schimbate doar apelind DOS SET ATTRIBUTES (in BASIC e comanda MOVE).

4.27.6 Antetul fisierelor

Fisierele +3DOS pot sa fie cu sau fara antet (header). Fisierele create cu +3 BASIC folosind comanda SAVE au antet. Antetul contine informatii valabile pentru sistem.

Fisierele +3DOS cu antet au pe primii 128 octeti ai fisierului (inregistrarea antet) informatii formind un antet unic. Din 128 octeti doar 8 octeti sint dedicati pentru sistemul BASIC, si sint folositi la fisierele create prin comenzi BASIC (vezi DOS OPEN). Inregistrarile antet sint detectate prin semnatura si prin suma de control. Daca semnatura si suma de control sint cele asteptate atunci avem o inregistrare antet, altfel nu. E putin probabil ca un fisier fara antet sa fie confundat cu un fisier cu antet.

Formatul inregistrarii antet este:

```
Octeti
 0 ...  7 - Semnatura "PLUS 3 DOS";
        8 - #1A (EOF - sfirsit de fisier);
        9 - numarul variantei;
       10 - numarul versiunii;
11 ... 14 - lungimea fisierului in octeti (cel mai putin
            semnificativ octet la adresa cea mai mica);
15 ... 22 - antet de +3 BASIC;
23 ...126 - rezervati (valoare normala 0);
      127 - suma de control (suma octetilor 01...126 modulo
            256).
```

Numarul de varianta si de versiune e dat pentru dezvoltari ulterioare.

+3DOS realizeaza toate operatiile pe antet. Se poate obtine adresa antetului BASIC folosind DOS REF HEAD. Nu e necesar sa se scrie direct cei 128 octeti antet.

4.27.7 Formatul discurilor

+3DOS permite un format al discurilor cu urmatoarele restrictii:

-512 octeti pe sector;
-maxim 255 sectoare pe pista;
-maxim 255 de piste;
-maxim 256 de directori;
-maxim 360 unitati de alocare.

+3DOS suporta acelasi format al discului ca CP/M Plus si LOGO Script (editor de texte) de pe familia de calculatoare AMSTRAD PCW.

4.27.8 Piste si sectoare logice

Subrutinele DOS lucreaza cu piste si sectoare logice. Acestea sint folosite pentru a face transparent pentru +3DOS modul de lucru al unitatii de disc: cu o fata sau cu 2 fete.

Pe un disc cu o fata, numarul logic de pista e acelasi cu numarul pistei fizice.

Pentru un disc intr-o unitate cu dubla fata sint 2 alternative:

-fete alternative (fata 0, pista 0 = pista logica 0):

```
F0-P0=LP0
F1-P0=LP1
F0-P1=LP2
F1-P1=LP3
.
.
.
etc.
```

sau

-fete succesive:

```
F0-P0=LP0
F0-P1=LP1
F0-P2=LP2
.
.
```

```
F1-P0=LP n/2
F1-P1=LP n/2+1
.
.
.
etc.
```

unde n este numarul total de piste.

Numerele pentru sectoare logice sint folosite pentru a **lucra** cu sectoare fizice. Pentru sectoare logice se incepe intotdeauna cu 0.

Sector logic = sector fizic - primul sector fizic

4.27.9 Specificatiile discului

Formatul discului utilizat in +3BASIC este acelasi cu formatul pista simpla al familiei AMSTRAD PCW.

Familia de formate contine:

AMSTRAD PCW - pista simpla (ex.: la model PCW 8256);
AMSTRAD PCW - pista dubla (ex.: model PCW 8512);
AMSTRAD CPC - formatul sistem;
AMSTRAD CPC - formatul comercial;
AMSTRSD CPC - formatul (doar) pentru date.

Formatul IBM al familiei AMSTRAD CPC nu e suportat de sistemul PLUS. Blocul extins de parametri pentru disc (XDPB) e acelasi la PLUS ca la primul format de mai sus. Pot fi folosite alte formate de disc prin programarea corespunzatoare a XDPB.

La fiecare disc din familie pe fata 0, pista 0 in sectorul 1 pe primii 16 octeti sint definite specificatiile discului.

Formatul utilizat de PLUS e acelasi cu formatul discului de tip 0 din tabelul de mai jos:

```
Octet 0 . . . Tipul discului:
                0=STANDARD PCW      DD SS ST (si +3)
                1=STANDARD CPC      DD SS ST - sistem
                2=STANDARD CPC      DD SS ST - data
                3=STANDARD PCW      DD DS DT
                celelalte valori rezervate

Octet 1 . . . Bitii 0 si 1 - impartirea:
                             0 = o fata
                             1 = dubla fata (alternativ)
                             2 = dubla fata (succesiv)
              Bitii 2 ...6 - rezervati (pe 0)
              Bit 7        - pista dubla

Octet  2 . . . Numar de piste pe o parte (fata)
Octet  3 . . . Numar de sectoare pe o pista
Octet  4 . . . Log2 (marimea sectorului) - 7
Octet  5 . . . Numarul de piste rezervate
Octet  6 . . . Log2 (marirea blocului/128)
Octet  7 . . . Numarul blocurilor de directori
Octet  8 . . . Lungimea de limitare (gap) in scriere/citire
Octet  9 . . . Lungime de limitare (gap) in format
Octeti 10 ... 14 Rezervati
Octet  15  Suma de control (folosita daca discul este  auto-
```

lansabil - bootable).

Cind discul e preluat de unitate, specificatiile de disc sint folosite pentru a initializa corespunzator blocul extins al parametrilor discului (XDPB).

4.27.10 Blocul extins al parametrilor discului

Fiecarei unitati logice de disc i se asociaza un bloc standard DPB acelasi cu cel utilizat de CP/M PLUS.

Blocul contine informatii necesare +3DOS pentru a suporta diverse formate. El poate fi programat astfel incit sa utilizeze diverse formate de disc, folosind tabelul :

```
Octeti  0 si  1 ...  SPT inregistrari pe pista
Octet   2  . . .     BHS Log2 (marime bloc /128)
Octet   3  . . .     BLM marime bloc/128 -1
Octet   4  . . .     EXM masca extinsa
Octeti  5 si  6 ...  DSM numarul ultimului bloc
Octeti  7 si  8 ...  DRM numarul ultimului entry la directori
Octet   9  . . .     AL0 harta pe biti a directorilor
Octet  10  . . .     AL1 harta pe biti a directorilor
Octeti 11 si 12 ...  CKS marime vector de suma de control
Octeti 13 si 14 ...  OFF numarul de piste de rezerva
Octet  15  . . .     PSH log2 (marime sector /128)
Octet  16  . . .     PHM marime sector /128-1
Octet  17  . . .     Bitii 0 si 1 - Impartirea
                                    0 - 1 fata
                                    1 - 2 fete alternat
                                    2 - 2 fete succesiv
                     Bit 2 ... 6  - Rezervati (0)
                     Bit 7        - Pista dubla
Octet  18  . . .     Numar de piste pe fata
Octet  19  . . .     Numar de sectoare pe pista
Octet  20  . . .     Numarul primului sector
Octeti 21 - 22 ...   Marimea sectorului
Octet  23  . . .     Lungime delimitare (gap) scriere/citire
Octet  24  . . .     Lungime delimitare (gap) format
Octet  25  . . .     Bit 7 - Operatii multi - pista
                                    1 = pista multipla
                                    0 = pista simpla
                     Bit 6 - Felul modulatiei
                                    1 = MFM
                                    0 = FM
                     Bit 5 - Sari peste adresa datelor sterse
                                    1 = DA
                                    0 = NU
                     Bit 0 ... 4 pe zero
Octet  26  . . .     Indicator:
                        #00 - autodetectie a formatului discului
                        #FF - fara autodetectie
```

Octetul 25 contine in mod normal #60 (96), nerecomandindu-se operatii multi-pista.

Cu octetul 26 se poate preveni +3DOS pentru detectia automata a formatului discului, atunci cind se foloseste XDPB pentru un format nestandard.

In continuare se prezinta standardul folosit in +3, AMSTRAD PCW in pista simpla, tipul 0:

```
SPT  = 36      ... inregistrari pe pista
```

```
BSH  =  3     ... deplasare bloc (shift)
BLM  =  7     ... masca blocului
EXM  =  0     ... masca extinsa
DSM  =174     ... numarul de blocuri - 1
DRM  = 63     ... numarul de directori - 1
AL0  =192(#C0)... 2 blocuri de directori
AL1  =  0
CKS  = 16     ... marime vector suma
OFF  =  1     ... piste rezervate
PSH  =  2     ... deplasare sector fizic (shift)
PHM  =  3     ... marca sector fizic
         0    ... o fata
        40    ... piste pe fata
         9    ... sectoare pe pista
         1    ... numarul primului sector
       512    ... marimea sectorului
        42    ... marime delimitare (gap) R/W
        82    ... marime delimitare (gap) format
        96(#60)... MFM mod
         0    ... autoselectie a formatului
```

Se prezinta in continuare standardul de tipul 1, AMSTRAD CPC, formatul sistem:

```
SPT = 36
BSH =  3
BLM =  7
EXM =  0
DSM =170
PRM = 63
ALO =192
AL1 =  0
CKS = 16
OFF =  2
PSH =  2
PHM =  3
       0       - o fata
      40       - piste pe fata
       9       - sectoare pe pista
      65 (#41) - numarul primului sector
     512       - octeti pe sector
      49       - lungime gap in R/W
      82       - lungime gap in format
      96(#60)  - mod MFM
       0       - autoselect
```

Se prezinta si standardul de tipul 2, AMSTRAD CPC formatul (doar) pentru date:

```
SPT =  36
BSH =   3
BLM =   7
EXM =   0
DSM = 179
DRM =  63
ALO = 192 (#C0)
AL1 =   0
CKS =  16
OFF =   0
PSH =   2
PHM =   3
```

```
  0       - o fata
 40       - piste pe fata
  9       - sectoare pe pista
193 (#C1) - numarul primului sector
512       - octeti pe sector
 42       - lungime gap R/W
 82       - lungime gap format
 96 (#60) - mod MFM
  0       - autoselect
```

4.27.11 Compatibilitatea cu fisierele CP/M

+3DOS utilizeaza structura fisierelor CP/M in urmatoarele restrictii:

- marime maxima a fisierului 8 Mb;
- marime maxima a capacitatii unitatii de disc 8 Mb;
- etichetele de directori sint ignorate;
- nu exista parole;
- nu exista cimpul de data si cimpul de timp;
- atributul arhiva e ignorat.

4.27.12 Model de fisier

Un fisier e un sir de octeti ce poate avea teoretic orice lungime intre 0 si 8 Mb, practic lungimea fiind limitata de capacitatea discului. Se asociaza fiecarui fisier deschis un indicator (pointer) de 24 de biti, care contine adresa urmatorului octet ce e citit/scris, si care e automat incrementat dupa fiecare citire/scriere. Utilizatorul il poate pozitiona la orice valoare necesara lui.

Pozitia de sfirsit a fisierului (EOF) e pozitia octetului urmator ultimului octet scris. Fisierele fara antet (header) pot sa-si inregistreze pozitia de EOF la inceputul urmatoarei inregistrari de 128 octeti. Fisierele cu antet au pozitia de EOF inregistrata exact. Scriind un octet dupa pozitia de EOF va extinde fisierul si va avansa pozitia de EOF.

Citind un octet la (dupa) pozitia de EOF va genera o eroare de citire. Citind un octet nescris dupa pozitia de EOF va da un octet fara sens sau va genera o eroare (nu se recomanda).

4.27.13 Schimbarea discului

In +3DOS schimbarea discului se va face cind discul nu lucreaza si nu e deschis nici un fisier pe acea unitate de disc.

Daca, accidental, un disc este schimbat in timp ce sint inca fisiere deschise pe acea unitate, atunci +3DOS va detecta dupa un timp acest eveniment, sesizind manevra de corectie. +3DOS va detecta aceasta schimbare accidentala de disc atunci cind va citi directorii de pe acel disc.

Schimbind discul in timp ce sint deschise fisiere pe disc in scriere se compromite continutul discului.

4.27.14 Unitati logice si fizice de disc

Daca e necesar, doua unitati logice de disc **A:** si **B:** pot fi simulate pe o unitate fizica de disc, unitatea 0.

Se foloseste rutina DOS MAP B si rutina CHANGE DISK. De cite ori se acceseaza unitatea 0 se verifica ce unitate logica e folosita. Rutina CHANGE DISK va genera mesajul :

Please put the disk for X: into
the drive then press any key

unde X: este A: sau B:
Rutina DOS MAP B poate fi folosita si pentru a simula discul B pe unitatea de disc fizica 1. Daca unitatea 1 nu exista atunci discul **B:** este dezafectat.

4.27.15 Mesaje de eroare la +3DOS

Rutinele +3DOS semnalizeaza erorile folosind indicatorul CY al microprocesorului si registrul A pentru codul de eroare.
Erorile nefatale sint semnalate cu :

0 Drive not ready (Disc nepregatit)
1 Disk is write protected (Disc protejat la scriere)
2 Seek fail (Cautare ratata)
3 CRC data error (eroare de date CRC)
4 No data (Nu sint date)
5 Missing address mark (Nu e adresa marca)
6 Unrecognised disk format (Format necunoscut)
7 Unknown disk error (eroare necunoscuta)
8 Disk changed whilst +3DOS was using it (Disc schimbat in lucru)
9 Unsuitable media for drive (Disc nepotrivit)

Erorile fatale sint:

20 Bad filename (Nume gresit)
21 Bad parameter (Parametru gresit)
22 Drive not found (Unitate de disc absenta)
23 File nou found (Fisier absent)
24 File already exists (Fisier deja creat)
25 End of file (Sfirsit de fisier)
26 Disk full (Disc plin)
27 Directory full (Directori completi)
28 Read-only file (Fisier protejat la scriere)
29 File number not open (Fisier nedeschis)
30 Access denied (Acces refuzat)
31 Cannot rename between drives (Renumire imposibila)
32 Extent missing (Lipseste extensia)
33 Uncached (Eroare soft)
34 File too big (Fisier prea lung)
35 Disk not bootable (Discul nu e autolansabil)
36 Drive in use (Unitati in lucru cu fisiere deschise)

De exemplu, mesajul **Unsuitable media for drive** e cauzat de incercarea de a scrie in modul pista simpla pe o unitate in pista dubla sau de incercarea de a scrie/citi pe disc in pista dubla pe o unitate de pista simpla.
Mesajul **Missing address mark** apare si cind se acceseaza un disc care e neformatat.

4.27.16 Mesaje din +3DOS

Daca mesajele de eroare sint permise (DOS SET MESSAGE) atunci daca apare o eroare nefatala +3DOS va pasa rutinei ALERT un mesaj si utilizatorul va trebui sa raspunda la:

Retry,Ignore or cancel?

Daca raspunsul e R se reincearca operatia cu discul. Daca raspunsul e I atunci eroarea e ignorata si daca e C atunci operatia e ignorata si o conditie la eroare e returnata la rutina ce a chemat ALERT. Daca mesajele de eroare sint dezafectate sau daca eroarea e fatala atunci nu se mai afiseaza nimic si nici o conditie de eroare nu e returnata.

Mesajele erorilor nefatale sint:

```
0 Drive X: not ready
1 Drive X: disc write protected
2 Drive X: track ttt, seek fail
3 Drive X: track ttt, sector sss, data error
4 Drive X: track ttt, sector sss, no data
5 Drive X: track ttt, sector sss, missing address mark
6 Drive X: bad format
7 Drive X: track ttt, sector sss, unknown error
8 Drive X: disk changed, please replace
9 Drive X: disk unsuitable
```

unde **X:** este **A:** sau **B:** si **ttt** e numarul pistei iar **sss** e numarul sectorului.

Mesajele anterioare sint urmate de:

-Retry, Ignore or Cancel?

Rutina ALERT e chemata sa produca unul din mesajele de mai sus daca eroarea se produce cind +3DOS **executa** o rutina DOS.

De exemplu daca DOS OPEN e chemata cu acces de scriere iar discul e protejat la scriere atunci se va returna CY = 0, A = 1 si rutina ALERT nu e chemata.

Daca pe cind se citesc date cu DOS - READ e gasit un sector compromis, rutina ALERT va fi chemata sa informeze utilizatorul. Se apasa R (pentru disc introdus prost), sau I (pentru a ignora sectorul compromis), sau C (pentru a abandona citirea, evident imposibila).

4.27.17 Cerinte ale +3DOS

Cind se apeleaza +3DOS e necesar sa fie reconfigurata memoria, astfel:

```
0     ... ROM2
#4000 ... RAM5
#8000 ... RAM2
#C000 ... RAM7
```

Stiva masina e necesar sa fie sub #BFE0 (49120) si peste #4000 (16384).

Se foloseste valoarea #BFE0 in loc de #C000 pentru ca 30 de octeti sint folositi pentru transferuri interpagini, aceasta zona nefiind folosita de +3DOS dar necesitind sa nu fie stiva acolo.

Stiva e necesar sa aiba circa 100 de octeti; +3DOS poate deschide 16 fisiere simultan.

Fisierele cu numerele 0,1 si 2 sint utilizate de **+3** BASIC. Fisierul 0 va fi inchis cind BASIC raporteaza o eroare.

Pentru toate rutinele +3DOS la intrare si la iesirea din ele intreruperile trebuie si sint validate.

4.27.18 Utilizarea memoriei +3DOS

Blocurile de RAM 1, 3, 4 si 6 sint tratate ca un tablou de 128 de sectoare tampon (numerotate 0 ... 127) de cite 512 octeti fiecare. RAM-discul M: si sectorul "cache" ocupa doua zone continue in acest tablou. Locarea si lungimea lor se initializeaza la initializarea sistemului.

Oricare din sectoarele tampon nefolosite de RAM-disc sau de "cache" sint libere oricarei utilizari. Daca se schimba marimea sau locarea RAM-discului se pierde tot continutul lui de fisiere.

Toate rutinele +3DOS pastreaza configuratia de memorie.

Adresele numelor de fisiere, a tampoanelor pasate acestor rutine trebuie sa fie vizibile, adica pagina RAM unde sint aceste informatii trebuie sa fie comutate in configuratia de memorie.

Blocul DOS de salt e localizat in ROM2 de la adresa #0100 (256) in sus.

4.28 Principalele rutine de sistem +3DOS

DOS INITIALISE #0100 (256)

Initializeaza +3DOS, unitatile de disc, RAM-discul si zona "cache". Nici un fisier nu e deschis. Unitatile de disc nu sint actionate. Daca interfata cu discul este prezenta discul de lucru este A:, altfel este M:. Numarul zonei utilizator in care se lucreaza este 0. Generarea mesajelor de eroare este dezafectata. Numarul de incercari initial e 15.

Intrare

Nici una.

Iesire

Daca OK (rutina se executa normal)
CY = 1 (indicator CY al microprocesorului Z80)
A distrus (registrul A din Z80 isi altereaza continutul)
altfel
CY = 0 (indicatorul CY semnifica existenta erorii)
A = cod eroare (registru A contine codul erorii)
registre distruse (cu continutul alterat):BC,DE,HL,IX
(celelalte registre neschimbate)

DOS VERSION #0103 (259)

Furnizeaza sursa DOS si numarul versiunii (de exemplu V1.0 sau V1.1).

Intrare

Nici una

Iesire

D = sursa (registru D contine de exemplu 1)
E = versiunea (registru E contine 0 - V1.0)
registre distruse : AF, BC, HL, IX

DOS OPEN #0106 (262)

Creaza sau/si deschide un fisier.
Daca fisierul exista deja se va deschide fisierul, altfel se va crea fisierul, cele 2 actiuni - de deschidere si de creare - specificindu-se in registrul DE.

Deschiderea

1. Eroare - fisierul exista deja.
2. Deschide fisierul si citeste antet (daca exista). Pozitioneaza pointerul fisierului dupa antet.
3. Deschide fisierul si ignora antetul. Se pozitioneaza pointerul fisierului la 0.
4. Daca numele fisierului este **nume.tip** atunci sterge **nume.BAK** (daca exista), renumeste **nume.tip** cu numele **nume.BAK** si continua cu crearea.
5. Sterge versiunea existenta si continua cu crearea.

Crearea

1. Eroare - fisierul nu exista.
2. Creaza si deschide noul fisier cu antet. Pozitioneaza pointerul fisierului dupa antet.
3. Creaza si deschide noul fisier fara antet. Pozitioneaza pointerul fisierului la 0.

Exemplu: Pentru a simula o actiune cu caseta de tipul "daca fisierul exista deschide-l altfel creaza-l cu antet" se realizeaza deschidere =1, creare =1 adica, E=1 si D=1.

Exemplu: Pentru a deschide un fisier si a semnala eroare daca el nu exista se pune deschidere E=1 si creare D=0.

Exemplu: Pentru a crea un nou fisier cu antet intii renumind versiunea existenta cu .BAK se pune deschidere E=3 si creare D=1.

Fisierele cu antet au pozitia de EOF ca fiind imediat urmatoarea dupa ultima pozitie scrisa a fisierului.

Fisierele fara antet au pozitia de EOF ca fiind pe octetul de inceput al primei inregistrari de 128 octeti nescrise cu octeti ai fisierului.

Caracterul de cod #1A (26) este octetul marcator pentru EOF asa cum il recunoaste rutina DOS BYTE READ. Acest caracter (soft.EOF) nu are nici o legatura cu pozitia de EOF a unui fisier.

Datele antetului se afla pe 8 octeti putind fi folosite de utilizator. Daca deschidere =1 si fisierul exista (si are antet), atunci datele antetului sint citibile de pe fisier, altfel in zona antetului se afla 0. Datele antetului sint disponibile ca locatii chiar daca fisierul nu are antet. Pentru a accesa aceste date se utilizeaza DOS REF HEAD.

+3BASIC utilizeaza 7 din cei 8 octeti astfel :

Octet	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Program	0			#8000 sau LINE		informatii	
Tablou numeric	1	lungime		XXX	nume	XXX	XXX
Tablou alfanumeric	2	fisier		XXX	nume	XXX	XXX
CODE sau SCREEN$	3			adresa de LOAD		XXX	XXX

XXX nu conteaza

Daca se creaza un fisier care sa poata fi incarcat cu LOAD atunci octetii trebuie sa fie pusi la valorile corespunzatoare.

Daca fisierul e deschis cu acces exclusiv la scriere sau exclusiv la scriere si citire (si are antet) atunci antetul va fi actualizat cind se inchide fisierul.

Un fisier care este deja deschis pentru acces multiplu la citire pe un alt numar de fisier poate fi deschis doar pentru accesul multiplu la citire pe acest numar de fisier.

Un fisier care e deja deschis pentru accesul exclusiv la citire, sau exclusiv la scriere, sau exclusiv la citire si scriere pe un numar de fisier, nu poate fi deschis pe alt numar de fisier.

Intrare

B = numarul fisierului (0 ... 15)
C = modul de acces. Bitii 0, 1, 2:
1 = exclusiv citire
2 = exclusiv scriere
3 = exclusiv citire si scriere
5 = citire multipla
Bitii 3 ... 7 = 0 (rezervati)
D = creare fisier
E = deschidere fisier
HL= Adresa numelui fisierului (fara * sau ?)

Iesire

Daca e creat un fisier nou
CY = 1
Z = 1 (indicator Z la Z80)
A = distrus
Daca exista deschis fisierul
CY = 1
Z = 0
A = distrus
altfel
CY = 0
A = codul erorii
registre distruse : BC, DE, HL, IX

DOS CLOSE #0109 (265)

Inchide fisierul si (daca exista) scrie antetul.

Inscrie date ramase in tampoane. Actualizeaza directorul fisierului. Elibereaza numarul de fisier.

Toate fisierele deschise trebuie si inchise (sau abandonate), altfel numarul de fisier nu poate fi reutilizat.

Intrari

B = numarul de fisier

Iesiri

Daca OK
CY = 1

```
          A  = distrus
altfel
          CY = 0
          A  = codul erorii
          registre distruse: BC, DE, HL, IX
```

DOS ABANDON #010C (268)

Abandoneaza un fisier. E similar cu DOS CLOSE, cu diferenta ca orice antet, date sau directori ce ar urma sa fie scrise pe disc sint neglijate. Poate fi folosita la inchiderea unui fisier care nu poate fi inchis cu DOS CLOSE (de exemplu daca discul e compromis).

Intrari

```
          B = numarul fisierului
```

Iesiri

```
          Daca OK
          CY = 1
          A  = distrus
altfel
          CY = 0
          A  = codul erorii
          registre distruse: BC, DE, HL, IX
```

DOS REF HEAD #010F (271)

Tipareste la antetul de date al fisierului in cauza. Zona de date antet poate fi folosita de utilizator si este disponibila chiar daca fisierul nu are antet. Doar fisierele cu antet si deschise cu acces in scriere vor avea datele de antet inregistrate pe disc (vezi DOS OPEN).

Intrari

```
          B = numarul fisierului
```

Iesiri

```
          Daca OK si fisierul este fara antet
               CY = 1
               Z  = 1
               A  = distrus
               IX = adresa antetului
          Daca OK si fisierul este cu antet
               CY = 1
               Z  = 0
               A  = distrus
               IX = adresa antetului
          altfel
               CY = 0
               A  = codul erorii
               IX = distrus
          registre distruse: BC, DE, HL
```

DOS READ #0112 (274)

Citeste octetii din fisier in memorie, avansind pointerul fisierului. Destinatia e in urmatoarea configuratie de memorie:

```
#C000 ... (49152...65535)- Pagina specificata in registrul C
#8000 ... (32768...49151)- RAM2
#4000 ... (16384...32767)- RAM5
#0000 ... (00000...16383)- ROM2 (DOS)
```

Rutina nu ia in considerare codul #1A (EOF - soft), citirea lui producind eroare.

Intrari

```
B = numarul fisierului
C = Pagina de memorie la #C000
DE= numarul octetilor de citit (0 inseamna 64K)
HL= adresa pentru octetii cititi
```

Iesiri

```
Daca OK
CY = 1
A,DE distrusi
altfel
CY = 0
A  = codul erorii
DE = numar de octeti ramasi necititi
registre distruse: BC, HL, IX
```

DOS WRITE #0115 (277)

Scrie octeti pe un fisier al discului din memorie, avansind pointerul de fisier.
Memoria sursa e data de aceeasi configuratie ca la DOS READ.

Intrari

```
B = numarul fisierului
C = pagina de RAM de la #C000
DE= numarul octetilor de scris (0 inseamna 64K)
HL= adresa octetilor de scris
```

Iesiri

```
Daca OK
CY = 1
A,DE distruse
altfel
CY = 0
A  = codul erorii
DE = numarul de octeti ramasi nescrisi
registre distruse: BC, HL, IX
```

DOS BYTE READ #0118 (280)

Citeste un octet din fisier, avansind pointerul fisierului. Face teste pentru #1A (26) (soft EOF). Rutina ce utilizeaza DOS

BYTE READ va decide daca codul #1A e de interes (in cazul citirii fisierelor ASCII). Soft-EOF este localizat, memorat si semnalat. Citirea pozitiei EOF va produce o eroare.

Intrari

B = numarul fisierului

Iesiri

Daca e OK si octetul e diferit de #1A
CY = 1
Z = 0
A = distrus
C = contine octetul citit
Daca e OK si octetul e #1A
CY = 1
Z = 1
A = distrus
C = octet citit
altfel
CY = 0
A = codul erorii
C = distrus
registre distruse: B, DE, HL, IX, (C)

DOS BYTE WRITE #011B (283)

Scrie un octet al fisierului pe disc, avansind pointerul de fisier.

Intrari

B = numarul fisierului
C = octetul ce trebuie scris

Iesiri

Daca OK
CY = 1
A = distrus
altfel
CY = 0
A = codul erorii
registre distruse: BC, DE, HL, IX

DOS CATALOG #011E (286)

Creeaza o zona tampon de memorie continind o parte din directorii (sortati). Numele de fisier specifica unitatea de disc, numarul utilizatorului si numele fisierului (posibil ambiguu).

Deoarece marimea zonei directorilor e variabila (putind fi destul de mare), rutina permite catalogarea directorilor pe sectiuni mici. Rutina ce cheama DOS CATALOG paseaza o zona tampon de memorie incarcata cu primul nume de fisier necesar, sau zerouri pentru inceputul directorilor.

Zona tampon e incarcata cu o parte (sau cu tot, daca incape) din directorii sortati in ordine alfabetica (ASCII). Daca mai

sint necesari directori se cheama din nou DOS CATALOG cu zona tampon reinitializata cu numele ultimului fisier din directorii anteriori. Procedura se repeta pina la catalogarea completa a fisierelor.

Sub +3DOS sint permise maxim 64 de directori (fisiere), deci formatul zonei tampon va fi:

```
Zona 0
Zona 1
  .
  .
  .
Zona n
```

Zona 0 trebuie incarcata cu primul nume necesar sub forma **nume.tip**. Zona 1 va contine primul nume de fisier ce se potriveste mai mare decit zona 0 (daca exista). Daca in zona 0 sint zerouri rezulta terminarea actiunii.

Daca zona tampon e prea mica pentru directori, rutina va fi apelata din nou cu zona 0 continind zona n anterioara, pentru a prelua restul de directori.

Formatul zonei i pe 13 octeti este:

```
Octetii 0...7  - numele (ASCII) de fisier aliniat la stinga,
                 completat cu spatii;
Octetii 8...10 - tipul (ASCII) aliniat la stinga si complet-
                 at cu spatii;
Octetii 11, 12 - marimea fisierului in Ko.
```

Marimea fisierului exprima totalitatea spatiilor de disc alocata fisierului, nu neaparat aceiasi cu marimea reala (numar de octeti) a fisierului.

Intrari

```
B = n+1 Marimea zonei tampon in zone ( > 2)
C = filtru
    bit 0 = inchide fisierele sistem (daca e pe 1)
    bit 1...7 =0 (rezervati)
DE= adresa zonei tampon (cu prima zona initializata)
HL= adresa numelui de fisier (sint permisi marcatorii
    si ?)
```

Iesiri

```
Daca e OK
CY = 1
A  = distrus
B  = numarul zonelor (0 la n)
Daca B = n catalogarea trebuie continuata.
altfel
CY = 0
A  = codul erorii
B  = distrus
registre distruse: C, DE, HL, IX
```

DOS FREE SPACE #0121 (289)

Arata cit spatiu este liber (disponibil) pe disc.

Intrari

A = litera unitatii de disc

Iesiri

```
Daca e OK
CY = 1
A  = distrus
HL = spatiu liber (in Ko)
altfel
CY = 0
A  = codul erorii
HL = distrus
registre distruse: BC, DE, IX
```

DOS DELETE #0124 (292)

Sterge un fisier existent, fisierul fiind deschis pe unul din numerele de fisier.

Intrari

HL= adresa numelui de fisier (marcatori * si ? permisi)

Iesiri

```
Daca OK
CY = 1
A  = distrus
altfel
CY = 0
A = codul erorii
registre distruse: BC, DE, HL, IX
```

DOS RENAME #0127 (295)

Schimba numele unui fisier existent, fisierul fiind deschis pe unul din numerele de fisiere. Nu trebuie sa existe un fisier cu numele cel nou. Noul nume trebuie sa specifice (sau implicit) aceiasi unitate de disc ca si vechiul nume.

Intrari

```
DE = adresa noului nume(fara marcatori)
HL = adresa vechiului nume (fara marcatori)
```

Iesiri

```
Daca OK
CY = 1
A  = distrus
altfel
CY = 0
A  = codul erorii
registre distruse: BC, DE, HL, IX
```

DOS BOOT #012A (298)

Rutina incarca si lanseaza in memorie de pe discul A: un sector incarcator (bootstrop sector). Acesta foloseste la incarcarea unor jocuri si a unor sisteme de operare.
Este necesara urmatoarea configuratie de memorie :

```
#C000 (49152) ... RAM 3
#8000 (32768) ... RAM 6
#4000 (16384) ... RAM 7
#0            ... RAM 4
```

Acest sector se va gasi pe disc pe fata 0, pista 0, sectorul 1, si e incarcat in memorie la #FE00 (65024) si lansat de la #FE10 (65040). Intreruperile sint dezafectate iar stiva masina e la #FE00 (65024). Suma tuturor bitilor in acest sector trebuie sa fie 3 (MOD 256). Octetii 0 la 15 ai sectorului contin specificatiile discului.

Intrari

nici una.

Iesiri

```
Daca e OK
Se lanseaza in executie de la #FE10
altfel
CY = 0
A  = codul erorii
registre distruse: BC, DE, HL, IX
```

DOS SET DRIVE #012D (301)

Specifica discul de lucru (adica discul implicat in toate comenzile unde nu se specifica discul de lucru). Discul de lucru initial este A:.
Nu face acces la disc, ci verifica daca exista o unitate de disc de lucru.

Intrari

A = discul ("A",...,"P" si #FF pentru discul initial)

Iesiri

```
Daca e OK
CY = 1
A  = discul de lucru
altfel
CY = 0
A  = codul erorii
registre distruse: BC, DE, HL, IX
```

DOS SET USER #0130 (304)

Specifica aria utilizator de lucru, adica aria implicata in toate operatiile unde nu se specifica numarul (ariei) utilizator. Initial numarul utilizator e 0.

Intrari

A = numar utilizator (0...15 si #FF pentru aria initiala)

Iesiri

Daca e OK
CY = 1
A = numar utilizator
altfel
CY = 0
A = codul erorii
registre distruse: BC, DE, HL, IX.

Subrutine aditionale

DOS GET POSITION #0133 (307)

Furnizeaza pointerul fisierului.

Intrari

B = numarul fisierului

Iesiri

Daca e OK
CY = 1
A = distrus
E,HL = **pointerul** (#0 la #FFFFFF)
(0 la 16777215)
altfel
CY = 0
A = codul erorii
E,HL = distruse
registre distruse: BC, D, IX

DOS SET POSITION #0136 (310)

Initializeaza pointerul, insa nu face acces la disc si nu verifica daca pointerul este > = 8 Mo.

Intrari

B = numarul fisierului
EHL= pointerul fisierului (0 la #FFFFFF)

Iesiri

Daca e OK
CY = 1
A = distrus
altfel
CY = 0
A = codul erorii
registre distruse: BC, DE, HL, IX

DOS GET EOF #0139 (313)

Furnizeaza pozitia de sfirsit de fisier (EOF) adica prima pozitie dupa ultimul octet scris al fisierului. Nu afecteaza pointerul fisierului si nu ia in considerare octetul #1A (soft EOF).

Intrari

B = numarul fisierului

Iesiri

Daca e OK
CY = 1
A = distrus
EHL = pointerul fisierului
altfel
CY = 0
A = codul erorii
EHL = distruse
registre distruse: BC, D, IX

DOS GET 1346 #013C (316)

Furnizeaza locatia curenta a zonei "cache" si a RAM-discului. Paginile 1, 3, 4 si 6 sint considerate ca un sir de 128 de sectoare. (de la 0 la 127), fiecare de 512 octeti. Zona "cache" si RAM-discul ocupa 2 regiuni continue si separate din acest sir de sectoare. Orice sector neutilizat poate fi folosit de utilizator.

Intrari

nici una

Iesiri

D = primul sector pentru memoria "cache"
E = numarul de sectoare ale memoriei "cache"
H = primul sector de RAM-disc
L = numarul de sectoare de RAM-disc
registre distruse : AF, BC, IX

DOS SET 1346 #013F (319)

Reconstruieste zona "cache" si RAM-discul. Rutina e folosita pentru a pune la dispozitia utilizatorului sau a DOS-ului sectoare de zona.

Atentie: orice mutare sau schimbare a marimii pe RAM-disc, sterge tot continutul RAM-discului.

Locarea si marimea pentru "cache" si RAM-disc pot fi specificate separat si sectoarele ramase pot fi utile programatorului.

Zona "cache" si RAM-discul nu trebuie sa-si suprapuna sectoare. +3DOS nu face verificari in acest sens.

Marimile zonelor pot fi mai mici decit cele folosite practic si exista o limita maxima pentru "cache" si o limita minima pentru RAM-disc (4 sectoare).

O marime nula pentru cache va inrautati serios lucrul cu discurile mecanice. Daca e un fisier deschis pe discul M: aceasta rutina va genera eroare.

Intrari

D = primul sector cache
E = numar de sectoare pentru cache
H = primul sector RAM-disc
L = numar de sectoare RAM-disc
(E necesar ca E + L < = 128)

Iesiri

Daca e OK
CY = 1
A = distrus
altfel
CY = 0
A = codul erorii
registre distruse: BC, DE, HL, IX

DOS FLUSH #0142 (322)

Inscrie pe disc antete, date si directori.
Rutina asigura actualizarea discului si poate fi chemata oricind, chiar daca sint deschise fisiere.

Intrari

A = litera unitatii ("A",...,"P")

Iesiri

Daca OK
CY = 1
A = distrus
altfel
CY = 0
A = codul erorii
registre distruse: BC, DE, HL, IX

DOS SET ACCESS #0145 (325)

Schimba modul de acces la un fisier deschis. Aceasta rutina va genera eroare daca fisierul e deschis deja intr-un mod de acces incompatibil sau daca accesul de scriere e necesar pentru un fisier protejat la scriere.

Intrari

B = numarul fisierului
C = modul de acces necesar
Bitii 0, 1, si 2
1 = citire exclusiva
2 = scriere exclusiva
3 = citire si scriere exclusiva
5 = acces multiplu la citire
Bitii 3 ... 7 = 0 (rezervati)

Iesiri

```
        Daca e OK.
        CY = 1
        A  = distrus
altfel
        CY = 0
        A  = codul erorii
        registre distruse: BC, DE, HL, IX
```

DOS SET ATTRIBUTES #0148 (328)

Marcheaza atributele fisierului.
Pot fi sterse sau marcate doar atributele f1 ... f4 si t1 atributele specificate in registrul D, apoi sterge atributele specificate in registrul E (adica E are prioritate).

Intrari

```
        D = atribute de inscris
        Bit  0 = t3    arhiva
             1 = t2    sistem
             2 = t1    protectie la scriere
             3 = f4
             4 = f3
             5 = f2
             6 = f1
        E = atribute de sters
        Bit  0 = t3  arhiva
             1 = t2  sistem
             2 = t1  protectie la scriere
             3 = f4
             4 = f3
             5 = f2
             6 = f1
        HL = adresa numelui fisierului (se permit marcatorii
             si ?)
```

Iesiri

```
        Daca e OK
        CY = 1
        A  = distrus
altfel
        CY = 0
        A = codul erorii
        registre distruse: BC, DE, HL, IX
```

DOS OPEN DRIVE #014B (331)

Deschide intreg discul ca fiind un singur fisier. Se prezinta intreg discul ca fiind un singur fisier indiferent de continutul real de fisiere de pe disc. Rutina poate fi folosita la examinarea/modificarea directorilor, a fisierelor etc. Nu e recomandat sa fie folosita.
Pune pointerul fisierului la valoarea 0. Daca sint fisiere deschise pe disc cu alte numere de fisiere cu acces multiplu la citire, atunci discul poate fi deschis doar cu acces multiplu de

la acest numar de fisier.
Daca exista fisiere deschise pe unitate de alte numere de fisier cu acces exclusiv, atunci discul nu poate fi deschis de acest numar de fisier.

Intrari

A = litera unitatii ("A" ... "P")
B = numar de fisier
C = modul de acces:
1 = exclusiv in citire
2 = exclusiv in scriere
3 = exclusiv in citire/scriere
5 = acces multiplu in citire

Iesiri

Daca e OK
CY = 1
A = distrus
altfel
CY = 0
A = codul erorii
registre distruse: BC, DE, HL

DOS SET MESSAGE #14E (334)

Permite sau interzice semnalarea mesajelor de eroare.
Rutina poate fi folosita in +3DOS pentru cuplarea cu o rutina ALERT proprie a utilizatorului. Cind +3DOS detecteaza o eroare va chema rutina ALERT care va afisa textul mesajului pe care +3DOS il transmite, apoi asteapta utilizatorul sa tasteze o optiune; se va transmite in registrul A valoarea 0, 1, 2 sau codul caracterului tastat ca optiune (depinde de versiunea DOS).

Intrari

A = validare/devalidare
#FF (255) permite mesaje
#00 (0) nu permite
HL= adresa rutinei ALERT

Iesiri

HL = adresa rutinei ALERT (0 daca nu e)
registre distruse: AF, BC, DE, IX
Remarca: daca se inlocuieste rutina ALERT cu alta scrisa de utilizator, intrarile sint conditii pasate rutinei iar iesirile vor fi valori produse de rutina proprie.
Exista 2 rutine ALERT dupa versiunile DOS.

ALERT (pentru DOS V1.0)

Intrari

DE= adresa sirului raspuns (in RAM7) terminat cu #FF

Iesiri

A = codul caracterului optiune
registre distruse: F, BC, DE, HL, IX

ALERT (pentru DOS V1.1)

Intrari

B = numarul erorii
C = unitatea de disc ("A" ... "P")
D = pista logica
E = sectorul logic
HL= adresa mesajului de eroare (in RAM 7) terminat cu #FF

Iesiri

A=raspunsul
0 = abandoneaza comanda
1 = reincearca executia
2 = ignora eroarea
registre distruse: F, BC, DE, HL, IX

Remarca: aceasta rutina e mai flexibila, permitind emiterea de mesaje non-standard la versiunile V1.1 si ulterioare. Se recomanda la folosirea unei rutine ALERT proprie sa se faca verificarea versiunii si in functie de aceasta sa se genereze actiunea corespunzatoare.

DOS REF XDPB #0151 (337)

Furnizeaza adresa blocului extins de parametri ai discului (XDPB) pentru unitatea de disc respectiva.

Intrari

A=litera unitatii ("A"..."P")

Iesiri

Daca e OK
CY = 1
A = distrus
IX = adresa XDPB
altfel
CY = 0
A = codul erorii
IX = distrus
registre distruse: BC, DE, HL

DOS MAP B #0154 (340)

Simuleaza unitatea de disc logic **B:** pe unitatile de disc fizic **0** sau **1**. Nu se va folosi daca **B:** are fisiere deschise.

Daca se foloseste **B:** pe unitatea **0** atunci de cite ori se va apela unitatea **0** se verifica daca discul logic e corect. Daca nu este, se cheama CHANGE DISK, rutina ce va dialoga cu utilizatorul.

Daca se foloseste **B:** pe unitatea **1** si unitatea **1** nu exista

atunci **B:** este dezafectat.

Intrari

C =unitatea fizica (0 sau 1)
HL=adresa rutinei CHANGE DISK (daca C = 0)

Iesiri

Daca e OK
CY = 1
A = distrus
HL = adresa rutinei CHANGE DISK (0, daca nu exista)
altfel
CY = 0
A,HL= distruse
registre distruse: BC, DE, IX

Remarca: Daca se va folosi o rutina CHANGE DISK a utilizatorului, atunci intrarile vor fi conditii pasate rutinei CHANGE DISK, iar iesirile sint registre ce pot fi distruse.

CHANGE DISK

Semnaleaza utilizatorului sa schimbe discul in unitatea 0, si asteapta cu utilizatorul sa valideze schimbarea.

Intrari

A= discul logic ("A"..."P")
HL=adresa mesajului (in RAM7) terminat cu #FF

Iesiri

registre distruse: AF, BC, DE, HL, IX

Subrutine de baza

Aceste subrutine sint folosite in lucrul direct cu unitatile de disc. Sint folosite de Tim-S Plus doar unitatile fizice 0 si 1, unitatea 0 fiind unitatea logica **A:** iar unitatea 1 fiind unitatea logica **B:**. E posibila simularea ambelor unitati logice **A:** si **B:** pe unitatea fizica 0.

Exceptind rutina DD INTERFACE nici o rutina nu poate fi apelata daca interfata fizica cu discul nu e prezenta.

Toate rutinele cer validarea intreruperilor la apel, validare ce ramine valabila si la revenirea din rutine.

DD INTERFACE #0157 (343)

Verifica prezenta interfetei fizice cu discurile. Bitul 4 al variabilei de sistem FLAGS3 contine aceasta informatie.

Intrari

nici una

Iesiri

Daca interfata e prezenta
CY =1
altfel
CY =0

registre distruse: A, BC, DE, HL, IX

DD INIT #015A (346)

Initializeaza unitatea de disc.

Intrari

nici una

Iesiri - registre distruse: AF, BC, DE, HL, IX

DD SETUP #015D (349)

Preia parametrii discului si transmite o comanda specifica.
Formatul blocului de parametri:
Octet 0 - timp de pornire motor;
1 - timp de oprire motor;
2 - timp de terminare scriere;
3 - timp de stabilizare cap;
4 - pasul;
5 - timp descarcare cap;
6 - timp de incarcare cap.

Intrari

HL = adresa blocului de parametri

Iesiri

registre distruse: AF, BC, DE, HL, IX

DD SET RETRY #0160 (352)

Incarca numaratorul de incercari. Valoarea 1 va incerca efectuarea operatiei o singura data, adica fara repetari.

Intrari

A = numarul de incercari (>=1)

Iesiri

registre distruse: AF, BC, DE, HL, IX

DD READ SECTOR #0163 (355)

Citeste un sector.

Intrari

```
          B = Pagina RAM de la #C000 la #FFFF
          C = unitatea fizica de disc (0 sau 1)
          D = pista logica (baza = 0)
          E = sector logic (baza = 0)
          HL= adresa tamponului
          IX= adresa XDPB
```

Iesiri

```
          Daca e OK
          CY = 1
          A  = distrus
altfel
          CY = 0
          A  = codul erorii
          registre distruse: BC, DE, HL, IX
```

DD WRITE SECTOR #0166 (358)

Scrie un sector.

Intrari

```
          B = pagina RAM de la #C000
          C = unitatea de disc (0 sau 1)
          D = pista logica
          E = sector logic
          HL= adresa tamponului
          IX= adresa XDPB
```

Iesiri

```
          Daca e OK
          CY = 1
          A  = distrus
altfel
          CY = 0
          A  = codul erorii
          registre distruse: BC, DE, HL, IX
```

DD CHECK SECTOR #0169 (361)

Verifica sectorul de pe disc cu zona din memorie. Daca pe sector sau in memorie este #FF atunci nu verifica.

Intrari

```
B = pagina RAM de la #C000
C = unitatea (0 sau 1)
D = pista logica
E = sector logic
HL= adresa copiei sectorului
IX= adresa XDPB
```

Iesiri

```
Daca este OK si identitate
     CY = 1
```

```
          Z  = 1
          A  = distrus
     Daca este OK si nu-i identitate
          CY = 1
          Z  = 0
          A  = distrus
altfel
          CY = 0
          A  = codul erorii
          registre distruse: BC, DE, HL, IX
```

DD FORMAT #016C (364)

Formateaza o pista.
E descris pe 4 octeti fiecare sector:
C = numarul pistei (0 ... 39)
H = numarul capului (0)
R = numarul sectorului (0 ... 9)
N = Log2 (marime sector)- 7, (adica 2 pentru 512)

Intrari

B = pagina RAM de la #C000
C = unitatea fizica (0 sau 1)
D = pista logica
E = octet de umplere #E5 (229)
HL= adresa buffer-ului de format
IX= adresa XDPB

Iesiri

```
Daca e OK
          CY = 1
          A  = distrus
altfel
          CY = 0
          A  = codul erorii
          registre distruse: BC, DE, HL, IX
```

DD READ ID #016F (367)

Citeste un identificator de sector.

Intrari

```
          C = unitatea (0 sau$1)
          D = pista logica (baza =0)
          IX= adresa XDPB
```

Iesiri

```
Daca e OK
          CY = 1
          A  = numarul sectorului cu identificator
altfel
          CY = 0
          A  = codul erorii
intodeauna
          HL = adresa buffer-ului rezultat in RAM7
```

registre distruse: BC, DE, IX

DD TEST UNSUITABLE #0172 (370)

Verifica daca discul e potrivit pentru a scrie pe el. Un disc pista simpla nu va merge intr-o unitate pista dubla si invers.

Intrari

C = unitatea (0 sau 1)
IX= adresa XDPB

Iesiri

Daca e potrivit
CY = 1
A = distrus
altfel
CY = 0
A = codul erorii
registre distruse: BC, DE, HL, IX

DD LOGIN #0175 (373)

Preia un disc nou, initializat XDPB.

Intrari

C = unitatea (0 sau 1)
IX= adresa destinatiei XDPB

Iesiri

Daca e OK
CY = 1
A = tipul discului
DE = marimea vectorului de alocare
HL = marimea tabelei
altfel
CY = 0
A = codul erorii
DE,HL = distruse
registre distruse: BC, IX, (DE, HL)

DD SEL FORMAT #0178 (376)

Initializeaza un XDPB pentru formatul standard.

Intrari

A = tipul discului
0 - format +3 (AMSTRAD PCW-DD SS ST)
1 - AMSTRAD CPC disc sistem
2 - AMSTRAD CPC disc date
3 - AMSTRAD PCW - DD DS DT
alte valori = eroare
IX = adresa XDPB

Iesiri

Daca e OK
CY = 1
A = tipul discului
DE = marimea vectorului de alocare (2 biti)
HL = marime tabela
altfel
CY = 0
A = codul erorii
DE,HL = distrusi
registre distruse: BC, HL, (DE)

DD ASK 1 #017B (379)

Verifica daca unitatea 1 este prezenta. +3BASIC memoreaza aceasta informatie in bitul 5 al variabilei de sistem FLAGS3.
Porneste motorul unitatii si supervizeaza starea unitatii. Daca unitatea 1 nu este pregatita si e protejata la scriere, atunci ea e absenta si motorul va fi oprit.

Intrari

Nici una.

Iesiri

Daca unitatea 1 e prezenta
CY = 1
altfel
CY = 0
registre distruse: A, BC, DE, HL, IX

DD DRIVE STATUS #017E (382)

Furnizeaza starea unitatii.

Intrari

C = unitatea/capul
bitii 0 si 1 => unitatea
bit 2 = capul
bitii 3...7 = 0

Iesiri

A = registrul de stare
registre distruse: F, BC, DE, HL, IX

DD EQUIPMENT #0181 (385)

Furnizeaza informatii despre disc.
Informatii despre piste pot fi date doar daca tipul discului a fost identificat.

Intrari

C = unitatea (0 sau 1)

IX= adresa XDPB

Iesiri

A = informatii
Bitii 0,1 = informatii despre fata (fetele) discului
0 = necunoscuta
1 = o fata
2 = doua fete
bitii 2,3 = pista
0 = necunoscuta
1 = pista simpla
2 = pista dubla
registre distruse: F, BC, DE, HL, IX

DD ENCODE #0184 (388)

Furnizeaza adresa rutinei ENCODE de protectie la copiere. Discurile protejate la copiere au anumite piste si sectoare codate. Inainte de a accesa aceste discuri rutina ENCODE e chemata sa codeze numele fizice ale pistelor si sectoarelor, care trebuie sa se potriveasca cu identificatorul de sector. Pistele 0, 1 si 2 nu trebuie codate.

Intrari

A = validare/devalidare
#00 fara codare
#FF cu codare
HL= adresa rutinei ENCODE

Iesiri

HL = adresa rutinei ENCODE (daca nu exista, 0)
registre distruse: AF, BC, DE, IX

Remarca: Daca utilizatorul isi face o rutina ENCODE proprie intrarile vor fi conditii pasate acestor rutine, iar iesirile vor fi valori produse de rutina si registre distruse.

ENCODE

Intrari

C = unitatea/fata
bitii 0, 1 = unitatea
bit 2 = fata
bitii 3...7 = 0
D = pista fizica
E = sector fizic
IX= adresa DPB

Iesiri

D = numarul codat al pistei fizice
E = numarul codat al sectorului fizic
registre distruse: AF

DDL XDPB #0187 (391)

Initializeaza XDPB pentru un anumit format.

Intrari

```
IX = adresa destinatiei XDPB
HL = adresa sursei specificatiilor de disc
```

Iesiri

```
Daca e OK
     CY = 1
     A  = tipul discului
     DE = marimea vectorului de alocare
     HL = marime tabela
Daca formatul e nepotrivit
     CY = 0
     A  = codul erorii
     DE,HL = distrus
     registre distruse: BC, IX
```

DDL DPB #018A (394)

Initializeaza DPB pentru un anumit format.

Intrari

```
IX = adresa destinatiei DPB
HL = adresa specificatiilor de disc
```

Iesiri

```
Daca e OK
     CY = 1
     A  = tipul de disc
     DE = marime vector de alocare
     HL = marime tabela
Daca formatul este nepotrivit
     CY = 0
     A  = codul erorii
     DE,HL = distruse
     registre distruse:BC, IX
```

DDL SEEK #018D (397)

Cauta o pista, daca greseste reincearca.

Intrari

```
C = unitate/cap
    bitii 0, 1  = unitatea
    bit 2       = capul
    bitii 3...7 = 0
D = pista
IX= adresa XDPB
```

Iesiri

```
Daca e OK
     CY = 1
     A  = distrus
```

altfel

CY = 0
A = codul erorii
registre distruse: BC, DE, HL, IX

DL READ #0190 (400)

Comanda de citire de baza. Citeste date, sterse sau nu. Citeste o pista. Blocul parametrilor are formatul:

Octet 0 - pagina RAM de la #C000
Octetii 1,2 - adresa buffer-ului
Octetii 3, 4 - numarul de octeti ce se transfera
Octet 5 - numarul octetilor de comanda
Octet 6... - octeti de comanda

Scrie comanda, citeste date si rezultate cu motorul pornit.

Intrari

HL = adresa blocului de parametri

Iesiri

HL = adresa buffer-ului rezultat in RAM7
registre distruse: AF, BC, DE, IX

DDL WRITE #0193 (403)

Comanda de scriere de baza. Scrie date sterse sau nu. Formateaza o pista.
Blocul parametrilor are formatul:

Octet 0 - Pagina RAM de la #C000
Octetii 1...2 - adresa buffer-ului
Octetii 3...4 - numar de octeti transferati
Octet 5 - numar de octeti de comanda
Octeti 6... - Octeti de comanda

Scrie comanda si date. Citeste rezultatele cu motorul pornit.

Intrari

HL = adresa blocului de parametri

Iesiri

HL = adresa buffer-ului in RAM7
registre distruse: AF, BC, DE, IX

DDL ON MOTOR #0196 (406)

Porneste motorul, folosind timpul de asteptare pus de DD SETUP.

Intrari

nici una

Iesiri

registre distruse: AF, BC, DE, HL, IX

DDLT OFF MOTOR #0199 (409)

Porneste numaratorul de oprire motor.

Intrari

nici una

Iesiri

registre distruse: AF, BC, DE, HL, IX

DDL OFF MOTOR #19C (412)

Opreste motorul.

Intrari

nici una.

Iesiri

registre distruse: AF, BC, DE, HL, IX.

4.29 Setul de caractere al calculatorului Tim-S Plus

Din sumar:

Coduri de control
Caractere
Mnemonice ale codului masina pe Z80

Va prezentam acum setul complet de caractere al calculatorului +3, cu codurile in zecimal si hexazecimal, si daca consideram aceste coduri, coduri ale instructiunilor din codul masina Z80, pe cele trei coloane din dreapta veti observa mnemonicele codului Z80. Anumite instructiuni pentru Z80 sint compuse si incep cu #CB sau #ED: acestea sint expuse in ultimele doua coloane din dreapta.

Cind un caracter este schimbat de la 48 BASIC la +3, versiunea de la 48 BASIC este pusa in paranteze dupa cea de la +3.

COD	CARACTER	HEXA	Z80	DUPA #CB	DUPA #ED
0	neutilizat	00	nop	rlc b	
1	"	01	ld bc,NN	rlc c	
2	"	02	ld (bc),a	rlc d	
3	"	03	inc bc	rlc e	
4	"	04	inc b	rlc h	
5	"	05	dec b	rlc l	
6	PRINT virgula	06	ld b,N	rlc(hl)	

7	[EDIT]	07	rlca	rlc a	
8	cursor stinga	08	ex af,af'	rrc b	
9	cursor dreapta	09	add hl,bc	rrc c	
10	cursor jos	0A	ld a,(bc)	rrc d	
11	cursor sus	0B	dec bc	rrc e	
12	[DELETE]	0C	inc c	rrc h	
13	[ENTER]	0D	dec c	rrc l	
14	numar	0E	ld c,N	rrc (hl)	
15	neutilizat	0F	rrca	rrc a	
16	control INK	10	djnz DIS	rl b	
17	control PAPER	11	ld de,NN	rl c	
18	control FLASH	12	ld (de),a	rl d	
19	control BRIGHT	13	inc de	rl e	
20	control INVERSE	14	inc d	rl h	
21	control OVER	15	dec d	rl l	
22	control AT	16	ld d,N	rl (hl)	
23	control TAB	17	rl a	rl a	
24	neutilizat	18	jr DIS	rr b	
25	"	19	add hl,de	rr c	
26	"	1A	ld a,(de)	rr d	
27	"	1B	dec de	rr e	
28	"	1C	inc e	rr h	
29	"	1D	dec e	rr l	
30	"	1E	ld e, N	rr (hl)	
31	"	1F	rra	rr a	
32	spatiu	20	jrnz,DIS	sla b	
33	!	21	ld hl,NN	sla c	
34	"	22	ld (NN),hl	sla d	
35	#	23	inc hl	sla e	
36	$	24	inc h	sla h	
37	%	25	dec h	sla l	
38	&	26	ld h,N	sla (hl)	
39	'	27	daa	sla a	
40	(	28	jrz,DIS	sra b	
41	)	29	add hl,hl	sra c	
42	*	2A	ld hl,(NN)	sra d	
43	+	2B	dec hl	sra e	
44	,	2C	inc l	sra h	
45	-	2D	dec l	sra l	
46	.	2E	ld l,N	sra (hl)	
47	/	2F	cp l	sra a	
48	0	30	jr nc,DIS		
49	1	31	ld sp,NN		
50	2	32	ld (NN),a		
51	3	33	inc sp		
52	4	34	inc (hl)		
53	5	35	dec (hl)		
54	6	36	ld (hl),N		
55	7	37	scf		
56	8	38	jr c,DIS	srl b	
57	9	39	add hl,sp	srl c	
58	:	3A	ld a,(NN)	srl d	
59	;	3B	dec sp	srl e	
60	<	3C	inc a	srl h	
61	=	3D	dec a	srl l	
62	>	3E	ld a,N	srl (hl)	
63	?	3F	ccf	srl a	
64	@	40	ld b,b	bit 0,b	in b,(c)
65	A	41	ld b,c	bit 0,c	out (c),b
66	B	42	ld b,d	bit 0,d	sbc hl,bc
67	C	43	ld b,e	bit 0,e	ld (NN),bc

68	D	44	ld b,h	bit 0,h	neg
69	E	45	ld b,l	bit 0,l	ret n
70	F	46	ld b,(hl)	bit 0,(hl)	im 0
71	G	47	ld b,a	bit 0,a	ld i,a
72	H	48	ld c,b	bit 1,b	in c,(c)
73	I	49	ld c,c	bit 1,c	out (c),c
74	J	4A	ld c,d	bit 1,d	adc hl,bc
75	K	4B	ld c,e	bit 1,e	ld bc,(NN)
76	L	4C	ld c,h	bit 1,h	
77	M	4D	ld c,l	bit 1,l	ret i
78	N	4E	ld c,(hl)	bit 1,(hl)	
79	O	4F	ld c,a	bit 1,a	ld r,a
80	P	50	ld d,b	bit 2,b	in d,(c)
81	Q	51	ld d,c	bit 2,c	out (c),d
82	R	52	ld c,d	bit 2,d	sbc hl,de
83	S	53	ld d,e	bit 2,e	ld (NN),de
84	T	54	ld d,h	bit 2,h	
85	U	55	ld d,l	bit 2,l	
86	V	56	ld d,(hl)	bit 2,(hl)	im 1
87	W	57	ld d,a	bit 2,a	ld a,i
88	X	58	ld e,b	bit 3,b	in e,(c)
89	Y	59	ld e,c	bit 3,c	out (c),e
90	Z	5A	ld e,d	bit 3,d	adc hl,de
91	[	5B	ld e,e	bit 3,e	ld de,(NN)
92	\	5C	ld e,h	bit 3,h	
93	]	5D	ld e,l	bit 3,l	
94	^	5E	ld e,(hl)	bit 3,(hl)	im 2
95	-	5F	ld e,a	bit 3,a	ld a,r
96	`	60	ld h,b	bit 4,b	in h,(c)
97	a	61	ld h,c	bit 4,c	out (c),h
98	b	62	ld h,d	bit 4,d	sbc hl,hl
99	c	63	ld h,e	bit 4,e	ld (NN),hl
100	d	64	ld h,h	bit 4,h	
101	e	65	ld h,l	bit 4,l	
102	f	66	ld h,(hl)	bit 4,(hl)	
103	g	67	ld h,a	bit 4,a	rrd
104	h	68	ld l,b	bit 5,b	in l,(c)
105	i	69	ld l,c	bit 5,c	out (c),l
106	j	6A	ld l,d	bit 5,d	adc hl,hl
107	k	6B	ld l,e	bit 5,e	ld hl,(NN)
108	l	6C	ld l,h	bit 5,h	
109	m	6D	ld l,l	bit 5,l	
110	n	6E	ld l,(hl)	bit 5,(hl)	
111	o	6F	ld l,a	bit 5,a	rl d
112	p	70	ld (hl),b	bit 6,b	in l,c
113	q	71	ld (hl),c	bit 6,c	
114	r	72	ld (hl),d	bit 6,d	sbc hl,sp
115	s	73	ld (hl),e	bit 6,e	ld (NN),sp
116	t	74	ld (hl),h	bit 6,h	
117	u	75	ld (hl),l	bit 6,l	
118	v	76	halt	bit 6,(hl)	
119	w	77	ld (hl),a	bit 6,a	
120	x	78	ld a,b	bit 7,b	in a,(c)
121	y	79	ld a,c	bit 7,c	out (c),a
122	z	7A	ld a,d	bit 7,d	adc hl,sp
123	{	7B	ld a,e	bit 7,e	ld sp,(NN)
124	!	7C	ld a,h	bit 7,h	
125	}	7D	ld a,l	bit 7,l	
126	~	7E	ld a,(hl)	bit 7,(hl)	
127		7F	ld a,a	bit 7,a	
128	semigrafic	80	add a,b	res 0,b	

129	"	81	add a,c	res 0,c	
130	"	82	add a,d	res 0,d	
131	"	83	add a,e	res 0,e	
132	"	84	add a,h	res 0,h	
133	"	85	add a,l	res 0,l	
134	"	86	add a,(hl)	res 0,(hl)	
135	"	87	add a,a	res 0,(a)	
136	"	88	adc a,b	res 1,b	
137	"	89	adc a,c	res 1,c	
138	"	8A	adc a,d	res 1,d	
139	"	8B	adc a,e	res 1,e	
140	"	8C	adc a,h	res 1,h	
141	"	8D	adc a,l	res 1,l	
142	"	8E	adc a,(hl)	res 1,(hl)	
143	"	8F	adc a,a	res 1,a	
144	(a) udg	90	sub b	res 2,b	
145	(b) udg	91	sub c	res 2,c	
146	(c) udg	92	sub d	res 2,d	
147	(d) udg	93	sub e	res 2,e	
148	(e) udg	94	sub h	res 2,h	
149	(f) udg	95	sub l	res 2,l	
150	(g) udg	96	sub (hl)	res 2,(hl)	
151	(h) udg	97	sub a	res 2,a	
152	(i) udg	98	sbc a,b	res 3,b	
153	(j) udg	99	sbc a,c	res 3,c	
154	(k) udg	9A	sbc a,d	res 3,d	
155	(l) udg	9B	sbc a,e	res 3,e	
156	(m) udg	9C	sbc a,h	res 3,h	
157	(n) udg	9D	sbc a,l	res 3,l	
158	(o) udg	9E	sbc a,(hl)	res 3,(hl)	
159	(p) udg	9F	sbc a,a	res 3,a	
160	(q) udg	A0	and b	res 4,b	ldi
161	(r) udg	A1	and c	res 4,c	cpi
162	(s) udg	A2	and d	res 4,d	ini
163	SPECTRUM (t)	A3	and e	res 4,e	out i
164	PLAY (u)	A4	and h	res 4,h	
165	RND	A5	and l	res 4,l	
166	INKEY$	A6	and (hl)	res 4,(hl)	
167	PI	A7	and a	res 4,a	
168	FN	A8	xor b	res 5,b	ldd
169	POINT	A9	xor c	res 5,c	cpd
170	SCREEN$	AA	xor d	res 5,d	ind
171	ATTR	AB	xor e	res 5,e	out d
172	AT	AC	xor h	res 5,h	
173	TAB	AD	xor l	res 5,l	
174	VAL$	AE	xor (hl)	res 5,(hl)	
175	CODE	AF	xor a	res 5,a	
176	VAL	B0	or b	res 6,b	ldir
177	LEN	B1	or c	res 6,c	cpir
178	SIN	B2	or d	res 6,d	inir
179	COS	B3	or e	res 6,e	otir
180	TAN	B4	or h	res 6,h	
181	ASN	B5	or l	res 6,l	
182	ACS	B6	or (hl)	res 6,(hl)	
183	ATN	B7	or a	res 6,a	
184	LN	B8	cp b	res 7,b	lddr
185	EXP	B9	cp c	res 7,c	cpdr
186	INT	BA	cp d	res 7,d	indr
187	SQR	BB	cp e	res 7,e	otdr
188	SGN	BC	cp h	res 7,h	
189	ABS	BD	cp l	res 7,l	

190	PEEK	BE	cp (hl)	res 7,(hl)
191	IN	BF	cp a	res 7,a
192	USR	C0	ret nz	set 0,b
193	STR$	C1	pop bc	set 0,c
194	CHR$	C2	jp nz,NN	set 0,d
195	NOT	C3	jp NN	set 0,e
196	BIN	C4	call nz,NN	set 0,h
197	OR	C5	push bc	set 0,l
198	AND	C6	add a,N	set 0,(hl)
199	<=	C7	rst 0	set 0,a
200	>=	C8	ret z	set 1,b
201	<>	C9	ret	set 1,c
202	LINE	CA	jp z,NN	set 1,d
203	THEN	CB		set 1,e
204	TO	CC	call z,NN	set 1,h
205	STEP	CD	call NN	set 1,l
206	DEF FN	CE	adc a,N	set 1,(hl)
207	CAT	CF	rst 8	set 1,a
208	FORMAT	D0	ret nc	set 2,b
209	MOVE	D1	pop de	set 2,c
210	ERASE	D2	jp nc,NN	set 2,d
211	OPEN#	D3	out (N),a	set 2,e
212	CLOSE#	D4	call nc,NN	set 2,h
213	MERGE	D5	push de	set 2,l
214	VERIFY	D6	sub N	set 2,(hl)
215	BEEP	D7	rst 16	set 2,a
216	CIRCLE	D8	ret c	set 3,b
217	INK	D9	exx	set 3,c
218	PAPER	DA	jp c,NN	set 3,d
219	FLASH	DB	in a,(N)	set 3,e
220	BRIGHT	DC	call c,NN	set 3,h
221	INVERSE	DD	prefix instr cu (ix)	set 3,l
222	OVER	DE	sbc a,N	set 3,(hl)
223	OUT	DF	rst 24	set 3,a
224	LPRINT	E0	ret po	set 4,b
225	LLIST	E1	pop hl	set 4,c
226	STOP	E2	jp po,NN	set 4,d
227	READ	E3	ex (sp),hl	set 4,e
228	DATA	E4	call po,NN	set 4,h
229	RESTORE	E5	push hl	set 4,l
230	NEW	E6	and N	set 4,(hl)
231	BORDER	E7	rst 32	set 4,a
232	CONTNUE	E8	ret pe	set 5,b
233	DIM	E9	jp (hl)	set 5,c
234	REM	EA	jp pe,NN	set 5,d
235	FOR	EB	ex de,hl	set 5,e
236	GO TO	EC	call pe,NN	set 5,h
237	GOSUB	ED		set 5,l
238	INPUT	EE	xor N	set 5,(hl)
239	LOAD	EF	rst 40	set 5,a
240	LIST	F0	ret p	set 6,b
241	LET	F1	pop af	set 6,c
242	PAUSE	F2	jp p,NN	set 6,d
243	NEXT	F3	di	set 6,e
244	POKE	F4	call p,NN	set 6,h
245	PRINT	F5	push af	set 6,l
246	PLOT	F6	or N	set 6,(hl)
247	RUN	F7	rst 48	set 6,a
248	SAVE	F8	ret m	set 7,b
249	RANDOMIZE	F9	ld sp,hl	set 7,c

```
250  IF          FA   jp m,NN       set 7,d
251  CLS         FB   ei            set 7,e
252  DRAW        FC   call m,NN     set 7,h
253  CLEAR       FD   prefix instr  set 7,l
                      cu (iy)
254  RETURN      FE   cp N          set 7,(hl)
255  COPY        FF   rst 56        set 7,a
```

4.30 Mesaje de eroare

Mesajele de eroare sint vizibile in partea de jos a ecranului si apar in cazul opririi din executie (accidentale sau normale) a unui program **BASIC.**

Forma unui mesaj de eroare este:

X TTT...TTT NNNN:III, cu semnificatiile:

```
X          - codul erorii;
TTT...TTT  - mesaj explicativ;
NNNN       - numarul liniei BASIC;
III        - numarul instructiunii din linia BASIC.
```

Mesajele contin informatii referitoare la cauza opririi, numarul liniei **BASIC** si numarul instructiunii din linia **BASIC** unde s-a produs oprirea din executie.(O comanda este considerata ca linie **BASIC** cu numarul 0).

In tabel sint prezentate toate mesajele de eroare: codul, mesajul cu **explicatii** si circumstantele aparitiei.

Pentru unele din mesajele de eroare referitoare la disc nu este dat codul erorii, ele fiind urmate de mesajul cu optiunile: (pe tabel se marcheaza RIC la codul erorii):

Retry, Ignore or Cancel?

Daca se alege din meniul RIC optiunea C atunci se va afisa un al doilea mesaj de eroare de obicei similar cu primul.

Comanda CONTINUE reia executia programului **BASIC** de la linia si instructiunea existenta in mesajul de eroare de la oprirea executie. Exceptie fac mesajele 0,9 si D.

COD	MESAJ SI EXPLICATII	CIRCUMSTANTE
0	**OK** Executie cu succes.Salt la o linie mai mare decit liniile existente in program.	normale
1	**NEXT without FOR** Variabila de control a buclei FOR nu exista nu a fost creata de o instructiune FOR) dar exista o variabila simpla de acelasi nume.	NEXT
2	**Variabile not found** Variabila simpla e folosita inainte de atribuire cu LET, READ sau INPUT, sau inaintea de incarcarea de pe disc sau caseta, sau de creare cu FOR.Variabila tablou este folosita înainte de dimensiunea cu DIM sau incarcarea de pe disc ori caseta.	diverse
3	**Subscript wrong** Indicele depaseste dimensiunea variabilei. Numar de indici diferit de cel din dimensio-	Variabile tablou Subsiruri

	narea variabilei. Daca indicele e<0 sau >65535, rezulta eroare B.	
4	**OUT of memory** Insuficient spatiu de memorie pentru ceea ce se intentioneaza. Daca pare blocat calculatorul, se va sterge linia de comanda si inca 2-3 din program (punindu-le apoi la loc) pentru a crea spatiu de lucru. Se poate folosi si CLEAR.	LET, INPUT, FOR, DIM, GOSUB, LOAD MERGE La evaluare expresii.
5	**OUT of screen** Instructiunea INPUT a incercat sa genereze mai mult de 23 de linii. S-a utilizat PRINT AT 22,X.Parametri necorespunzatori la DRAW CIRCLE	INPUT PRINT AT DRAW CIRCLE
6	**Number too big** Calculele au generat un numar mai mare decit 10.E38.	Calcule aritmetice
7	**RETURN without GOSUB** S-au executat mai multe instructiuni RETURN decit instructiuni GOSUB.	RETURN
8	**End of file**	Operatii cu discul compatibil INTERFACE 1
9	**STOP statement** S-a executat instructiunea STOP. CONTINUE reia executia cu urmatoarea instructiune dupa STOP.	STOP
A	**Invalid argument** Functia are argument necorespunzator.	SQR, LN, ASN, ACS, USR cu argument sir
B	**Integer out of range** Prin rotunjirea la cel mai apropiat intreg a argumentului in virgula mobila, intregul rezultat nu e in domeniul permis de instructiune/functie(vezi si eroare 3).	RUN, RANDOMIZE POKE,DIM, GOTO, LIST, GOSUB, PLOT LLIST, CHR$ PAUSE, USR, PEEK cu argument numeric
C	**Nonsense in BASIC** Textul argumentului nu e expresie valida pentru functie. Argumentul functiei sau comenzii este afara din domeniu.	VAL, VAL$
D	**BREAK...CONT repeats** S-a tastat BREAK pe timpul lucrului cu periferice. CONTINUE repeta instructiunea (vezi mesaj L).	LOAD, SAVE VERIFY, MERGE Daca apare mesaj scroll ? si se tasteaza N, (BREAK) SPACE
E	**Out of DATA** S-a executat READ dupa terminarea listei DATA	READ
F	**Invalid filename** Numele de fisier utilizat de SAVE este vid	SAVE

	sau la caseta >10 caractere).	
G	**No room for line** Insuficient spatiu in memorie pentru editarea liniei BASIC.	Editare linie BASIC
H	**STOP in INPUT** Datele pentru INPUT incep cu simbolul STOP. Comanda CONTINUE repeta instructiunea INPUT.	INPUT
I	**FOR without NEXT** Bucla FOR ce nu se poate executa (ex. FOR n=1 TO 0) si nu exista instructiunea NEXT corespunzatoare.	FOR
J	**Invalid I/O device** Incercare de transfer caractere la/de la un canal de periferice ce nu admite sensul cerut (ex.: INPUT#2, a$).	OPEN#, CLOSE#, INPUT#, Operatii pe canale
K	**Invalid colour** Numarul specificat nu este corespunzător.	INK, PAPER, BORDER, FLASH, OVER, BRIGHT, INVERSE
L	**BREAK into program** S-a tastat BREAK. Numarul de linie si de instructiune din mesajse refera la instructiunea deja existenta. CONTINUE reia executia cu urmatoarea instructiune.	oricind
M	**RAMTOP no good** Numar prea mare sau prea mic pentru RAMTOP.	CLEAR, RUN,
N	**Statement lost** Salt la o instructiune care nu exista.	RETURN, NEXT CONTINUE
O	**Invalid Stream** Lucrul cu un canal care nu a fost deschis (OPEN#/, sau numarul de canal nu este in domeniul 0...15.	INPUT#, OPEN#, PRINT#
P	**FN without DEF** Functia nu a fost definita (DEFM) in program.	FN
Q	**Parameter error** Numar de parametri necorespunzator sau tip nepotrivit de parametru.	FN
R	**Tape Loading error** Fisierul din care s-a inceput citirea sau verificarea nu mai poate fi citit (inregistrare compromisa).	VERIFY, LOAD, MERGE
d	**Too many brackets** Numar necorespunzator de paranteze la unul argumentele instructiunii.	PLAY
j	**Invalid baud rate** Rata transmisiei pentru RS 232 a fost pusa zero.	FORMAT LINE
k	**Invalid note name** Nota sau comanda nerecunoscuta de instructiunea PLAY.	PLAY
l	**Number too big** Parametrul comenzii este cu un ordin de marime prea mare.	PLAY
m	**Note out of range** Nota muzicala afara din registrul sonor posibil.	PLAY
n	**Out of range**	PLAY

	Parametrul pentru o comanda e prea mic sau prea mare. Daca eroarea este mult prea mare rezulta eroare 1.	
o	**Too many tied notes** S-a incercat legarea (_) prea multor note impreuna .	PLAY
	Bad file name Nume de fisier necorespunzator folosit la lucrul cu discul.	CAT, COPY, ERASE, SAVE FORMAT, LOAD, MOVE, MERGE
	Bad parameters +3DOS a primit de la interpretorul de BASIC o valoare afara din domeniul admis.	Improbabil
RIC	**CRC data error** Suma de control (CRC) incorecta pentru un sector (daca discul afost compromis mecanic sau magnetic).	CAT, COPY, ERASE, SAVE, MOVE LOAD, MERGE
	Code lenght error Incercarea de incarcare a unui fisier cod (CODE) de pe disc, care e mai lung decit valoarea data in comanda LOAD.	LOAD..CODE
	Destination cannot be wild Comanda COPY nu admite ca destinatie un fisier definit cu marcatorul * atunci cind fisierul sursa e definit cu marcatorul * . Daca marcatorul sursa este definit cu marcatorul * atunci ca destinatia se admite doar litera ce specifica discul(:A sau :B sau :M)	COPY...TO
	Destination must be drive Numele fisierului sursa in comanda COPY contine marcatorul * si destinatia este un nume de fisier. Destinatia poate fi doar o litera de disc (:A sau :M)	COPY...TO
	Directory full S-a incercat crearea celui de-al 65-lea fisier pe disc. Sint posibile doar 64.	COPY, SAVE
	Disk full In SAVE sau COPY s-a ocupat tot spatiul disponibil pe disc. In COPY orice fisier partial scris este sters. In SAVE partea scrisa deja ramine pe disc si ea trebuie stearsa deoarece nu poate fi utilizata.	COPY,SAVE
RIC	**Disk has been changed** In cursul executiei unei comenzi discul prezentat la inceputul executiei e schimbat cu altul.	CAT, COPY, ERASE, MOVE, SAVE LOAD, MERGE
	Daca un program in cod a deschis un fisier pe disc, apoi discul este schimbat, la incercarea unei comenzi de lucru cu discul, va rezulta mesajul de mai sus.	
	Disk is not bootable S-a incercat incarcarea unui program autoincarcator de pe un disc ce nu are sector autoincarcator.	LOAD "*"
RIC	**Disk is write protected** S-a incercat srierea unui disc protejat la scriere (orificiu de protectie deschis).	COPY, MOVE ERASE, FORMAT,

		SAVE
	Drive B: is not prezent	FORMAT
	S-a incercat FORMAT pe discul :B, care nu e cuplat fizic.	
	Drive in use	Improbabil
	S-a incercat remaparea pe o unitate de disc care are fisiere deschise.	
	Drive not found	CAT, COPY
	Comanda contine o litera de unitate de disc ce nu este conectata.	ERASE, LOAD, MOVE, SAVE MERGE
RIC	**Drive not ready**	CAT, COPY,
	Unitatea de disc nepregatita sa execute comanda (sau nu e discheta in unitate). Se tasteaza R.	ERASE, FORMAT, LOAD, MERGE, MOVE, SAVE
	End of file found	Improbabil
	S-a incercat citirea unui octet dupa marcatorul END-OF-FILE.	
	File already exists	MOVE...TO
	Exista pe disc un fisier cu numele identic cu al fisierului destinatie.	
	File already in use	Improbabil
	Un program in cod a deschis deja fisiere pe care comenzi BASIC incearca sa le deschida.	
	File is read only	COPY,
	Fisierul este protejat la scriere. Protectia poate fi inlaturata folosind atributul "-P".	ERASE, MOVE, SAVE
	File not found	COPY, LOAD
	Fisierul cu numele dat in comanda de citire nu exista pe disc.	ERASE, MERGE, MOVE
	File not open	Improbabil
	S-a incercat lucrul cu un fisier care nu a fost deschis (OPEN).	
	File too big	Improbabil
	S-a incercat scrierea unui fisier >8Mo.	
	Invalid atribute	MOVE...TO
	Comanda MOVE are alt caracter atribut dupa +/- decit P,S sau A.	
	Invalid drive	FORMAT
	Comanda FORMAT are alta litera decit :A sau :B.	
	Missing address mark	CAT, COPY,
	A fost citit un sector care nu contine informatie uzuala folosita de sistem pentru a identifica unde se afla pe disc. De obicei acesta inseamna disc neformatat, sau disc compromis sau disc cu protectie incorporata.	ERASE, LOAD MERGE, MOVE, SAVE
	Missing extent	Improbabil
	Fisierele sint constituite din unitati de cite 16K. Eroarea va rezulta cind, dupa citirea informatiilor despre fisier si inainte de a citi, o unitate de 16K constitutiva a fisierului, se schimba discul.	
RIC	**NO data**	COPY,
	Nu poate fi gasit un identificator de sector	ERASE,
	Eroare posibila cind se incearca copierea unui disc ce are protectie incorporata.	LOAD, MOVE MERGE

	No rename between drives	MOVE...TO
	Fisierele sursa si destinatie din instructiu-	
	nea MOVE nu sint pe aceeasi unitate de disc.	
RIC	**Seek fail**	CAT, COPY,
	Unitatea de disc nu poate localiza pista	ERASE,
	ceruta pe disc (posibila recalibrare).	FORMAT,
		MOVE, SAVE
		LOAD
	Unchanged	Improbabil
	Eroare interna de sistem.	
RIC	**Unknown disk error**	Improbabil
	Eroare neclasificata aparuta in sistem.	
RIC	**Unrecognised disk format**	CAT, COPY
	Sistemul nu recunoaste formatul discului	ERASE,
	scris/citit (a citit specificatiile de disc	MERGE,
	dar a gasit informatie fara sens). Discul	MOVE, SAVE
	poate prezenta protectie incorporata.	LOAD
RIC	**Unsuitable media**	CAT, COPY,
	Discul din unitate are un format nepotrivit.	ERASE,
	(Exemplu: numar nepotrivit de piste intre	LOAD,
	unitate si formatarea anterioara a discului).	MERGE,
		MOVE, SAVE
		FORMAT

4.31 Breviar de BASIC+3

Din sumar:

Variabile
Siruri
Functii
Sumar de cuvinte cheie
Operatii matematice

Numerele sint memorate cu o precizie de 9 sau 10 cifre. Cel mai mare numar ce il puteti obtine este aproximativ 10^38 si cel mai mic numar (pozitiv) este aproximativ 4*10^-39.

Daca un numar nu este o putere a lui doi, exista posibilitatea sa apara erori la operatii matematice repetate. Este preferat lucrul cu numere intregi, daca este necesara precizie mare.

Un numar este memorat in calculator in virgula mobila (flotanta) pe un octet pentru exponent (1<=e<=255) si patru octeti pentru mantisa (1/2<=m<1). Aceasta reprezinta numarul m*2^(e-128).

Deoarece 1/2<=m<1, cel mai semnificativ bit al mantisei este intotdeauna 1. Deci poate fi inlocuit cu bitul de semn al mantisei, 0 pentru numere pozitive si 1 pentru numere negative.

Numerele intregi mici au o reprezentare speciala in care primul octet este #00(0), al doilea este octetul de semn (#00 sau #FF) si al treilea si al patrulea reprezinta intregul (in complement fata de doi).

Variabilele numerice au nume de lungimi diferite, incepind cu o litera si continuind cu litere sau cifre. Spatiile sint ignorate si toate literele sint convertite intern in litere mici.

Variabilele de control pentru buclele FOR...NEXT au numele formate dintr-o singura litera.

Tablourile numerice au de asemenea numele dintr-o singura litera, care poate fi aceeasi cu a unei variabile simple. Valoarea de start a indicelui este 1.

Sirurile sint flexibile ca lungime. Numele unui sir consta dintr-o litera, urmata de semnul $.

Tablourile de siruri au numele format dintr-o litera urmata de semnul $, dar nu poate fi acelasi cu numele unei variabile sir simple. Toate sirurile dintr-un tablou au aceeasi lungime fixa, care este specificata in instructiunea DIM. Valoarea de start a indicelui este 1.

Subsirurile pot fi atribuite cu instructiunea LET.

4.31.1 Functii

Argumentul unei functii nu necesita paranteze daca este o constanta sau o variabila.

FUNCTIA	TIPUL ARGUMENTULUI	REZULTAT
ABS x	numar	Valoare absoluta
ACS x	numar	Arccosinus in radiani. Eroare A daca x nu este in intervalul -1, +1.

a AND b	operatie binara; operandul drept intotdeauna un numar; operandul sting: numeric	Operatia "si". Are prioritate 3. a AND b= ¦ a daca b<>0 ¦ 0 daca b=0
	sir	a$ AND b= ¦ a$ daca b<>0 ¦ "" daca b=0
ASN x	numar	Arcsinus in radiani. Eroare A daca x nu este in intervalul -1, +1.
ATN x	numar	Arctangenta in radiani.
ATTR (x,y)	doua argumente; ambele numere, intre paranteze	Un numar care in binar codeaza atributele caracterului de la linia x si coloana y de pe ecran. Bitul 7 (cel mai semnificativ) marcheaza clipirea; bitul 6 marcheaza stralucirea; bitii 5...3 sint culoarea hirtiei; bitii 2...0 sint culoarea cernelii. Eroare B daca x nu este in intervalul 0...23 si y in intervalul 0...31.
BIN x	numar binar	Nu este de fapt o functie ci o notatie pentru numere. BIN urmata de o succesiune de 0 si 1 este numarul care are o asemenea reprezentare in binar.
CHR$ x	numar	Caracterul al carui cod este x rotunjit la cel mai apropiat intreg.
CODE x$	sir	Codul primului caracter din x$ sau 0 daca x$ este sirul nul (nici un caracter).
COS x	numar (in radiani)	Cosinusul lui x.
EXP x	numar	e^x.
FN		FN urmat de o litera apeleaza o functie definita de utilizator (vezi DEF). Argumentele trebuie puse intre paranteze (chiar daca nu sint argumente, parantezele trebuie sa fie prezente).
IN x	numar	Rezultatul citirii la nivel microprocesor de la portul x (0<=x<=#FFFF). Incarca regis-

		trul pereche BC cu x si executa instructiunea in a,(c) a limbajului de asamblare pentru Z80.
INKEY$	nu are	Citeste tastatura. Rezultatul este caracterul reprezentat de tasta apasata (daca a fost apasata una) sau sirul nul.
INT x	numar	Partea intreaga a lui x (intotdeauna rotunjita in jos)
LEN x$	sir	Lungimea sirului x$.
LN x	numar	Logaritmul natural (in baza e). Eroare A daca x<=0.
NOT x	numar	0 daca x<>0 si 1 daca x=0. Functia NOT are prioritate 4.
a OR b	operatie binara; doua argumente, numere	a OR b= { 1 daca b<>0 ; a daca b=0. OR are prioritatea 2.
PEEK x	numar	Valoarea octetului din memorie a carui adresa este x (rotunjit la cel mai apropiat intreg). Eroare B daca valoarea lui x nu apartine intervalului (0...65535)
PI	nu are	Numarul PI (3.1415927...).
POINT (x,y)	doua argumente, numere puse intre paranteze	1 daca punctul de cordonate x si y este de culoarea cernelii si 0 daca punctul este de culoarea hirtiei. Eroare B daca x nu este in intervalul (0...255) si y nu este in intervalul (0...175).
RND	nu are	Da urmatorul numar pseudo-aleator dintr-o secventa generata luind puterile lui 75 modulo 65537, scazind 1 si impartind cu 65536. Rezulta un numar cuprins in intervalul 0<=x<1.
SCREEN$ (x,y)	doua argumente, numere puse intre paranteze	Caracterul care este (normal sau inversat) pe ecran la linia x si coloana y. Intoarce sirul nul daca caracterul nu este recunoscut. Eroare B daca x nu este cuprins in intervalul (0...23) si daca y nu este cuprins in

		intervalul (0...31).
SGN x	numar	Semnul lui x. Intoarce -1 daca x este negativ, 0 daca x este 0 si 1 daca x este pozitiv.
SIN x	numar (in radiani)	Sinusul lui x.
SQR x	numar	Radacina patrata (radical) a lui x. Eroare A daca x<0.
STR$x	numar	Sirul de caractere care ar fi afisate daca x s-ar afisa pe ecran.
TAN x	numar	Tangenta lui x.
USR x	numar	Apeleaza subrutina in cod masina a carei adresa de inceput este x. La inceputul executiei rutinei memoria este configurata astfel: #0000...#3FFF (0...16383) este ocupata de ROM3 (48 BASIC) #4000...#7FFF (16384...32767) este ocupata de RAM pagina 5 #8000...#BFFF (32768...49151) este ocupata de RAM pagina 2 #C000...#FFFF (49152...65535) este ocupata de RAM pagina 0. Daca sint apelate rutine din +3DOS RAM pagina 7 este pusa in zona #C000...#FFFF (49152 ...65535) si ROM2 (+3DOS) va fi pusa in zona #0000... #3FFF (0...16383). Vezi sectiunea 4.9.2 pentru detalii. Intoarce continutul registrului pereche BC.
USR x$	sir	Adresa caracterului grafic definit de utilizator corespunzator lui x$. Eroare A daca x nu este o singura litera intre a si u sau un caracter grafic definit de utilizator.
VAL x$	sir	Evalueaza x (fara ghilimele) la o expresie numerica. Eroare C daca x contine eroare de sintaxa sau da un sir. Alte erori posibile depind de expresie.
VAL$ x$	sir	Evalueaza x (fara ghilimele) la o expresie sir. Eroare C daca x are valoare numerica. Alte erori posibile la fel ca

-x	numar	Negare.

Urmatoarele sint operatii binare:

+	adunare (pt. numere) sau concatenare (pt. siruri)	
-	scadere	
*	inmultire	
/	impartire	
^	ridicare la putere (Eroare B daca operandul sting este negativ)	
=	egal	ambii operanzi trebuie sa fie de acelasi tip. Rezultatul este un numar: 1 daca comparatia este adevarata; 0 daca nu.
>	mai mare	
<	mai mic	
<=	mai mic sau egal	
>=	mai mare sau egal	
<>	diferit (nu e egal)	

Functiile si operatiile au urmatoarele prioritati:

Operatia	Prioritate
Despartire in subsiruri	12
Toate functiile cu exceptia lui	
NOT si minus (negatie)	11
^ (ridicare la putere)	10
- Minus (negare)	9
*,/ (inmultire, impartire)	8
+,- (adunare, scadere)	6
=,>,<,<=,>=,<> (operatori relationali)	5
NOT	4
AND	3
OR	2

4.31.2 Instructiuni

In continuare vom folosi urmatoarele notatii:

l	- reprezinta o singura litera
v	- reprezinta o variabila
x,y,z	- reprezinta o expresie numerica
m,n	- reprezinta o expresie numerica rotunjita la cel mai apropiat intreg
e	- reprezinta o expresie
f	- reprezinta o expresie evaluata la sir
d	- reprezinta un sir care este evaluata la un driver valid: A:, B:, M: sau T:
u	- reprezinta un nume de fisier DOS neambiguu
a	- reprezinta un nume de fisier, care poate fi ambiguu (de exemplu poate sa contina * sau ?)
s	- reprezinta o secventa de instructiuni separate prin virgula
c	- reprezinta o secventa de atribute de culoare, fiecare terminat cu virgula sau punct si virgula. Un articol are forma: PAPER, INK, FLASH, BRIGHT, INVERSE sau OVER.

Expresiile optionale sint puse in paranteze drepte. Toate instructiunile cu exceptia INPUT, DEF FN si DATA pot fi utilizate ca si comenzi sau in programe. O comanda sau o linie de program

poate avea mai multe insiructiuni separate prin semnul doua puncte. Nu exista restrictii asupra locului in care o instructiune poate sa apara intr-o linie; (cititi despre IF si REM).

BEEP x,y

Produce o nota muzicala, in difuzorul calculatorului pentru x secunde, de inaltimea y semitonuri deasupra notei DO de jos (C mediu pentru notatia americana). Daca y este negativ atunci va fi sub nota DO.

BORDER m

Stabileste culoarea marginii din jurul ecranului si culoarea hirtiei pentru partea de jos a ecranului. Eroare K daca m nu e cuprins intre 0 si 7.

BRIGHT m

Stabileste stralucirea caracterelor ce urmeaza a fi afisate; 0 pentru normal, 1 pentru stralucitor si 8 pentru transparent.

CAT [#n,] [d] [a]

Comanda CAT scoate un catalog in ordine alfanumerica a fisierelor de pe un disc. Daca se utilizeaza forma CAT #n, ... catalogul (iesirea) este trimis pe canalul n. Daca este inclus un nume de fisier anumit (sau nume cu marcatorii * sau ?), atunci numai acele fisiere, care "se potrivesc", vor fi afisate. Cind CAT este urmat numai de o litera, marcatoare a unitatii de disc, atunci numai fisierele de pe discul din drive-ul respectiv vor fi afisate. Daca litera specificata este T se va afisa un catalog cu numele fisierelor de pe banda (impreuna cu informatii utile pentru transferul banda-disc).

CAT [#n,] [d] [a] EXP

La fel ca si comanda CAT, dar da un catalog extins, care include si fisierele protejate la scriere cit si fisierele arhiva si fisierele de sistem (vezi MOVE u TO f).

CIRCLE x,y,z

Deseneaza un cerc cu centrul in (x,y) si de raza z.

CLEAR

Sterge toate variabilele, lasind liber spatiul pe care acestea l-au ocupat anterior. Executa de asemenea si instructiunile RESTORE si CLS si pune pozitia pentru PLOT in coltul din stinga jos si curata stiva GOSUB.

CLEAR n

Ca si CLEAR, dar daca este posibil schimba variabila de sistem RAMTOP la n si muta stiva GOSUB corespunzator. Aceasta comanda poate fi utilizata pentru a ne asigura ca stiva masina este sub #BFE0 (49120) cind se introduce o

rutina care apeleaza +3DOS din BASIC.

CLOSE #n

Marcheaza fluxul (sirul) de date n ca nefiind atasat la nici un canal. Poate fi deci folosit intr-o instructiune ulterioara lui OPEN# n, f.

CLS

(Sterge ecranul). Sterge fisierul ecran.

CONTINUE

Continua executia unui program din punctul in care a fost oprit cu mesaj (in afara de mesaj 0). Daca mesajul a avut nr. 9 sau L, atunci executia continua cu urmatoarea instructiune (se iau si salturile in considerare), altfel se repeta instructiunea la care a aparut mesajul. Daca ultimul mesaj a fost generat de o linie de comanda, atunci CONTINUE asteapta sa continue comanda'si va intra in bucla daca eroarea a fost 0:1; va genera mesaj 0 daca mesajul a fost 0:2; va genera mesaj N daca a fost 0:3 sau mai mare.

COPY

Trimite o copie a celor 22 linii de ecran catre imprimanta (daca este conectata). Eroare D daca s-a apasat tasta BREAK.

COPY EXP [INVERSE]

Trimite o copie a 24 linii din ecran catre imprimanta. Fiecare punct colorat de pe ecran este tiparit astfel incit apar diverse nuante de gri pentru fiecare culoare. Este luat in considerare si atributul BRIGHT. INVERSE, daca este prezent, produce inversarea copiei (ca un negativ).
Eroare D daca a fost apasata tasta BREAK.

COPY u1 TO u2
COPY a TO d
COPY d TO d

Copiaza fisierul cu primul nume in fisierul cu al doilea nume. Cele doua nume trebuie sa fie diferite. Literele pentru unitatea de disc si numar utilizator (user number) pot fi specificate in numele de fisier.
Daca sursa (u1) are marcatori sau este un anumit nume, atunci destinatia (u2) trebuie sa fie doar o litera ce desemneaza unitatea de disc. (In acest caz fisierul destinatie va avea acelasi nume cu al fisierului sursa). Daca sursa si destinatia sint doar litere care specifica unitatea de disc, se va face un transfer complet disc-disc (toate fisierele existente anterior pe discul destinatie vor fi sterse). Daca discul destinatie nu este formatat pentru +3 atunci transferul disc-disc nu se va efectua.

La copiere, daca exista deja un fisier cu numele fisierului destinatie, atunci va apare mesajul "File already exists" (Fisier deja existent), daca apare mesajul "Missing address mark" atunci probabil ca discul nu este formatat.

COPY u TO SCREEN $

Afiseaza continutul unui fisier de pe disc pe ecran. Caracterele de control vor fi inlocuite cu spatii. Poate fi utilizata la vizualizarea fisierelor ASCII.

COPY u TO LPRINT

Continutul fisierului de pe disc cu numele u este trimis la imprimanta. Daca s-a folosit inainte comanda FORMAT LPRINT "R", atunci aceasta forma a instructiunii COPY poate fi utilizata ca o trecere a programelor pe alta masina.

COPY u TO SPECTRUM FORMAT

Face ca header-ul de fisiere sa fie adaugat la un fisier binar creat pe un alt tip de masina. Se creaza un nou fisier cu numele u.HED.

DATA e1, e2, e3,...

Parte din lista DATA. Trebuie pusa in program (nu comanda) altfel nu are nici un efect.

DEF FN l(l1,...,lk)=e

Defineste functii definite de utilizator. Trebuie sa fie pusa in program altfel nu are nici un efect. l, l1,...lk pot fi sau o singura litera sau o litera urmata de semnul $ pentru argument sau rezultat sir.
Daca nu sint argumente, are forma:

```
DEF FN l()=e
```

DIM l(n1,...nk)

Sterge orice tablou existent cu numele l si creaza un nou tablou cu numele l de k dimensiuni, pe care il initializeaza cu zero.

DIM l$(n1,...,nk)

Sterge orice tablou de siruri sau sir existent cu numele l$ si creaza un tablou de siruri cu numele l$ de k dimensiuni si initializeaza toate valorile la "" (sirul nul). Poate fi considerat si ca un tablou de siruri de lungime fixa nk cu k-1 dimensiuni (n1,...nk-1). Un tablou este nedefinit, pina cind nu este declarat in instructiunea DIM. Eroare 4 daca nu este loc pentru tablou in memorie.

DRAW x,y

Este cazul particular DRAW x,y,0.

DRAW x,y,z

Deseneaza o linie din pozitia curenta, pe o lungime data de x puncte pe orizontala si y puncte pe verticala cu o curbura data de unghiul z (pentru z=0 va fi

desenata o linie dreapta).
Eroare 8 daca linia iese afara din ecran.

ERASE a
ERASE d

Daca este specificat un singur fisier, atunci acel fisier va fi sters de pe unitatea de disc implicita sau de pe cea specificata in numele fisierului. Daca numele fisierului are marcatori, va apare mesaj pentru confirmare. Daca se apasa y, toate fisierele care "se potrivesc" specificatiei vor fi sterse. Daca ERASE este urmata de o litera pentru unitatea de disc, toate fisierele de pe unitatea respectiva vor fi sterse fara a se mai cere confirmare.

FLASH n

Defineste daca caracterele vor clipi sau vor fi statice; n=0 pentru static, n=1 pentru clipitor, n=8 pentru nici o schimbare.

FOR l=x TO y

Este forma particulara FOR l=x TO y STEP 1.

FOR l=x TO y STEP z

Sterge orice alta variabila existenta cu numele l si creaza o variabila de control cu numele l, de valoare initiala x, finala y si pas z si adresa de intoarcere in bucla, adresa primei instructiuni dupa instructiunea FOR. Verifica daca valoarea initiala este mai mare (daca z>=0) sau mai mica (daca z<=0) decit valoarea finala si daca da, sare la instructiunea NEXT l, generind mesaj de eroare 1, daca aceasta nu exista. Vezi NEXT. Eroare 4, daca nu este loc pentru variabila de control.

FORMAT d

Pregateste discul din unitatea specificata (A: sau B:) pentru a fi utilizat. Daca discul a fost deja formatat pe un +3, se va genera un mesaj ce face ca operatia sa fie abandonata. Discurile formatate pe alte masini nu vor fi recunoscute.

FORMAT LINE n

Stabileste rata de transmisie la interfata RS232 la n. Ratele valide sint in intervalul (75...19200).

FORMAT LPRINT fi[;f2]

Daca sirul f1 este "C", tiparirile ce urmeaza se vor face prin interfata paralela (mufa PRINTER). Daca sirul f1 este "R" atunci tiparirile vor fi directionate catre mufa RS232. Sirul f1 poate fi si "E" (pentru extins) in care caz caracterele pina la CHR$ 32 nu sint trimise la imprimanta si cele dupa CHR$ 127 sint convertite la cuvintele cheie

din. BASIC (token-uri). Cind sirul f1 este "U" (neextins) toate caracterele care urmeaza a fi imprimate vor fi trimise fara a fi translatate. In acest caz se transmit si secventele ESC (escape). Daca f1 este "C" sau "R" un al doilea sir, f2, poate fi specificat si acesta poate fi "E" sau "U".

GOSUB n

s

Pune in stiva adresa urmatoarei instructiuni dupa GOSUB, dupa care executa operatia similara instructiei GO TO n. In continuare poate sa apara eroare 4 daca nu sint destule instructiuni RETURN.

GO TO n

Sare la linia n (sau daca linia n nu exista, la prima linie dupa ea).

IF x THEN s

Daca x este adevarat (diferit de zero), atunci se executa s. s comprima toate instructiunile pina la sfirsitul liniei. Forma IF x THEN numar linie nu este permisa.

INK n

Stabileste culoarea cernelii cu care se vor afisa caracterele; n este in domeniul (0...7) pentru culoare, 8 pentru transparent si 9 pentru contrast. Eroare K daca n nu este in domeniul (0...9).

INPUT [#n,]...

#n este o succesiune de articole (expresii) in INPUT, separate (ca la PRINT) de virgula, punct si virgula sau apostrof.
Un articol din INPUT poate fi:
1. orice articol din PRINT care nu incepe cu o litera
2. un nume de variabila
3. LINE urmat de nume de variabila sir.
Articolele si separatorii pentru PRINT din 1, sint tratati la fel ca la PRINT, doar ca totul se afiseaza in partea de jos a ecranului. Pentru 2, calculatorul asteapta introducerea unei expresii de la tastatura si valoarea acesteia este atribuita variabilei. Pentru expresiile tip sir, buffer-ul de intrare este initializat la doua ghilimele (care pot fi sterse daca e necesar). Daca primul caracter introdus este STOP, programul se opreste cu eroare H. 3 este ca si 2, doar ca introducerea este tratata ca un sir fara ghilimele si mecanismul de STOP nu va functiona; pentru oprire trebuie tastat cursor jos.

INVERSE n

Controleaza inversarea caracterelor ce vor fi afisate. Daca n=0 caracterele se vor afisa normal (culoarea cernelii pe culoarea hirtiei). Daca n=1, caracterele

se vor afisa in video invers (culoarea hirtiei pe culoarea cernelii).
Eroare K daca n nu este 0 sau 1.
Nota: pentru BASIC 48K, tastind INV VIDEO este echivalent cu INVERSE 1 si tastind TRUE VIDEO este echivalent cu INVERSE 0.

LET v=e

Atribuie valoarea lui e variabilei v. Cuvintul BASIC LET nu poate fi omis. O variabila simpla nu este definita pina cind nu este atribuita printr-o instructiune LET, READ sau INPUT. Daca v este o variabila sir indiciata sau un subsir, atribuirea este de tip Procust (de lungime fixa) astfel: valoarea lui e este sau trunchiata sau completata cu spatii la dreapta, pentru a avea aceeasi lungime cu cea specificata pentru v.

LIST [#m]

Este forma particulara LIST [#m,] 0.

LIST [#m,]n

Listeaza programul (afiseaza) in partea de sus a ecranului, incepind cu prima linie a carui numar este cel putin n si pune linia curenta, linia cu numarul n. Daca este prezent #m, iesirea este trimisa pe canalul asignat m.

LLIST

Forma particulara LLIST 0.

LLIST n

La fel ca LIST, dar se utilizeaza imprimanta. Implicit, iesirea se face la mufa imprimantei paralele (PRINTER); iesirea poate fi directionata utilizind comanda FORMAT LPRINT "R" catre mufa RS232.
Ca listing-ul BASIC sa apara corect, codurile token-urilor (cuvinte cheie BASIC) sint extinse la literele ce compun fiecare token (codurile pina la 32 nu sint tiparite). Comanda FORMAT LPRINT "E" poate fi utilizata pentru a reveni la aceasta stare, daca ea a fost schimbata (cu FORMAT LPRINT "U").

LOAD d

Face ca toate intrarile (citirile) pentru urmatoarele operatii cu discul (COPY, ERASE, MOVE, etc.) sa fie facute de pe unitatea cu numele specificat. Daca litera specificata este T: atunci toate incarcarile urmatoare se vor face de pe banda.

LOAD f

Incarca programul si variabilele de pe disc (sau banda). Sirul f, care specifica fisierul ce trebuie incarcat poate include (optional) si o litera care sa specifice unitatea si un numar utilizator cind se lucreaza cu discul. Daca nu este specificata o litera pentru unitate, atunci se utilizeaza unitatea

implicita.
Daca sirul contine doar un asterisc (LOAD "*") se va incarca sectorul incarcator de pe discul din unitatea A (boot strap). Aceasta poate fi folosita pentru a incarca diverse sisteme de operare sau jocuri de pe disc.

LOAD f DATA 1()

Incarca un tablou numeric 1() din fisierul f.

LOAD f DATA 1$()

Incarca un sir de caractere 1$() din fisierul f.

LOAD f CODE m,n

Incarca n octeti, incepind cu adresa m.

LOAD f CODE m

Incarca octeti la adresa m. Daca un fisier de pe alt calculator a fost trecut in formatul Spectrum (utilizind comanda COPY u TO SPECTRUM FORMAT) atunci se utilizeaza aceasta forma de LOAD pentru a-l incarca.

LOAD f CODE

Incarca octeti la adresa de la care au fost salvati.

LOAD f SCREEN $

Forma particulara **LOAD f CODE 16384, 6912.** Incarca fisierul f in fisierul ecran.

LPRINT

Ca si PRINT, dar utilizeaza imprimanta. Utilizati comanda FORMAT LPRINT pentru a directiona iesirea la mufa PRINTER sau RS232. Implicit, iesirea este la mufa imprimantei paralele (PRINTER) cu tokenurile extinse si codurile sub 32 nu se imprima. Daca doriti sa se imprime si secventele ESC (escape) utilizati comanda FORMAT LPRINT "U" inainte de a utiliza LPRINT. Daca printerul a fost directionat catre RS232 (utilizind comanda FORMAT LPRINT "R") atunci LPRINT poate fi utilizat pentru a transmite siruri de caractere la un terminal.

MERGE f

Ca si LOAD f, dar sterge liniile de program si variabilele existente in memorie doar ca sa se elibereze loc in memorie pentru noile linii care au acelasi numar cu cele vechi sau variabile cu acelasi nume. Ca la LOAD, numele de fisier poate contine o litera marcatoare de unitate si un numar utilizator (user number). Daca nu este specificata litera pentru unitate va fi luata in considerare unitatea implicita.

MOVE f1 TO f2

Aceasta va redenumi fisierul f1 cu numele f2. Ambele fisiere trebuie sa fie pe acelasi drive.

MOVE u TO f

Sirul f poate fi "+P","+S","+A","-P","-S" sau "-A". Aceasta face ca atributele fisierului specificat prin u sa fie valabile (+) sau nu (-). Literele atribute din sirul f controleaza protectia la scriere (P), starea sistemului (S) sau starea arhivei (A). Comanda CAT...EXP poate fi utilizata pentru a afisa atributele curente. Fisierele protejate nu pot fi sterse, salvate peste vechea varianta sau sa fie supuse oricarei operatii ce le-ar schimba atributele. Fisierele de sistem nu pot fi vizualizate cu catalogarea normala ci numai cu comanda CAT...EXP. Starea arh furnizeaza compatibilitatea cu CP/M si nu are alta legatura cu +3.

NEW

Restarteaza sistemul BASIC, stergind orice program si variabile, utilizind memoria pina la inclusiv octetul a carui adresa este in variabila de sistem RAMTOP. Variabilele de sistem UDG, PRAMT, RASP si PIP sint rezervate. Da controlul meniului principal, dar nu sterge fisierele existente pe drive-ul M (RAM disc-ul).

NEXT l

1. Cauta variabila de control l.
2. Ii aduna la valoare, valoarea pasului.
3. Daca pasul este >=0 si valoarea variabilei de control este > decit valoarea finala, sau pentru pas <0 si valoarea variabilei de control este < decit valoarea finala, atunci sare la instructiunea de reluare a buclei.

Eroare 2 daca nu exista variabila l.
Eroare 1 daca variabila l nu se potriveste cu variabila din instructiunea FOR.

OPEN #n,f

Permite ca sirul (fluxul) de date n sa fie atasat canalului identificat prin sirul f. Numerele pentru fluxuri pot fi in intervalul (0...15), dintre care sistemul le utilizeaza pe (0...3). Siruri posibile sint "S" pentru ecran, "K" pentru tastatura si "P" pentru imprimanta. Canalul de imprimanta poate fi mai departe redirectionat catre mufa imprimantei paralele (PRINTER) sau RS232 utilizind instructiunea FORMAT LPRINT.
Citirea de la un flux atasat unui canal care permite doar iesirea sau invers, va genera mesajul de eroare "Invalid I/O device".

OUT m,n

Trimite octeti la portul m la nivel procesor. (Incarca registrul pereche BC cu m, registrul A cu n si executa in-

structiunea **out (c),a** a limbajului de asamblare pentru Z80. Eroare B daca m nu este cuprins intre 0 si 65535 si daca n nu este in intervalul (-255...255).

OVER n

Controleaza supraimprimarea caracterelor ce urmeaza a fi afisate . Daca n=0, vechile caractere din pozitia adresata vor fi sterse, inainte de a scrie noul caracter. Daca n=1, noile caractere se vor afisa suprapunindu-se peste cele vechi.
Eroare K daca n nu este 0 sau 1.

PAPER m

Ca si INK, dar controleaza hirtia (culoarea fondului).

PAUSE n

Opreste executia si afiseaza fisierul ecran timp de n cadre TV (sint 50 cadre pe secunda) sau pina se apasa o tasta. Daca n=0, atunci pauza nu este temporizata, ci tine pina cind se apasa o tasta.
Eroare B daca n nu este in intervalul (0...65535).

PLAY f1[,f2,...,f8]

Poate genera sunete pe maxim 8 canale simultan. Primele trei canale se pot amplifica de la cupla audio-casetofon. Cuplat corespunzator cu un sintetizator de muzica (MIDI) poate genera sunete simultan pe toate cele 8 canale.

PLOT c;m,n

Afiseaza un punct de cerneala in punctul de coordonate (m,n) mutind aici pozitia pentru PLOT. Daca articolul de culoare c, nu specifica altfel, culoarea cernelii din punctul de coordonate (m,n) este schimbat la culoarea permanenta a cernelii si celelalte atribute (culoarea hirtiei, clipirea si stralucirea) sint lasate neschimbate.
Eroare B, daca m nu este cuprins intre 0 si 255 si daca n nu este cuprins intre 0 si 175.

POKE m,n

Pune valoarea n in octetul de adresa m.
Eroare B daca m nu este cuprins intre 0 si 65535 si daca n nu este in intervalul (-255,255).

PRINT [#n,]

#n este o secventa de articole (expresii) PRINT separate de virgula, punct si virgula sau apostrof. Cind se utilizeaza forma PRINT #n, iesirea este directionata spre sirul de date n si nu spre ecran. Punctul si virgula dintre articole nu are efect; este utilizat pentru delimitarea articolelor. O virgula muta afisarea la urmatoarea zona, iar apostroful genereaza "carriage return" si

"line feed" (ceea ce se genereaza implicit daca o instructiune PRINT nu se termina cu punct si virgula, virgula sau apostrof).
Un articol din PRINT poate fi:
1. Nimic (gol)
2. O expresie numerica. Prima data se afiseaza semnul minus daca valoarea ei este negativa. Fie x modulul valorii. Daca x<=10^-5 sau >=10^13, atunci afisarea se face utilizind notatia stiintifica. Mantisa are pina la 8 cifre (fara zerouri in completare) si punctul zecimal (absent daca este o singura cifra) este dupa prima cifra. Exponentul este format din litera E , urmata de + sau - si de una sau doua cifre. Altfel x este afisat in notatie zecimala cu pina la opt cifre semnificative si fara zero-uri dupa punct. Daca punctul zecimal este chiar la inceput, este urmat intotdeauna de un zero. De exemplu .03 si 0.3 sint astfel afisate. Zero este tiparit ca o singura cifra.
3. O expresie sir. Token-urile (cuvintele cheie) din sir sint extinse, posibil cu un spatiu inainte sau dupa. Caracterele de control au efectul lor specific de control. Caracterele nerecunoscute vor fi tiparite ca ? (semnul intrebarii).
4. AT m,n. Trimite caracterul de control AT urmat de un octet pentru m (numarul liniei) si un octet pentru n (numarul coloanei).
5. TAB n. Trimite caracterul de control TAB urmat de doi octeti pentru n (cel mai putin semnificativ octet primul).
6. Un articol de culoare care are forma PAPER, INK, FLASH, BRIGHT, INVERSE sau OVER.

RANDOMIZE

Forma particulara RANDOMIZE 0.

RANDOMIZE n

Initializeaza variabila de sistem (numita SEED) care se utilizeaza pentru a genera urmatoarea valoare pentru RND. Daca n<>0 atunci SEED are valoarea n. Daca n=0, atunci SEED are valoarea altei variabile de sistem, numita FRAMES, care numara cadrele afisate pe ecran.
Eroare B daca n nu este cuprins in intervalul (0...65535).

READ v1, v2,...,vk

Atribuie variabilelor v1,...vk, expresii succesive din lista DATA. Eroare C daca o expresie este de tip gresit. Eroare E daca mai ramin variabile de citit, cind lista DATA este epuizata.

REM ...

Nu are nici un efect. Cimpul de informa-

tie ... poate fi orice secventa de caractere terminata cu ENTER. Nici o instructiune din linie nu va actiona daca este pusa dupa REM si virgulele nu vor fi tratate ca separatori.

RESTORE

Forma particulara RESTORE 0.

RESTORE n

Stabileste pointer-ul din lista DATA la prima instructiune DATA in linia n. Daca linia n nu exista (sau nu este o instructiune DATA), atunci este luata in considerare prima instructiune DATA dupa linia n si urmatoarea instructiune READ va incepe citirea de aici.

RETURN

Ia adresa din stiva GOSUB si sare la linia data de aceasta adresa. Eroare 7, daca nu exista adresa in stiva GOSUB.

RUN

Forma particulara RUN 0.

RUN n

Executa CLEAR si GO TO n.

SAVE d

Face unitatea de disc specificata sa fie unitatea implicita pentru toate operatiile cu discul, care urmeaza (COPY, ERASE, MOVE, etc.). Daca litera marcatoare de unitate este T: atunci toate salvarile ce urmeaza vor fi implicit pe banda.

SAVE f

Salveza programul si variabilele pe disc (sau banda), dind inregistrarii numele f. Numele de fisier poate include optional o litera pentru unitatea de disc si un numar utilizator cind se opereaza cu discul. Daca litera pentru unitatea de disc nu e specificata, atunci se utilizeaza unitatea implicita. Eroare F daca f este gol sau mai mare de 10 caractere (pe banda).

SAVE f LINE m

Salveaza programul si variabilele, astfel ca la incarcare, se face automat un salt la linia m.

SAVE f DATA 1()

Salveaza tabloul numeric 1() in fisierul f.

SAVE f DATA 1$()

Salveaza tabloul de siruri 1$() in fisierul f.

SAVE f CODE m,n

Salveaza n octeti incepind cu adresa m.

SAVE f SCREEN$

Forma particulara SAVE f CODE 16384,6912. Salveaza fisierul ecran curent.

SPECTRUM

Trece din +3 BASIC in 48K BASIC, mentinind orice program existent in RAM.

Trecerea inversa nu este posibila. Schimbarea ROM/RAM nu este invalidata cind se intra in 48 BASIC cu aceasta comanda (nu este si cazul cind se selecteaza optiunea 48 BASIC din meniul general).

STOP

Opreste executia programului cu mesaj 9. Comanda CONTINUE va relua programul de la instructiunea urmatoare.

VERIFY f

Ca si LOAD (de pe banda), dar informatia de pe banda nu este incarcata in RAM ci este comparata cu ceea ce exista deja in RAM. Daca numele de fisier specifica un fisier de pe disc (sau daca unitatea implicita este A: sau B:) atunci nu se intreprinde nici o actiune.
Eroare R daca rezulta erori din compararea octetilor.

4.32 Binar si hexazecimal

Din sumar:
Sisteme de numeratie
Biti si octeti

In acest subcapitol este descris modul in care "numara" calculatorul, utilizind sistemul binar.
In sistemul zecimal de numarare folosim cifrele 0,1,2,3,4,5,6,7,8,9, iar numerele sint grupate cite 10:

20,21,22,23,....,29
30,31,32,33,....,39
40,41,42,43,....,49 s.a.m.d.

In locul sistemului zecimal (bazat pe 10 cifre) la calculatoare se utilizeaza sistemul hexazecimal (bazat pe 16 cifre si litere). Aceste cifre si litere sint: 0,1,2,3,4,5,6,7,8,9,A,B,C, D,E,F. Pentru comparare intre cele doua sisteme de numeratie cititi tabelul de mai jos:

Zecimal	Hexa
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

16	10
17	11
.	.
.	.
.	.
25	19
26	1A
27	1B
.	.
.	.
.	.
31	1F
32	20
33	21
.	.
.	.
.	.
158	9E
159	9F
160	A0
161	A1
.	.
.	.
.	.
255	FF
256	100 s.a.m.d.

De obicei cind se scriu numere in sistemul hexazecimal, la sfirsitul numarului se adauga litera H (de la hexa). De exemplu pentru 158 (zecimal) se scrie 9EH si se citeste "noua E hexa".

O alta modalitate - des folosita in cadrul acestui manual - de a marca un numar scris in sistemul hexazecimal este insotirea acestuia de catre caracterul #, care precede numarul. Echivalentul acestui tip de notatie pentru exemplul de mai sus este: #9E.

Calculatoarele de fapt utilizeaza in calcule sistemul de numeratie binar, bazat pe doua cifre, 0 si 1, reprezentate de nivel de tensiune scazut (pentru 0) si nivel de tensiune ridicat (pentru 1). O cifra (o pozitie din numar) se numeste bit.

Deci tabelul anterior, extins si pentru sistemul binar devine:

Zecimal	Hexa	Binar
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	10000

17	11	10001
.	.	.
.	.	.
.	.	.

Trecerea (conversia) din binar in hexa este al'naibii de usoara. Astfel, numarul binar se imparte in grupe de cite 4 biti (incepind din dreapta numarului) si se converteste fiecare grupa de 4 biti in numarul corespunzator din sistemul hexazecimal (utilizati tabelul de mai sus). In final puneti cifrele hexa una linga alta si obtineti numarul complet in hexazecimal. De exemplu sa convertim numarul binar 10110100 in hexazecimal. Cele doua grupe de cite 4 biti (de la dreapta spre stinga) sint 0100 si 1011; deci cifrele hexazecimale corespunzatoare sint 4 si B iar daca aceste cifre sint puse in ordine va rezulta numarul hexazecimal #B4.

Daca numarul binar este mai lung de 8 biti , se continua cu impartirea pe grupe de cite 4, pina cind se acopera toate cifrele (eventual se completeaza cu zerouri la stinga ultima grupa de 4 biti daca este incompleta). De exemplu: 111010111110000; cele 4 grupe vor fi 0000 ,1111, 1010 si 0011 cu cifrele corespunzatoare 0,F,A,3 si puse in ordine dau numarul #3AF0.

Pentru a converti un numar din hexazecimal in binar, se schimba fiecare cifra (litera) cu grupul de 4 cifre in binar corespunzator, pornind din nou de la dreapta. De exemplu sa convertim numarul #F3 in binar: trecem intii 3 in binar , care este 0011 (se completeza la stinga cu zerouri pina se ajung la 4 cifre) si apoi se converteste F, care este 1111; puse una linga alta in ordine , obtinem numarul binar 11110011.

Calculatorul utilizeaza in calcule sistemul binar, dar pentru oameni este mai usoara notatia in hexazecimal (si cu mai putine sanse de eroare la scriere, intrucit se lucreaza cu mai putine cifre). De exemplu, este mai usor sa scriem #3AF0 decit 0011101011110000.

In calculator, numerele binare sint grupate cite 8 si formeaza asa numitul "octet". Un octet reprezinta un numar cuprins intre 0 si 255 in zecimal (FF in hexazecimal sau 11111111 in binar).

Doi octeti pot fi grupati impreuna si atunci formeaza un "cuvint". Un cuvint are deci 16 cifre binare sau 4 cifre hexazecimale si reprezinta un numar intre 0 si 65535 in zecimal (1111 1111 1111 1111 in binar sau #FFFF in hexazecimal).

Un octet este format intotdeauna din 8 biti, dar lungimea unui cuvint difera de la calculator la calculator.

Instructiunea BIN , prezentata in sectiunea 4.14 a acestui manual , este un mijloc de introducere in calculator a numerelor binare. Astfel BIN 111 reprezinta 7 in zecimal, BIN 1111 1111 reprezinta 255 in zecimal, s.a.m.d.

Deci, in instuctiunea BIN se utilizeaza doar cifrele 0 si 1. Nu se utilizeaza nici semnele negativ sau pozitiv, ca de exemplu BIN - 11 pentru -3; se poate utiliza -BIN 11 in loc. De asemenea, numarul nu are voie sa fie mai mare de 65535 (deci nu poate avea mai mult de 16 cifre de 0 si 1). Daca completati numarul in binar cu zerouri la stinga, calculatorul le va ignora, astfel BIN 00000001 este BIN 1.

4.33 Exemple de programe

Programe:

Renumerotare

Ceas
Demolare
Tenis

Renumerotare

Acest scurt program este un ajutor pentru facilitatea de renumerotare din meniul de editare. Daca adaugati cu MERGE acest program la programul la care lucrati puteti introduce atit linia de start cit si pasul dintre linii.

Tastati RUN 9000 pentru a executa programul; introduceti linia de start intre 1 si 9999); introduceti pasul (intre 1 si 9999) apoi apasati tasta EDIT si selectati optiunea Renumber din meniul de editare.

```
9000 INPUT "Linie start", st
9010 INPUT "Pasul", sp
9020 LET hst=INT (st/256)
9030 LET hsp=INT (sp/256)
9040 POKE 23413,st-256*hst
9050 POKE 23414,hst
9060 POKE 23415,sp-256*hsp
9070 POKE 23416,hsp
9080 PRINT "Tastati EDIT si apoi selectati optiunea
     Renumber"
```

Ceas

Si-acum, un program ceva mai ... serios! Pentru a-l lansa in executie tastati RUN, introduceti ora (intre 1 si 12) si minutele (intre 0 si 59). Apoi ceasul va porni:

```
10 DIM s(60) : DIM c(60)
20 BORDER 0 : PAPER 0 : BRIGHT 1 : INK 7 : CLS
30 PRINT AT 10,1;"Asteptati putin"
40 PRINT
50 GOSUB 370
60 LET z$="00"
70 CLS
80 INPUT "Ce ora este?";h
90 INPUT "Cite minute?";m
100 LET s=0 : POKE 23672,0 : POKE 23673,0
110 IF h=12 THEN LET h=0
120 LET xc=112 : LET yc=90 : LET r=70 : LET rh=r/2 : LET
    rm=r*3/4 : LET rs=r*5/6
130 CIRCLE xc,yc,r
140 INK 1
150 FOR i=0 TO 359 STEP 30
160 PLOT (r+1)*s(i/6+1)+xc,(r+1)*c(i/6+1)+yc
170 NEXT i
180 INK 4
190 OVER 1 : GOSUB 500
200 GOSUB 470
210 GOSUB 440
220 LET tm=INT((PEEK 23672 + 256 * PEEK 23673)/50)
230 IF s+1=tm THEN LET os=s : LET s=s+1 : GO TO 250
240 GO TO 220
250 IF s=60 THEN LET s=0 : POKE 23672,0 : POKE 23673,0 :
    LET om=m : LET m=m+1 : GO TO 290
260 PLOT xc, yc  DRAW rs*s (os+1), rs*c (os+1)
270 GOSUB 440
```

```
280 GO TO 220
290 IF m=60 THEN LET m=0 : LET oh=h : LET h=h+1 : GO TO 330
300 PLOT xc, yc : DRAW rm*s(om+1), rm * c (om+1)
310 GOSUB 470
320 GO TO 260
330 IF h=12 THEN LET h=0
340 PLOT xc, yc : DRAW rh*s(oh*5+1),rh*c(oh*5+1)
350 GOSUB 500
560 GO TO 300
370 PRINT AT 14,0
380 FOR i=6 TO 360 STEP 6
390 PRINT ".";
400 LET s(i/6)=SIN((i-6)*PI/180)
410 LET c(i/6)=COS((i-6)*PI/180)
420 NEXT i
430 RETURN
440 PLOT xc, yc : DRAW rs*s(s+1),rs*c(s+1)
450 LET s$=STR$(s) : PRINT OVER 0; AT 18,27; INK 4;":";
    INK 6; z$(TO 2-LEN (s$)); s$
460 RETURN
470 PLOT xc,yc : DRAW rm*s(m+1), rm*c(m+1)
480 LET m$=STR$(m) : PRINT OVER 0; AT 18,24; INK 2;":";
    INK 5; z$(TO 2 - LEN (m$)); n$
490 RETURN
500 PLOT xc, yc : DRAW rh*s(h*5+1),rh*c(h*5+1)
510 LET ph=h : IF ph=0 THEN LET ph=12
520 LET h$=STR$(ph) :  PRINT OVER 0; INK 3; AT 18,22;"  "(TO
    2 - LEN (h$));h$
530 RETURN
```

Demolare

Acest program este un joc distractiv intre un jucator si calculator.

Pentru a juca, tastati RUN si apoi apasati orice tasta pentru start.

Instructiuni:

Cursor stinga - paleta la stinga;
Cursor dreapta - paleta la dreapta;
Spatiu - da o sansa (viata) pentru un nou ecran.

Nota Cind introduceti programul, retineti urmatoarele:

1. Sirul "BBBBB..." din liniile 30 si 50 sint caractere grafice. Se obtin tastind o data GRAPH (pentru a intra in modul grafic) si apoi tastind B pentru a obtine respectivele caractere grafice si la sfirsit din nou GRAPH pentru a iesi din modul grafic.

2. Sirul "3333" din linia 210 sint tot caractere grafice. Se obtin de asemenea intrind in modul grafic cu GRAPH si apoi tastind 3, de patru ori, apoi din nou GRAPH.

3. Sirul "A" din linia 430 este tot un caracter grafic. Tastati GRAPH, apoi A o data si din nou GRAPH.

```
10 BORDER 0 : INK 0 : PAPER 0 : CLS : BRIGHT 1
20 GOSUB 560
30 LET b$="BBBBBBBBBBBBBBBBBBBBBBBBBBBB" : REM 28 de B
40 LET s$="    ...." : REM 32 spatii
50 PRINT AT 3,12; INK 7; FLASH 1; "DEMOLARE"; FLASH 0;
   AT 6,9; INK 1; "B"; INK 7;" = 20 Puncte"; AT 8,9;
   INK 4;"B"; INK 7;" = 10 Puncte"; AT 10,9; INK 2;
   "B"; INK 7;" = 5 Puncte"
```

```
60 PRINT AT 14,1; INK 4; "Tastati SPACE pentru o viata"
70 PAUSE 200
80 LET hiscor=0
90 LET tscor=0
100 LET vieti=5
110 LET scor=0
120 CLS
130 INK 7 : PLOT 12,13: DRAW 0,160 : DRAW 230,0 :
    DRAW 0,-160 : INK 0
140 PRINT AT 1,2: INK 1; b$; AT 2,2; INK 4; b$
150 FOR r=5 TO 6 : PRINT AT r,2; INK 2; b$ : NEXT r
160 LET bx=9
170 PRINT AT 19,5; INK 6; "Orice tasta pentru start";
    AT 17,4; "Utilizati < si > pentru a misca paleta"
180 PAUSE 0
190 PRINT AT 19,5; INK 0; s$(TO 24); AT 20,0; s$ (To 32);
    AT 17,4; s$ (TO 24)
200 PRINT AT 21,0; INK 0; s$ (TO 32) : GOSUB 540 :
    GO TO 220
210 PRINT AT 20,bx; INK 0; " "; INK 5; "3333"; INK 0;
    " "; RETURN
220 LET xa=1 : LET ya=1 : IF INT(RND*2)=1 THEN LET xa=-xa
230 GOSUB 210
240 LET x=bx+4 : LET y=11 : LET xc=x : LET yc=y
250 REM bucla principala
260 IF scor>1100 THEN GO TO 110
270 IF  INKEY$=" " OR INKEY$="0" THEN IF vieti>1  THEN  LET
    vieti=vieti -1 : GO TO 110
280 LET xc=x+xa : LET yc=yc+ya
290 REM scanare tastatura
300 GOSUB 470
310 IF yc=20 THEN IF ATTR (yc,xc)=69 THEN PLAY "N1g" :
    LET ya=-ya : LET yc=yc-2 : IF xc=bx+1 OR xc=bx+4 THEN
    LET xa=xa : LET xc=x+xa
320 IF yc=21 THEN PLAY "03M7#d : PRINT AT y,x; " " :
    GO TO 450
330 GOSUB 470
340 IF yc=20 THEN GO TO 430
350 LET t=ATTR (yc,xc)
360 IF t=71 THEN GO TO 410
370 IF t=64 THEN GO TO 420
380 LET ya=-ya : LET xz=xc : LET yz=yc : LET yc=yc+ya :
    GOSUB 510 : IF t=66 THEN PLAY "N1e" : LET scor=scor+5 :
    LET tscor = tscor+5 : GOSUB 540 : GO TO 350
390 IF t=68 THEN PLAY "N1c" :P LET scor=scor+10 :
    LET tscor=tscor+10 : GOSUB 540 : GO TO 350
400 IF t=65 THEN PLAY "N1a" : LET scor=scor+20 :
    LET tscor=tscor+20 : GOSUB 540: GO TO 350
410 LET xa=-xa : LET xc=xc+2*xa : PLAY "N1f"
420 IF yc=1 THEN LET ya=1
430 PRINT AT y,x; INK 0; " "; AT yc,xc; INK 3; "A" :
    LET x=xc : LET y=yc
440 GO TO 250
450 LET vieti=vieti-1 : IF vieti=0 THEN GO TO 530
460 GOSUB 540 : GO TO 220
470 LET a$=INKEY$
480 IF (a$=CHR$(8) OR a$="6") AND bx>1 THEN LET bx=bx-1 :
    GOSUB 210 : RETURN
490 IF (a$=CHR$(9) OR a$="7") AND bx<25 THEN LET bx=bx+1 :
    GOSUB 210 : RETURN
500 RETURN
```

```
510 IF yz=20 THEN RETURN
520 PRINT AT yz,xz; INK 0; " " : RETURN
530 GOSUB 540 : PRINT AT 10,10; INK 7; "SFIRSIT"; AT 12,8;
    "Ati obtinut scorul:"; tscor : FOR i=1 TO 300 : NEXT i:
    GO TO 90
540 IF tscor>hiscor THEN LET hiscor=tscor
550 PRINT AT 21,11; INK 6; "MAX "; hiscor; AT 21,1; "SCOR ";
    tscor; AT 21,24; "VIETI "; vieti : RETURN
560 FOR i=USR "a" TO USR "b" +7
570 READ b
580 POKE i,b
590 NEXT i
600 RETURN
610 REM minge
620 DATA 0,60,126,126,126,126,60,0
630 REM caramida
640 DATA BIN 11111111
650 DATA BIN 10000001
660 DATA BIN 10111101
670 DATA BIN 10111101
680 DATA BIN 10111101
690 DATA BIN 10111101
700 DATA BIN 10000001
710 DATA BIN 11111111
```

Tenis

Acest joc poate fi jucat de doi jucatori, sau un jucator si calculatorul.

Pentru lansarea in executie, tastati RUN si apoi 1 sau 2 (pentru numarul de jucatori) dupa care puteti incepe jocul.

Instructiuni:

```
Jucatorul 1 - A - muta paleta in sus;
              Z - muta paleta in jos;
Jucatorul 2 - K - muta paleta in sus;
              M - muta paleta in jos;
```

Primul jucator care ajunge la 15 puncte cistiga.

Note pentru introducerea programului:

1. Sirul "66" din linia 150 este constituit din caractere grafice. Ele se obtin tastind GRAPH, apoi tastind caracterele cu tasta 6 si din nou GRAPH.

2. "8"-urile din liniile 50,250 si 540 sint de asemenea caractere grafice. Se obtin tastind GRAPH, apoi simultan CAPS SHIFT si 8 si din nou GRAPH.

3. "A"-ul din linia 330 se obtine tot intrind in modul grafic cu GRAPH, si apasind tasta A si din nou GRAPH.

```
10 PAPER 4 : INK 0 : BRIGHT 0 : BORDER 4
20 CLS
30 GOSUB 730
40 DIM x(2) : DIM y(2) : DIM p(2)
50 LET comp=1 : LET sc1=0 : LET sc2=0 : LET z$="0"
60 PRINT AT 2,9; INK 7; "TENIS"
70 PRINT AT 8,3; "CITI JUCATORI?(1/2)"
80 LET i$=INKEY$
90 IF i$="1" THEN PRINT AT 12,8;  "Utilizati A pentru  sus";
   AT 14,8; "si Z pentru jos" : GO TO 120
100 IF i$="2" THEN PRINT AT 10,3;  "Jucatorul 1 foloseste";
```

```
    AT12,5; "A pentru sus si Z pentru jos."; AT 14,3;
    "Jucatorul  2 foloseste":  AT 16,5;  "K pentru sus si  M
    pentru jos" : LET comp=0 : GO TO 120
110 GO TO 80
120 FOR i=0 TO 200 : NEXT i
130 LET x(1)=2 : LET y(1)=3
140 LET x(2)=29 : LET y(2)=18
150 LET e$="8" : LET f$="66"
160 PRINT AT 1,0;
170 GOSUB 400 : REM partea de sus
180 FOR i=3 TO 19
190 PRINT AT i,0; INK 6; f$; INK 0; TAB 30; INK 6; f$
200 NEXT i
210 PRINT AT 20,0;
220 GOSUB 400 : REM partea de jos
230 PRINT AT 0,0; INK 1; "Jucatorul 1 : 00"; AT 0,19; INK 2
    "Jucatorul 2 : 00"
240 LET n=INT (RND*2)
250 FOR i=1 to 2 : PRINT AT y(i), x(i); INK i; "8";
    AT y(i)+1,x(i); "8" : NEXT i
260 IF n=0 THEN LET xb=21 : LET dx=1 : GO TO 280
270 LET xb=19 : LET dx=-1
280 LET yb=12 : LET dy=INT (RND*3)-1
290 GOSUB 440 : REM muta paleta
300 LET xb=xb : LET yb=yb : LET scd=0
310 GOSUB 580 : REM misca mingea
320 PRINT AT yb,xb; INK 0; " "
330 PRINT AT yb,xb; INK 7; "A"
340 IF scd=0 THEN GO TO 290
350 PRINT AT yb,xb INK 0; " "
360 GOSUB 380
370 GO TO 240
380 PRINT AT 0,10;  INK 1;  7$ (TO 2 - LEN(STR$ (sc 1)));
    sc 1; AT 0,30; INK 2; z$(TO 2-LEN (STR$ (sc 2))); sc 2
390 RETURN
400 FOR i=1 TO 64
410 PRINT INK 5; e$
420 NEXT i
430 RETURN
440 LET a$=INKEY$
450 IF a$="a" THEN LET p(1)=-1
460 IF a$="z" THEN LET p(1)=2
470 IF comp=1 THEN LET p(2)=(2*(y(2)<y(b))-(y(2)>(yb))) :
    GO TO 500
480 If a$="k" THEN LET p(2)=-1
490 IF a$="m" THEN LET p(2)=2
500 FOR i=1 TO 2
510 LET a=ATTR (y(i)+p(i),x(i))
520 If p(i)=2 THEN LET p(i)=1
530 IF a=32 THEN PRINT INK 0; AT y(i),x(i); " ";
    AT y(i)+1,x(i); " " : LET y(i)=y(i)+p(i)
540 PRINT AT y(i),x(i); INK i;  8" AT y(i)+1,x(i); "8"
550 LET p(i)=0
560 NEXT i
570 RETURN
580 LET w=ATTR (yb+dy,xb+dx)
590 IF w=32 THEN LET xb=xb+dx : LET yb=yb+dy : RETURN
600 IF w=33 OR w=34 THEN LET dx=-dx : PLAY "V1507N1g" :
    LET dy=INT(RND*3)-1 : RETURN
610 IF w=38 THEN GO TO 640
620 IF w=37 THEN PLAY "V1507N1c" : LET dy=-dy
```

```
630 RETURN
640 PLAY "03V15#d" : IF dx>0 THEN LET sc1=sc1+1 : GO TO 660
650 LET scv2=sc2+1
660 LET d=(scv 1=15)+2*(c 2=15) : LET scd=1
670 IF d<.0 THEN GOSUB 380 : PRINT INK 7 : AT 10,8;
    "Jucatorul ";d;"cistiga"; AT 12,7; "Un nou joc? (d/n)";
    GO TO 690
680 RETURN
690 IF INKEY$=""THEN GO TO 690
700 IF INKEY$="d" THEN RUN
710 IF INKEY$="n" THEN STOP
720 GO TO 690
730 FOR i=0 TO 7
740 READ n
750 POKE USR "a"+i,n
760 NEXT i
770 RETURN
780 DATA 0,60,126,126,126,126,60,0.
```

4.34 Utilizarea ca si calculator

Din sumar:

Selectarea calculatorului
Introducerea numerelor
Utilizarea functiilor matematice
Initializarea variabilelor
Functii definite de utilizator
Iesirea din modul calculator

Pentru a utiliza modul calculator, selectati optiunea "Calculator" din meniul general (daca nu stiti cum, cititi capitolul 3).

Daca lucrati cu optiunea +3 BASIC, puteti selecta modul calculator alegind optiunea **Exit** din meniul de editare (prin aceasta reveniti la meniul general) si apoi selectati optiunea Calculator. Programul BASIC la care ati lucrat (cind ati selectat calculatorul) va fi memorat si regasit cind iesiti din calculator si reveniti la +3BASIC.

Cind selectati optiunea Calculator, pe ecran va apare scris jos **Calculator** si cursorul va fi in coltul din stinga sus.

Din acest moment, calculatorul este gata pentru introducere de date, tastati:

```
6 + 4
```

Cind tastati ENTER, va apare raspunsul 10. (Nu trebuie sa apasati semnul = ca la un calculator obisnuit). Cursorul va fi pozitionat la dreapta raspunsului si deci puteti introduce urmatoarea operatie. Deci cu cursorul la dreapta numarului 10, tastati:

```
/5
```

si obtineti raspunsul 2. Acum introduceti:

```
*PI
```

si va rezulta 6.2831853. Observati ca puteti utiliza functiile matematice implementate pe calculatorul Tim-S Plus. Pentru a verifica aceasta, tastati:

```
* ATN 60
```

si va rezulta 9.7648943.

Puteti sa si editati continutul ecranului. Mutati de exemplu cursorul la stinga, la inceputul liniei si introduceti INT, astfel incit linia va fi:

```
INT 9.7648943
```

si la apasarea tastei ENTER se va afisa raspunsul 9.

Introduceti acum:

```
1E6
```

si veti primi valoarea expresiei.

O alta trasatura a modulului calculator este ca permite atribuirea de valori variabilelor si apoi utilizarea acestora in calcule. Aceasta se realizeaza cu instructiunea LET (la fel ca la BASIC). Pentru exemplificare apasati ENTER si introduceti urmatoarele:

```
LET x=10
```

(trebuie sa apasati ENTER de doua ori pentru ca Tim-S Plus sa accepte atribuirea variabilei). Acum verificati ca variabila x a fost [illegible]ta si tastati:

```
x+90
```

apoi

```
+x=x
```

Daca ati utilizat +3BASIC inainte de a intra in calculator, atunci orice variabila utilizata in modul calculator, trebuie aleasa astfel incit sa nu intre in conflict cu cele utilizate de programul BASIC abandonat.

Nu este permisa utilizarea cuvintelor cheie BASIC ca nume de variabile.

Cind terminati lucrul in modul calculator, apasati tasta EDIT. Pe ecran vor apare cu titlul Optiuni, optiunile **Calculator** si **Exit**. Selectati Exit si va puteti reintoarce in meniul principal. Daca inainte de a intra in calculator ati lucrat la un program in +3 BASIC, va puteti reintoarce la program selectind optiunea '+3BASIC' din meniul principal. (Daca doriti sa continuati utilizarea calculatorului, selectati din nou optiunea 'Calculator').

Daca ati definit functii proprii (cu instructiunea DEF FN) cind ati lucrat la programul in BASIC, puteti apela acea functie din modul calculator.

De exemplu, daca ati definit:

```
9000 DEF FN c(n)=n*n*n
```

si ati iesit din +3BASIC si ati selectat modul calculator, puteti utiliza aceasta functie definita, la fel ca orice functie implementata pe calculatorul **Tim-S Plus**. Introduceti de exemplu:

```
FN c(3)
```

si calculatorul va afisa numarul 27 (cubul numarului 3).

4.35 Utilitare pentru modul de lucru Spectrum

4.35.1 Monitor dezasamblor - MONS

MONS este furnizat intr-o forma relocatabila. Se incarca la adresa de la care se doreste executarea, si se lanseaza in executie. Daca se doreste relansarea lui (din BASIC) atunci se executa de la o adresa cu 29 (zecimal) mai mare decit adresa originala. MONS are 5K lungime odata ce a fost relocatat, dar consuma aproape 6K pentru incarcare din cauza "tabloului de relocatari" a adreselor care urmeaza dupa codul principal. MONS contine propria sa stiva si e un program de sine statator. La intrarea in MONS pe ecran (panoul frontal)$se afiseaza diverse informatii in hexagesimal (adrese, date, etc.).

Adresele se pot converti in zecimal cu comanda SS+3. Cu toate acestea adresele trebuie introduse intotdeauna in hexagesimal. Comenzile se introduc de la claviatura in urma cursorului ">" in forma de litere mari sau mici. Unele comenzi, al caror efect poate fi dezastros daca sint utilizate gresit, necesita apasarea lui SS impreuna cu litera de comanda. Comenzile se executa imediat si nu este nevoie de incheierea lor cu ENTER.

Unele comenzi necesita introducerea de numere hexagesimale. Daca terminatorul e semnul "-" atunci se introduce forma negativa in complementul lui 2. Daca se introduc mai mult de 4 caractere, atunci doar ultimele 4 se iau in considerare. In orice moment ne putem reintoarece la interpretorul BASIC prin apasarea lui CS+1 (functia EDIT din BASIC).

4.35.1.1 Comenzi

SS+3 comuta baza de numeratie a adreselor afisate (hexazecimal->zecimal). Aceasta comanda afecteaza toate adresele afisate de MONS inclusiv cele generate in cursul dezasamblarii, dar nu schimba afisarea continutului memoriei, care se face intotdeauna in hexa.

SS+4 (sau $) afiseaza o pagina dezasamblata incepind de la adresa continuta in "MEMORY POINTER". La a 2-a apasare se face reintoarcerea la "panoul frontal" si orice alta tasta continua cu urmatoarea pagina de dezasamblat.

ENTER incrementeaza "MEMORY POINTER" cu 1. astfel incit cei 24 de bytes afisati sint concentrati in jurul unei adrese mai mari cu 1 decit cea precedenta.

CS+7 decrementeaza "MEMORY POINTER" cu 1.

CS+5 decrementeaza "MEMORY POINTER" cu 8.

CS+8 incrementeaza "MEMORY POINTER" cu 8.

> modifica "MEMORY POINTER" astfel incit sa contina adresa curenta a stivei (indicata de SP). Aceasta comanda e utila cind se doreste o privire de ansamblu asupra adreselor de intoarcere a rutinelor apelante.

G cauta in memorie un anumit sir (GET). Se afiseaza ":" si se introduce primul byte care se cauta, urmat de ENTER, si tot asa in continuare pina se defineste intreg sirul.

H converteste un numar zecimal in forma sa echivalenta hexazecimala.

I (copie fidela) este folosit pentru copierea unui bloc de memorie dintr-o zona in alta. Fidelitatea ei consta in aceea ca blocul de memorie poate fi copiat corect si in locatii unde cele

doua zone implicate in transfer se suprapun. "I" raspunde intrebind "FIRST:" care sint adresele (inclusiv) de inceput si sfirsit ale blocului care trebuie copiat si apoi "TO:" care, reprezinta adresa la care blocul trebuie copiat.

Daca adresa de start este mai mare decit adresa de sfirsit comanda nu se executa. Numerele se introduc in forma hexazecimala.

J executa codul de la adresa specificata. Comanda raspunde cu ":" asteptind un numar in hexagesimal care odata introdus duce la resetarea stivei interne, stergerea ecranului si transferul executiei la adresa specificata. Daca se doreste reintoarcerea la "panoul frontal" dupa executarea codului, se alege un punct de intrerupere cu comanda "W" in locul la care se doreste intoarcerea la ecran.

Nota:J corupe registrele Z80 inainte de a executa codul. Daca se doreste executarea codului cu anumite valori ale registrelor, se executa "SS+K" de mai jos.

SS+K continua executia de la adresa curenta a lui PC. Aceasta comanda s-ar utiliza cel mai frecvent impreuna cu "W".

L tabeleaza sau listeaza un bloc de memorie incepind de la adresa curenta continuta in MP.

Comanda sterge ecranul si afiseaza reprezentarea hexazecimala si echivalentul ASCII a 80 de bytes de memorie, incepind de la valoarea curenta a MP. Adresele se afiseaza in hexagesimal sau in zecimal in functie de starea lui "SS+3".

Ecranul este format din 20 de rinduri a 4 bytes/rind, codurile ASCII fiind plasate la sfirsitul rindului.

Pentru scopurile acestei comenzi orice octet din memorie este interpretat drept simbolul "." atunci cind valoarea lui nu se incadreaza in intervalul #20...#7F. Octetii ai caror valori apartin acestui interval sint interpretati conform generatorului de caractere prezentat anterior.

La sfirsitul paginii ne putem reintoarce la "panoul frontal" cu "CS+5", sau continua listarea, prin actionarea oricarei alte taste.

M seteaza MP pe adresa specificata. Raspunde cu ":" si asteapta un numar in hexagesimal. MP este actualizat cu adresa introdusa si imaginea "panoului frontal" se modifica corespunzator.

N initiaza editarea urmatorului sir specificat de comanda "G".

Comanda "G" permite definirea unui sir, urmata de cautarea primei sale aparitii. Comanda "N" permite cautarea urmatoarelor aparitii. Cautarea incepe de la MP si actualizeaza ecranul la aparitia sirului cautat.

O merge la destinatia unei deplasari relative. Comanda ia byte-ul adresat curent de MP si-l trateaza ca pe un deplasament relativ, actualizind ecranul corespunzator. De retinut ca deplasarile mai mici de #7F (127) sint tratate ca negative de microprocesorul Z80, fapt de care comanda "O" tine cont (vezi comanda "U").

P umple memoria dintre limitele specificate cu un octet specificat.

Intreaba "FIRST:", "LAST:" si "WITH:". Se introduc numerele in forma kexazecimala: adresa de inceput si de sfirsit a blocului de memorie care trebuie umplut si byte-ul cu care se doreste umplerea blocului de memorie.

Q cauta setul de registre. La intrarea in "panoul frontal" setul de registre afisat este cel standard (AF, HL, DE, BC). Folosirea lui "Q" va afisa setul alternativ de registre (AF/, HL/, DE/, BC/) care se poate distinge de cel standard prin semnul

"/" dupa numele registrului. Daca "Q" este folosit cind se afiseaza registrul alternativ, se trece la cel standard.

SS+T stabileste un punct de intrerupere dupa instructiunea curenta si continua executia.

T dezasambleaza o portiune de cod, optional la imprimanta. Prima data intreaba "FIRST:" si "LAST:" adresele codului pentru care se doreste dezasamblarea, in hexagesimal. Daca adresa de inceput este mai mare decit adresa de sfirsit, comanda nu se executa. Dupa introducerea acestor adrese se intreaba "Printer?". Se raspunde cu Y (capital) daca se doreste ca dezasamblarea sa fie directionata spre imprimanta sau cu orice altceva daca se face pe ecran. Urmeaza intrebarea "Text?" pentru a introduce in hexagesimal adresa de start a fisierului text pe care-l produce dezasamblarea. Daca nu se doreste generarea unui fisier text se apasa ENTER la aceasta intrebare. Fisierul se produce intr-o forma in care poate fi utilizata de catre asamblorul GENS3. Daca se doreste utilizarea textului cu GENS3 trebuie generat - sau mutat - la prima adresa data de comanda "X" a asamblorului, deoarece aceasta este adresa de start a "fisierului text" asteptat de GENS3. Trebuie specificata si adresa de sfirsit a textului. Aceasta se face luind adresa "End of text" data de dezasamblor si punind-o in locatia TEXTEND a lui GENS3 (vezi prezentarea lui GENS3, urmatorul paragraf). Apoi se introduce GENS3 cu startul cald pentru a pastra textul. Daca pe durata generarii fisierului text acesta ar ajunge sa se scrie peste MONS, dezasamblarea este abandonata si se apasa orice tasta pentru revenirea la "panoul frontal".

Daca se specifica o adresa pentru fisierul text se intreaba in continuare "workspace:" - adresa spatiului de lucru - care ar trebui sa fie inceputul locului gol al memoriei, care este folosit pentru o tabela de simboluri primitive pentru orice label (eticheta) generat in procesul dezasamblarii. Cantitatea de memorie necesara este de 2 bytes pentru orice label generat. Daca se apasa ENTER, adresa subinteleasa este #6000 in hexagesimal.

Apoi se intreaba in mod repetat "FIRST:" si "LAST:" adresele (inclusive) pentru blocurile de date care la dezasamblare vor fi interpretate ca functie DEFB (define byte).

Daca valoarea byte-ului de date este intre 32 si 127 (#20 si #7F) inclusiv, atunci se da respectivului octet interpretare in caractere ASCII. Cind s-au terminat domeniile de date specificate sau daca nu se specifica nici un domeniu se apasa ENTER pentru ambele intrebari.

Comanda "T" foloseste domeniului de la sfirsitul lui MONS pentru a inmagazina adresele domeniilor de date, astfel incit se pot alege atitea domenii de date cita memorie disponibila exista; fiecare domeniu de date necesita 4 bytes pentru inmagazinare. De remarcat ca aceasta comanda distruge punctele de intrerupere (comanda "W").

In acest moment, ecranul va fi sters. Daca s-a cerut crearea fisierului text va urma o scurta intirziere (dependenta de marimea sectiunii de memorie care trebuie dezasamblata) cauzata de construirea tabelului de simboluri. Odata construit acest tabel, listingul dezasamblat va aparea pe ecran sau la imprimanta. Listarea poate fi intrerupta cu tastele ENTER sau SPACE respectiv CS+5 pentru intoarcerea la "panoul frontal", sau orice alta tasta (mai putin CS+1) pentru a continua dezasamblarea. Daca un "opcode" invalid este intilnit, este dezasamblat ca un NOP si urmat de un "*" (asterix) dupa "opcode"-ul (codul instructiei Z80, sau codul operatiei) din listing.

La sfirsitul dezasamblarii ecranul va astepta si daca s-a cerut producerea unui fisier text se afiseaza mesajul "End of

text XXXXX", XXXXX fiind adresa in hexagesimal sau zecimal la care trebuie incarcata ("L" mai intii) adresa fisierului text, in GENS3, la locatia TAXTEND pentru ca ansamblorul sa poata "culege" adresa fisierului text la un start cald. Cind s-a incheiat dezasamblarea se apasa tasta CS+5 pentru intoarcere la "panoul frontal" sau CS+1 care (re)trimite in BASIC.

Label-urile sint generate (cind sint relevante) in forma L XXXXX, unde XXXXX este adresa absoluta in hexagesimal a labelului, dar doar daca respectiva adresa se afla in limitele dezasamblarii. Daca adresa se afla in afara acestui domeniu, nu se genereaza nici un label ci se da pur si simplu adresa zecimala sau hexagesimala. Daca o anume adresa s-a referit la o instructiune in dezasamblare atunci label-ul va aparea in cimpul labelului (inainte de mnemonic) doar daca listingul este directionat la un fisier text.

U folosit impreuna cu comanda "O".

Dupa cum s-a precizat comanda "O" actualizeaza ecranul corespunzator unei deplasari relative (de regula pentru a arata efectul unei instructiuni de tip JR sau JR NZ). "U" este folosit pentru a actualiza ecranul cu valorile dinaintea executarii comenzii "O" (a ultimei comenzi "O").

V este folosit in combinatie cu comanda "X". Este similara comenzii "U", cu deosebirea ca actualizeaza ecranul acolo unde era inainte de executia ultimei comenzi "X".

W determina un punct de intrerupere la adresa MP. Un "punct de intrerupere" - din punctul de vedere al lui MONS - este o instructiune de tip CALL la o subrutina (temporara) care afiseaza "panoul frontal" permitind programatorului sa opreasca executia programului si sa inspecteze registrele, flag-urile si orice adrese relevante. Cei 3 bytes ai rutinei de CALL sint inlocuiti cu cei originali imediat dupa executia opririi. MONS foloseste spatiul de la sfirsit pentru actiunea de creare a rutinelor temporare si deci se pot alege atitea puncte de intrerupere cita memorie mai exista disponibila, de la sfirsitul lui pina la prima locatie de memorie utilizata. Fiecare punct de intrerupere necesita 5 bytes pentru inmagazinare.

Nota: Atentie la locul unde se alege un punct de intrerupere pentru a nu patrunde in corpul unei instructiuni pe mai multi octeti. Mai exact, daca vreti sa evitati surorizele neplacute, atunci faceti asa fel incit comanda W sa fie actionata numai in momentul in care cursorul este pozitionat pe primul cod al unei instructii.

X folosita pentru a actualiza MP cu destinatia unei instructiuni de tip CALL sau JP absolut.

X ia o adresa (pe 16 biti) specificata de la MP si MP+1 si apoi actualizeaza ecranul astfel incit sa fie centrat in jurul acestei adrese. De retinut ca primul byte este cel mai putin semnificativ (vezi comanda "V").

Y introduce ASCII de la MP. "Y" da o noua linie in care se pot introduce caractere ASCII direct de la tastatura. Acestea se introduc in forma hexazecimala echivalenta si se incarca in memorie incepind de la valoarea curenta a lui MP. Sirul de caractere trebuie terminat cu CS+5. Cu DELETE (CS+0) se poate sterge ultimul caracter din sir. Cind s-a terminat introducerea caracterelor si CS+5 ecranul se actualizeaza astfel incit MP este pozitionat imediat dupa sfirsitul sirului in memorie.

SS+Z executie pas cu pas. Inainte de folosirea acestei comenzi atit PC cit si MP trebuie stabilite la adresa instructiunii care se doreste executata. Comanda executa instructiunea curenta si actualizeaza "panoul frontal" pentru a reflecta schimbarile determinate de executarea instructiunii.

Se poate merge pas cu pas si in RAM si in ROM, dar numai daca intreruperile sint dezactivate.

" (SS+P) Aceasta comanda este exact ca si "L". (LIST), cu deosebirea ca iesirea se face la canalul imprimantei in loc de ecran. De retinut ca la sfirsitul paginii se apasa CS+5 pentru intoarcerea la "panoul frontal" sau orice alta tasta pentru urmatoarea pagina (mai putin CS+1)..

4.35.1.2 Modificarea memoriei

Continutul adresei date de MP poate fi modificat prin introducerea numarului hexagesimal de la un terminator. Daca terminatorul nu este ceva valid comanda nu se executa.

4.35.1.3 Modificarea registrelor

Daca un numar hexagesimal este introdus ca raspuns la intrebarea ">" si este terminat cu ".", atunci numarul specificat va fi introdus in registrul Z80 adresat curent de sageata "->". La intrebarea MONS "->" (indicind PC) introducind "." ca terminator al numarului hexagesimal, el va modifica PC. Daca se foloseste "." de unul singur (nu ca terminator) pointerul se va roti circular de la PC la AF. Nu este posibila adresarea (deci nici schimbarea) SP (STACK pointer, registrul indicator de stiva) sau a registrului IR (registrul vectorului de intrerupere). Simbolul "." poate fi folosit si pentru modificarea setului alternativ de registre, daca acesta este afisat. Se foloseste comanda "@" pentru a comuta setul de registre.

4.35.1.4 Ce este panoul frontal?

Primele 9 linii ale display-ului contin registrele Z80, numele lor, valoarea lor prezenta si continutul a 7 locatii de memorie, incepind de la adresa specificata de registru. Registrul cu FLAG-urile este decodificat pentru a arata FLAG-urile setate.

Pointer-ul de registru "->" arata registrul adresat curent; cei 24 bytes afisati in partea de jos, sint organizati ca adrese urmate de continutul lor, centrate in jurul valorii marcate >...<.

4.35.2 Asamblorul - GENS

GENS este un asamblor Z80, usor de utilizat, asemanator cu asamblorul ZILOG standard. Are lungimea de 7K (GENS3M are 9K), odata relocat ceva mai putin si foloseste stiva proprie. GENS contine propriul sau editor, care plaseaza fisierul text (sursa) imediat dupa zona de memorie pe care o ocupa, urmata de tabela de simboluri. Se recomanda incarcarea in partea inferioara a memoriei.

4.35.2.1 Generalitati

Se incarca cu **LOAD "" CODE xxxxx**
Se lanseaza cu **RANDOMIZE USR xxxxx** (prima data)
si-i relocabili **RANDOMIZE USR xxxxx + 56** (start rece), sau
RANDOMIZE USR xxxxx + 64 (start cald).

La inceput apare mesajul: "Buffer size?:". Se introduce un numar intre 0 si 9 inclusiv urmat de ENTER sau numai ENTER, pentru valoare implicita. Numarul introdus reprezinta factorul de multiplicare a 256 de octeti. Daca se doreste minimizarea spatiului ocupat de GENS si spatiul sau de lucru (si nu se ia in considerare folosirea eficienta a optiunii INCLUDE), se poate apasa 0, asigurind cel mai mic buffer posibil (64 octeti).

In continuare apare simbolul ">", care indica intrarea in asamblor.

Atentie: GENS dezactiveaza intreruperile si corupe valoarea registrului IY.

4.35.2.2 Detalii

La apelarea asamblorului (comanda A) acesta intreaba la inceput (Table size?:". Raspunsul este un numar (in zecimal) care reprezinta cantitatea de memorie ce va fi alocata pentru tabelele de simboluri. Valoarea implicita (obtinuta prin apasarea lui ENTER) se apreciaza de asamblor functie de lungimea textului si este, in general, perfect acceptabila. Folosind optiunea INCLUDE trebuie specificata o tabela de simboluri mai mare decit cea implicita, asamblorul neputind anticipa marimea fisierului care va fi inclus.

Urmeaza cererea optiunilor, cu mesajul "Optional":

1 - produce listarea tabelei de simboluri la sfirsitul celei de a doua treceri a asamblorului;
2 - nu genereaza cod obiect;
4 - nu listeaza programul asamblat;
8 - listeaza programul asamblat la imprimanta.
16 - pune codul obiect (daca e generat) dupa tabela de simboluri; numaratorul de locatii este controlat de directive ORG, deci codul obiect poate fi plasat intr-o portiune de memorie care nu este aceiasi cu zona pentru care a fost gindit sa ruleze;
32 - nu mai verifica unde se plaseaza codul obiectului (util pentru asamblari rapide).

Daca s-ar utiliza optiunea 16, directiva ENT nu mai are efect.

Adresa de inceput a codului obiect se poate afla folosind comanda "X" pentru a gasi sfirsitul textului, la valoarea obtinuta adaugindu-se 2.

Asamblarea are loc in doua treceri. La prima trecere, GENS cauta erorile si completeaza tabelele de simboluri. La a doua trecere se genereaza codul obiect, mai putin daca s-a folosit optiunea 2. La prima trecerea nu se afiseaza nimic pe ecran sau pe printer decit daca se detecteaza o eroare. In acest caz se afiseaza numarul liniei care contine eroarea impreuna cu un cod al erorii; asamblarea se opreste si se poate apasa "E" pentru intoarcerea la editor sau oricare alta tasta pentru a continua asamblarea. La sfirsitul primei treceri apare mesajul "Pass 1 errors: nn" si in cazul in care sint erori nu se trece mai departe. Poate aparea si mesajul "WARNING label absent" pentru fiecare label care lipseste. Dupa a doua trecere se genereaza listingul asamblarii (mai putin daca s-a utilizat optiunea 4).

Listingul asamblat este in general de forma:

```
COOD    210100    25 label
                    ld HL,1
```

1 6 15 21 26 (nr. coloana)

Primul cimp al liniei reprezinta valoarea numaratorului de locatii la inceputul lucrului la linie, mai putin daca mnemonicul din aceasta linie este un pseudomnemonic (ORG EQU ENT) in care caz va reprezenta valoarea din cimpul operandului din instructie. In general valoarea se afiseaza in hexagesimal dar se poate afisa si in zecimal, fara semn, prin folosirea comenzii "*D+" a asamblorului.

Urmatorul cimp din coloana 6, care poate avea pina la 8 caractere lungime (deci 4 octeti), este codul obiect produs de instructiunea curenta (atentie la comanda asamblorului "*C").

Urmeaza numarul de linie, intreg intre 1 si 32767 inclusiv.

Coloanele 21-26 din prima linie contin primele 6 caractere ale oricarui label definit in aceasta linie.

Dupa fiecare label urmeaza o noua linie. Pe aceasta linie mnemonicul este afisat intre coloanele 21-24.

Urmeaza cimpul operandului din coloana 26 a acestei linii si continutul care trebuie incarcat la sfirsitul liniei, generind noi linii cind e necesar. Formatul de mai sus ajuta la redactabilitatea listingului asamblarii pe un ecran ingust ca cel al SPECTRUM-ului, fara a-si defini propriul set de caractere, care ar duce la ocuparea unui spatiu nejustificat de mare de GENS si la imposibilitatea folosirii rutinelor din softul de baza (48K BASIC).

Comanda "xC" data asamblorului produce o linie de asamblare mai scurta, prin faptul ca omite cele 9 caractere reprezentind codul obiect al liniei, astfel incit majoritatea liniilor asamblate sa incapa intr-o singura linie listata. Se poate modifica impartirea liniei modificind (cu instructiunea POKE) urmatoarele trei locatii din GENS:

START+51 - numarul de caractere continute intr-o linie;
START+52 - coloana de la care incepe fiecare linie pe ecran;
START+53 - cite caractere din reminder-ul liniei asamblate se afiseaza pe fiecare ecran, dupa prima linie.

Exemplu: sa presupunem ca am vrea ca prima linie a oricarei linii asamblate sa contina 20 de caractere (fara cimpul labelului) si fiecare linie ce urmeaza sa inceapa in coloana 1 umplind intreaga linie. Presupunind ca GENS e incarcat de la 24064 se reda controlul interpretorului BASIC si se tasteaza:

```
POKE 24115,20
POKE 24116,1
POKE 24117,31
```

Modificarile se pot aplica doar daca nu s-a folosit comanda "xC".

Listingul asamblarii se poate intrerupe cu CS+SPACE. Apasind "E" se trece in editor. Daca vrem sa continue apasam orice alta tasta. Singurele erori ce pot apare la a doua trecere sint "EROR 10" si "BAD ORG". Eroarea 10 nu este fatala, asamblarea putind fi continuata. La sfirsit apare "Pass 2 errors:nn" si mesajul de atentionare pentru labelurile inexistente, de asemenea mesajul "Table used xxxxx from yyyyy". Daca s-au folosit directive ENT in mod corespunzator, apare mesajul "Execute nnnnn" care reprezinta locul de unde programul se poate executa cu comanda "R". Daca s-a specificat optiunea 1 se afiseaza si o lista alfabetica a label-urilor folosite si a valorilor lor asociate.

Numarul label-urilor afisate se poate schimba prin POKE

START+50, valoarea dorita". In continuare controlul revine editorului.

4.35.2.3 Formatul instructiunii

```
art    LD     HL,label   ;pick  up     "label"
¦ un simbol ce reprezinta 16 biti de informatie
       ¦ mnemonic
              ¦ operand
                          ¦ comentariu
```

Prin ¦ am marcat debutul unui cimp specific din format.

Daca un label este asociat cu o valoare peste 8 biti si este apoi utilizat intr-un context in care ar trebui sa aiba 8 biti, apare "ERROR 10" la a doua trecere. Pentru labeluri sint legale:

```
literele (A - Z; a-z);
cifrele  (0-9);
semnele  (1 - etc.)
```

cu mentiunea ca un label trebuie sa inceapa cu o litera.
Exemple: **LOOP, loop, a, b2, ...**

4.35.2.4 Contorul de locatii

Asamblorul mentine contorul de locatii astfel incit unui simbol din cimpul labelului sa-i fie asociata o adresa si apoi sa fie introdus in tabela de simboluri. Acest contor de locatii poate fi initializat la orice valoare, conform directivei ORG. Simbolul **$** poate fi folosit pentru a se referi la valoarea curenta a contorului de locatii: (LD HL +5).

4.35.2.5 Tabela de simboluri

Cind un label e intilnit pentru prima data, el se intoarce intr-un tabel impreuna cu 2 indicatori ce-i arata valoarea asociata....etc. Acest tip de tabela se numeste "Binary True Simbol Table". Lungimea unei noi intrari este de 8 pina la 13 octeti, functie de lungimea simbolului.

4.35.2.6 Expresii

O expresie este un operand constituit dintr-un singur termen, sau o combinatie de termeni separati de catre un separator. Exemple:

```
TERMENI: constante zecimale 1029;
         constante hexa     $405;
         constante binare   %1011100001;
         caracter constant  "a";
         label I029.
$ arata valoare curenta a contorului de locatii

OPERATORI: +;
           -;
           & (si);
           ? (sau);
           ! (sauexclusiv);
```

```
x (inmultire intreaga);
\ (impartire intreaga);
? (MOD) (a?b=a-(a/b)xb)).
```

La citire se iau cei mai putin semnificativi 16 biti. Expresiile se evalueaza strict de la stinga la dreapta. O expresie dintre paranteze inseamna o referire la continutul dintre paranteze. Domeniul valorilor relative este -128, +127 de la instructiunea de dupa valoarea din cimpul operandului. Se poate folosi $ pentru domeniul -126, 129.

Exemple:
```
/5000 label;
%100101%1011;
"A" + 128;
"Y" - ";" +7;
$-label+C.
```

Se pot insera spatii intre termeni si operatori. Daca apar valori nepermise se semnaleaza "ERROR 15" (depasire la inmultire) sau "ERROR 14" (impartire cu 0). In alte cazuri depasirea se ignora.

4.35.2.7 Directivele asamblorului

ORG expresie
EQU expresie - trebuie precedat de un tabel caruia ii da valoarea expresiei;
DEFB expresie... - defineste o constanta pe 8 biti;
DEFW expresie... - defineste o constanta pe 16 biti (cel mai putin semnificativ, LSB, este primul);
DEFS expresie - creste valoarea contorului de locatii cu valoarea expresiei (pentru a rezerva loc in memorie);
DEFM "s" - defineste continutul a n octeti de memorie egal cu reprezentarea ASCII a sirului s, unde n este lungimea sirului, ce poate fi teoretic cuprinsa intre 1 si 255 inclusiv, desi practic e limitata de lungimea maxima a liniei ce poate fi introdusa din editor; primul caracter din cimpul operandului (in cazul nostru ") e considerat ca separator si lungimea este data intre doi separatori; caracterul "sfirsit de linie" actioneaza ca sfirsit de sir;
ENT expresie - stabileste adresa de executie a codului obiect asamblat la valoarea expresiei; se foloseste impreuna cu comanda "R" a editorului si nu are valoarea implicita.

4.35.2.8 Pseudomnemonice conditionate

Acestea permit programatorului sa includa sau nu anumite sectiuni ale textului in procesul asamblarii.

IF expresie - evalueaza expresia si daca rezultatul este 0 asamblarea se opreste pina cind se intilnesc ELSE sau END; pentru o valoare diferita de 0 se continua asamblarea;
ELSE - daca asamblarea este pornita (ON), inainte de ELSE ea se opreste (OFF), si viceversa.

4.35.2.9 Comenzile asamblorului

Spre deosebire de directive, comenzile nu au efect asupra codului obiect; sint linii ale textului sursa care incep cu "a".

*E - (EJECT) se trimit la ecran sau imprimanta 3 linii albe. Este util in separarea modulelor.

*Hs - face ca sirul s sa fie luat ca "heading" tiparit dupa fiecare EJECT (xE), xH face automat si xE.

*S - determina oprirea listarii la aceasta linie. Listarea reincepe apasind orice tasta. Comanda este utila la citirea adreselor in mijlocul unui listing si este recunoscuta si dupa comanda "xL", neoprind tiparirea.

*L - face ca listarea si tiparirea sa se opreasca dupa aceasta linie.

*L+ - listarea si tiparirea reincep dupa aceasta linie.

*C- - scurteaza listingul asamblarii dupa linia urmatoare. namaiafisind codul obiect generat.

*C+ - revine la listarea completa a asamblarii.

*F - (filename); permite asamblarea textului de pe banda.

Fisierul text e introdus intr-un buffer pe blocuri si asamblat de acolo. Astfel, sint posibile coduri obiect lungi, deoarece textul odata asamblat nu mai ocupa mult spatiu in memorie. "Filename" are pina la 10 caractere si trebuie precedat de un spatiu. Daca nu se precizeaza "filename" se incarca primul gasit. Acest text trebuie pus pe banda cu comanda editorului "T" si nu cu comanda "P".

Lungimea blocului din buffer se ia in multiplii de 156 octeti.

Valoarea implicita este 4x256=1024 octeti, pentru care trebuie specificat acelasi "Buffer size" (marime a bufferului). Comanda "F" actioneaza la ambele treceri.

4.35.2.10 Editorul

Este transparent pentru utilizator si comprima spatiile, avind urmatoarele functii:

ENTER - ENTER-ul de pe SPECTRUM;
CC - CS+1 - renunta la intrare;
CH - CS+0 - sterge inapoi;
CI - CS+8 - avanseaza la urmatorul TAB;
CX - CS+5 - uita linia introdusa.

La cererea editorului - semnalata cu "?" - se poate raspunde cu o comanda de urmatoarea structura:

C N1, N2, S1, S2 si/sau ENTER

unde C este comanda de executat, N1 si N2 numere intre 1 si 32767, iar S1 si S2 siruri.

4.35.2.11 Comenzile editorului

Inserare de text - textul poate fi inserat in fisierul sursa introducind un numar de linie, un spatiu si apoi textul dorit, sau prin folosirea comenzii "I". Daca se scrie numai un numar de linie si apoi se apasa ENTER textul de la linia respectiva se sterge. Oricind se introduce un text, se pot folosi functiile de control **CX** (sterge de la inceputul liniei), **CI** (trece la urmatorul TAB) si **CC** (intoarce la bucla de comanda). Tasta DELETE

va produce o stergere inapoi (nu inainte de inceputul liniei de text). Textul se introduce intr-un buffer intern a lui GENS iar daca aceasta se umple nu se mai poate introduce alt text si trebuie folosite functiile CH sau CX pentru a face loc in buffer. In timpul inserarii textului editorul detecteaza daca sfirsitul textului se apropie de virful RAM-ului, caz in care afiseaza mesajul "BAD MEMORY". Aceasta arata ca nu mai poate insera text si fisierul sursa sau cel putin o parte din el trebuie salvat pe banda, pentru o redare ulterioara.

I n,m - folosirea acestei comenzi determina intrarea in mod automat de inserare, cu incepere de la linia n si incrementul m. Pentru a iesi din acest mod se foloseste functia CC (EDIT). Daca se specifica numarul unei linii existente, aceasta va fi stearsa. Daca se depaseste 32767 **modul** de inserare automata se opreste. Daca introducind textul se ajunge la capatul ecranului cu o linie, fara a introduce toate cele 64 de caractere (marimea bufferului), ecranul se va face scroll si se poate continua.

Listarea textului. Se face cu comanda I, iar modificarea numarului de linii ce se listeaza odata se poate face cu comanda "X".

L n,m - listeaza textul de la linia n la linia m. Valoarea implicita a lui n este intotdeauna 1 iar, cea a lui m 32767 si nu valorile precedente. Pentru a lista intreg textul se introduce comanda "L" fara argumente. Liniile se formeaza cu o margine la stinga. Tabularea liniei este automata realizind o separare clara a diferitelor cimpuri. Numarul de linii listate dintr-o data pe ecran poate fi controlat din comanda "K".

K n - aceasta comanda stabileste numarul de linii care se afiseaza (sau listeaza) deodata inainte de pauza. Valoarea se inregistreaza in (n MOD 256). De exemplu K5 produce listarea a 5 linii deodata.

Editarea textului

O data ce textul a fost creat, va urma inevitabil nevoia de a edita o parte din el. Se pot folosi urmatoarele comenzi:

D n,m - toate liniile de la n la m inclusiv sint sterse din fisierul text. Daca n>m sau se specifica mai putin de doua argumente, nu se executa nimic, pentru a evita stergerile din greseala. O singura linie poate fi stearsa introducind numarul de linie si ENTER.

M n,m - textul de la linia n este introdus la linia m, stergind continutul acesteia. Linia n nu se modifica. Daca nu exista numarul de linie specificat, nu se executa nimic.

N n,m - folosirea comenzii "N" face ca fisierul text sa fie renumerotat de la linia n cu incrementul m. Trebuie specificate atit n cit si m.

F n,m,f - textul dintre liniile n si m este cautat dupa sirul f. Daca acest sir este gasit, linia in care este gasit se editeaza. In modul de editare se poate cauta urmatoarea aparitie sau se poate in prealabil modifica textul si apoi sa se treaca la urmatoarea aparitie. Comanda isi pastreaza parametrii si pentru repetarea ei este sufucient sa se apese din nou E.

E n - editeaza linia cu numarul n. Daca n nu exista, nu se executa nimic. In caz contrar linia se copiaza intr-un buffer si aici se poate lucra la ea, linia originala raminind in tot acest timp neschimbata.

Subcomenzi

- **SPACE** - incrementeaza "text pointerul" (TP - indicatorul de

text) cu o unitate. Nu se poate depasi ultimul caracter.

DELETE - decrementeaza TP cu o unitate, mergind inapoi pe linie. Nu se poate reveni inaintea primului caracter.

CS+S - cauta urmatorul TAB.

ENTER - opreste editarea mentinind toate modificarile facute.

Q - paraseste editarea ignorind modificarile facute.

R - reincarca bufferul cu text, ignorind modificarile facute.

L - listeaza restul liniei care se editeaza (de exemplu comentariul). Se ramine in modul de editare cu TP repozitionat la inceputul liniei.

K - (KILL) sterge caracterul de la pozitia curenta a TP.

Z - sterge toate caracterele de la (inclusiv) pozitia curenta a TP pina la sfirsitul liniei.

F - gaseste urmatoarea aparitie a sirului definit cu comanda "F".

S - substituie sirul definit cu comanda "F".

I - insereaza caractere de la pozitia curenta a TP. Se ramine in acelasi mod pina la apasarea tastei ENTER, cind se revine in modul de baza de editare cu TP pozitionat pe ultimul caracter inserat.

X - avanseaza TP la sfirsitul liniei si trece in modul de inserare.

C - permite rescrierea caracterului de la pozitia curenta a TP si apoi avansarea acestuia. Se ramine in acest mod pina la apasarea tastei ENTER, cind se revine la modul de editare cu TP pozitionat dupa ultimul caracter modificat.

4.35.2.12 Comenzile casetofonului

P n,m,s - domeniul dinre liniile n si m inclusiv este salvat pe banda cu numele de fisier specificat de sirul s. Argumentele isi pastreza valoarea data de comanda precedenta.

G,s - se cauta pe banda un fisier cu numele s. Cind acesta este gasit, se incarca la sfirsitul textului curent. Daca nu se specifica nici un nume se va incarca primul fisier de pe banda. Dupa ce se da comanda G apare mesajul "Start tape..."; se apasa PLAY la casetofon si incepe cautarea fisierului. Pot sa apara mesajele "Using filename" sau "Find filename". Daca exista deja un text in memorie, cel de pe banda se va adauga la acesta.

Tn,m,s - aceasta comanda trimite blocul de text dintre liniile n si m pe banda, intr-un format adecvat includerii ulterioare prin comanda asamblorului "F". Fila este salvata cu numele de fisier s. Trimiterea incepe imediat dupa apasarea tastei ENTER. Aceasta comanda nu se poate folosi ca inlocuitor al comenzii P.

4.35.2.13 Comenzile de lucru cu discul

Se pot utiliza exact aceleasi comenzi ca si la casetofon cu deosebirea ca trebuie specificat si numarul discului, iar numele fisierului e obligatoriu si la comanda G. Formatul comenzilor poate fi:

```
P,n,m,nr:s;
G,,nr:s;
Tn,m,nr:s.
```

unde nr reprezinta numarul discului. Toate celelalte observatii de la comenzile caracterului ramin valabile, mai putin cele de la mesajele ce pot sa apara. In plus poate sa apara mesajul "file not found".

Nota: salvarea pe disc se face ca si bloc de date si nu ca pe banda, unde se salveaza ca si cod masina. Se pot face inregistrari cu acelasi nume pe acelasi disc, caz in care inregistrarea veche se pierde. De asemenea, daca se salveaza o rutina care are mici modificari fata de una care exista deja pe disc, sint salvate doar aceste modificari, iar numele sub care e salvata aceasta fila nu apare la comanda "CAT", desi in unele cazuri fila poate fi reincarcata cu comanda G. Formatul cu care se scrie pe disc poate fi vizualizat din BASIC cu comanda:

```
MOVE""m":nr;"a"TO=2".
```

Codul obiect obtinut in urma unei asamblari reusite se poate salva pe disc cu ajutorul comenzii O.

Pentru cazul in care se doreste salvarea pe disc a unui cod obiect obtinut in urma asamblarii unui program sursa sub GENS se completeaza comanda:

```
O
```

Comanda cere in prealabil o salvare a fisierului sursa pe disc, dar atentie, codul obiect va fi inregistrat pe disc sub numele ultimului fisier sursa salvat pe acelasi disc. Ulterior, acelasi cod obiect se poate prelua de pe disc ca si un pachet de date obisnuit.

4.35.2.14 Asamblarea si rularea in editor

A - asambleaza textul incepind de la prima linie.

R - daca sursa a fost corect asamblata, fara erori, si adresa de executie specificata prin directiva ENT, atunci comanda executa programul obiect. Daca programul obiect contine o instructiune de tip RET, la sfirsitul executiei, se poate face intoarcerea in editor atit timp cit nu s-a modificat pozitia stivei.

4.35.2.15 Alte comenzi

B - reda controlul sistemului de operare. Reintrarea se face de regula la cald, dar nu ne-mpiedica nimeni s-o facem si la rece.

C - permite convertirea fisierelor text produse de GENS la forma comprimata a lui GENS. Se incarca fisierul cu GENS1. Se comprima si se salveaza cu T. Executia acestei comenzi este lunga si comanda nu are argumente.

S,,d - permite schimbarea separatorului argumentelor intr-o linie de comanda. Separatorul nu poate sa fie spatiu si este implicit ";".

V - afiseaza N1, N2, S1, S2 cu valorile lor curente.

W,n,m - sectiunea de text dintre n si m este listata la imprimanta. Listarea se face conform comenzii K, reactivindu-se dupa apasarea oricarei taste.

X - afiseaza in zecimal adresele de inceput si sfirsit ale fisierului text.

4.35.2.16 Codul erorilor

4.35.2.16.1 Erori tipice

Am cuprins in aceasta categorie acele erori pe care dumneavoastra aveti cele mai mari sanse de a le face...

1 - eroare in contextul liniei;
2 - mnemonic necunoscut;
3 - instructiune formulata gresit;
4 - simbol multidefinit;
5 - linia contine caracter ilegal;
6 - operand ilegal;
7 - un simbol e cuvintului rezervat;
8 - mismasuri cu registrele;
9 - prea multe registre inr-o linie;
10 - o expresie ce ar trebui sa evalueze 8 biti are de evaluat mai mult;
11 - instructiuni JP (IX+n) sau JP (IY+n) ilegale;
12 - eroare in formarea unei directive;
13 - referinta ilegala (SGU cu un simbol inexistent);
14 - impartire cu 0;
15 - depasire de inmultire.

4.35.2.16.2 Erori speciale

In general, aceste erori sint caracteristice numai celor initiati. Dar, subliniem, numai in general. Asta-nseamna ca ele pot fi si la indemina celor care au depasit nivelul de cunoastere al celor initiati, uneori chiar mai mult de cit ne-am putea noi inchipui.

BAD ORG - directiva ORG ar duce la stricarea fisierului text sau a tabelei de simboluri.
Out of table space (No table space) - nu s-a alocat suficienta memorie pentru tabela de simboluri.
BAD memory - nu mai este loc pentru text.

4.35.2.17 Cuvinte rezervate

In aceasta categorie intra simbolurile a caror utilizare nu este permisa sub forma unor cuvinte cheie (cum ar fi etichetele), deoarece reprezinta repere de lucru ale asamblorului, in procesul asamblarii. Urmeaza lista acestor simboluri:

A, B, C, D, E, H, L, I, R, S, AF, AF', BC, DE, HL, IX, IY, SP, NC, Z, NZ, M, P, PE, PO.

5 Extensia de limbaj BASIC Interface I

Informatiile acestui capitol au la baza prima versiune Interface I implementata pe Tim-S Plus.

Modul de lucru **Interface I** este operational la Tim-S Plus sub controlul softului versiunilor ZX-Spectrum incepind cu **48K** si mergind pina la **+2**. Utilizarea acestui mod de lucru se bazeaza pe existenta - in cadrul paginii pi1 a blocului BR2 de memorie RAM - a unor rutine care permit o extensie de dialect BASIC, asa fel incit utilizatorul sa poata comanda prin program urmatoarele resurse hard ale calculatorului: unitatile de discuri flexibile, reteaua locala omogena si legatura seriala RS232 proprie. Concret, noile rutine contin o extensie de analizor sintactic si o extensie de interpretor BASIC.

Extensia de limbaj se refera la urmatoarele comenzi (instructiuni): FORMAT, OPEN#, CAT, ERASE, MOVE, SAVE, LOAD, VERIFY, MERGE, CLS si CLEAR, care vor fi analizate in continuare.

5.1 FORMAT

Sintaxa instructiunii este:

FORMAT 12345

semnificatia fiecarui cimp marcat printr-o cifra fiind urmatoarea:

1 - o litera intre ghilimele semnificind subsistemul la care se refera instructiunea;
2 - un separator virgula (,) sau semicoloana (;);
3 - o expresie numerica;
4 - un separator (,) sau (;);
5 - o expresie ce da un sir (maxim 10 caractere).

Cimpurile **4** si **5** nu sint intodeauna obligatorii.

Sint 3 subsisteme programabile direct prin Interface I de pe Tim-S Plus: unitatile de discuri (litera M sau m), reteaua locala (N sau n) si legatura seriala (T, t, B sau b).

5.1.1 FORMAT & unitatile de disc

Cu "m" sau "M" in cimpul 1 de sintaxa se apeleaza unitatile de disc. Efectul este formatarea dischetei din unitatea de disc logica (1 sau 2) specificata de cimpul **3**. Prin formatare orice informatie anterior memorata pe disc se pierde. Discheta este initializata cu numele prezent in cimpul 5.

Exemple: - se formateaza un disc pe unitatea 1

1) **FORMAT "M"; 1; "Cuzco"**

discul va avea numele **Cuzco vizibil** la executia instructiunii CAT.

2) **FORMAT A$, B; C$**

A$ este necesar sa contina "M" sau "m", expresia B are valoarea 1 sau 2 si C$ este un sir alfanumeric de lungime intre 1 si 10.

Daca o expresie sau variabila folosita este afara din domeniul permis ca valoare se va genera mesaj de eroare.

5.1.2 FORMAT & reteaua

Daca pe cimpul 1 este "N" sau "n" atunci se va inscrie in variabilele de sistem numarul din cimpul 3 ca fiind numarul statiei pe retea (implicit acesta este 1). Cimpurile 4 si 5 nu sint neaparat necesare (se ignora).

Exemple:

1) **FORMAT "N", 2**

am definit calculatorul drept statia 2 din retea.

2) **FORMAT A$;B**

variabila B contine o valoare intre 1 si 64 iar A$ este "N" sau "n".

Pentru expresii in afara domeniului se genereaza mesaje de eroare.

5.1.3 FORMAT & RS232

Daca in cimpul 1 este "T", "t", "B" sau "b" atunci comanda se adreseaza legaturii seriale RS232 si permite specificarea vitezei de transmisie in bauds pentru lucru cu RS232. Viteza implicita este 9600 bauds. Valorile permise pentru cimpul 3 la specificarea vitezei sint: 50, 110, 300, 600, 1200, 2400, 4800, 9600 si 19200.

Exemple:

1) **FORMAT "T"; 1200**

definim viteza de 1200 bauds pentru interfata RS232.

2) **FORMAT A$,B**

variabila A$ poate fi "T", "t", "B" sau "b" si variabila B ia una din valorile permise pentru viteza de transfer. Daca B>65535 sau A$ contine caractere nepermise se va genera mesaj de eroare explicit.

Daca B este diferit de valorile specificate pentru viteza de transmisie, nu se genereaza mesajul de eroare dar se ia valoarea specificata in tabelul imediat mai mica decit valoarea data in cimpul 3 al comenzii.

5.2 OPEN#

Sintaxa instructiunii este:

OPEN# 1234567

cimpurile marcate prin cifra avind urmatoarele semnificatii:

1 - un numar de canal (normal intre 4 si 15);
2 - un separator (,) sau (;);

3.- o litera intre ghilimele ce semnifica subsistemul;
4 - un separator (,) sau (;);
5 - o expresie numerica;
6 - un separator (,) sau (;);
7 - o expresie ce da un sir (maxim 10 caractere).

Ultimele patru cimpuri nu sint intodeauna obligatorii.

Rolul instructiunii OPEN# este de a asocia un subsistem cu o anumita zona de memorie tampon necesara, pentru a putea comanda subsistemul cu: PRINT#n,INPUT#n si INKEY$#n, unde n este un numar de canal asociat subsistemului.

Implicit canalele 0 si 1 sint asociate tastaturii, canalul 2 iesirii spre fisierul de afisare pe TV iar canalul 3 asociat iesirii la imprimanta. Aceasta programare implicita a canalelor poate fi modificata cu ajutorul instructiuni OPEN#.

Deschiderea (OPEN) unui canal pentru un subsistem implica doua etape: intii canalul de date necesar unui subsistem (buffer) este plasat in zona de informatii despre canale apoi diferenta intre adresa de baza a canalului de date si variabila de sistem CHANS e plasata in zona de date referitoare la zona tampon (buffer).

OBS.: In cazul canalelor "m", "M", "b" si "B" cimpul 2 necesita strict separatorul ";".

5,2.1 OPEN# & unitatile de disc

Daca pe cimpul 3 este "M" sau "m" atunci folosind OPEN# se poate asocia un canal unui anumit fisier creat pe una din unitatile de disc.

Exemple:

1) **OPEN#4;"M",1,"TAMBUR"**

se asociaza fisierului numit "TAMBUR" creat pe unitatea 1, canalul 4.

2) **OPEN#A;B$;C,D$(2)**

unde A este in limitele 0-15. B$ contine "M" sau "m", C are valoarea 1 sau 2 iar D$(2) este un sir alfanumeric de maxim 10 caractere.

Dupa executia unei comenzi OPEN# de felul acesta fisierul specificat va fi deschis pentru scriere daca este un fisier nou sau va fi deschis pentru citire daca este un fisier deja existent. La Tim-S Plus nu este posibil sa se adauge ceva unui fisier deja existent (si inchis).

Daca un canal deja deschis este deschis din nou se genereaza mesaj de eroare (exceptind canalele 0-3). Mesaj de eroare se genereaza si daca o expresie este afara din domeniul permis sau daca nu mai este disponibila memorie pentru a permite marirea zonei de canale cu 595 octeti.

Folosirea lui OPEN# pentru a asocia un canal unui fisier anumit nu creeaza fisierul in sine. Crearea se face numai in cazul in care se scriu mai mult de 512 octeti spre fisier sau daca se utilizeaza CLOSE# pentru a inchide canalul deschis; procesul de creare presupune inregistrarea pe disc a unor sectoare de informatie, care intra in componenta fisierului.

5.2.2 OPEN# & retea

Daca in cimpul 3 este "N" sau "n" aceasta permite utilizatorului sa asocieze un canal cu receptia sau cu transmisia datelor prin retea spre sau de la un alt calculator Tim-S Plus.

Exemple:

1) OPEN#4;"N",44

asociaza canalul 4 cu calculatorul, care este pe post de statie cu numarul 44.

2) OPEN#A,B$,C

e necesar 0<A<=15 ,B$="N" sau "n" si 0<C<64.

Daca s-a deschis deja canalul, OPEN# va genera mesaj de eroare (exceptie canalele 0...3).

Expresia care da valoare afara din domeniu si memorie insuficienta (pentru a permite extinderea memoriei tampon cu 276 octeti) va genera mesaje de eroare.

Asocierea canalului pentru retea nu specifica tipul: receptie sau transmisie. Datele receptionate pot fi doar citite iar datele existente spre a fi transmise pot fi doar transmise.

5.2.3 OPEN# & RS232

Daca pe cimpul 3 este "T", "t", "B" sau "b" aceasta asociaza un anumit canal legaturii seriale RS232, fie pentru transmisie, fie pentru receptie date. Rata de transmisie este neafectata de OPEN#.

Exemple:

1) OPEN #4; "T"

canalul 4 este asociat RS232 iar ceea ce circula prin acest canal este interpretat ca text (adica date codificate ASCII).

2) OPEN# A; B$

A este cuprins intre 0-15. Daca B$="B" sau "b", atunci datele care circula vor fi interpretate ca octeti binari.

Daca o expresie este afara din domeniu atunci se genereaza mesaj de eroare. Memoria tampon utilizata de RS232 este de 11 octeti.

5.3 CAT

Sintaxa instructiunii este:

CAT 1234

cimpurile macate prin cifra avind urmatoarele semnificatii:

1 - simbolul "#";
2 - numar de canal;
3 - separator (,) sau (;);
4 - numarul unitatii de disc (1 sau 2).

Primele trei cimpuri sînt optionale, ultimul cimp este obligatoriu.

Instructiunea listeaza pe display numele dischetei urmat de numele fisierelor de pe discheta (din unitatea specificata in cimpul 4, unul sub altul, exceptind fisierele protejate (al caror nume incepe cu CHR$0). La sfirsit se afiseaza si numarul de Ko ramasi liberi pe discheta.

Fara cimpurile optionale informatia se va scrie in canalul 2, adica pe display.

Exemple:

1) **CAT 1**

lista fisierelor existente pe discheta din unitatea de disc 1 este scrisa in canalul 2.

2) **CAT@A, B**

cu A cuprins intre 0 si 15 si B egal cu 1 sau 2. Se va genera mesaj de eroare daca:

- o expresie este afara din domeniu;
- canalul ce se utilizeaza nu este deschis;
- nu exista discheta in unitatea specificata.

Daca se comanda CAT fara parametri aceasta este o eroare care nu va fi detectate la analiza sintactica, ci doar in executie.

5.4 ERASE

Sintaxa:

ERASE 12345

1 - "M" sau "m";
2 - separator (,) sau (;);
3 - numarul unitatii logice este 1 sau 2;
4 - separator (,) sau (;);
5 - o expresie ce da un sir alfanumeric (nume de fisier de pe disc).

Instructiunea sterge blocurile de date ce apartin fisierului cu numele definit de cimpul 5 de pe unitatea data de cimpul 3. Sectoarele "sterse" sint gata a fi reutilizate de alte fisiere.

Exemple:

1) **ERASE "M"; 1; "Aristotel"**

fisierul cu numele **Aristotel** este sters de pe discheta din unitatea 1.

2) **ERASE A$, B, C$**

unde A$="M" sau "m", B=1 sau 2 si C$ contine un nume de fisier.

Se genereaza mesaj de eroare daca o expresie este afara din domeniu sau daca nu este discheta in unitatea specificata.

5.5 MOVE

Sintaxa:

MOVE 12345

1 - simbol "#"
2 - numar de canal
3 - cuvintul BASIC: TO
4 - simbol "#"
5 - numar de canal

Cimpurile 2 si 5 pot fi scrise si explicit. Scrierea explicita presupune completarea unui sir de forma

abcde

in locul fiecarui nume de canal, dupa urmatoarea logica:
a - o litera ce specifica subsistemele;
b - separator (,) sau (;);
c - expresie numerica;
d - separator (,) sau (;);
e - este expresie ce da un sir alfanumeric (nume de fisier).

Primul cimp este obligatoriu, urmatoarele patru fiind (pe grupe de cite doua) optionale.

Instructiunea MOVE permite utilizatorului sa receptioneze date de la un canal sursa si sa transmita date spre un canal destinatie.

Exemple:

1) **MOVE #4 TO #5**

datele receptionate de canalul 4 sint transmise spre canalul 5

2) **MOVE "N",20 TO "M"; 1; "DATERET"**

octetii receptionati de pe retea de la statia 20 vor fi stocati in fisierul cu numele **DATERET** de pe unitatea logica 1.

Se vor genera mesaje de eroare daca:
- expresiile sint in afara domeniului;
- canalul nu este deschis;
- nu este disc in unitatea specificata;
- fisierul sursa nu este gasit;
- fisierul destinatie este deja existent.

Comanda MOVE foloseste doar pentru fisiere de lungime finita, adica fisiere terminate cu separator de sfirsit de fisier. Lucrul cu fisiere fara separator de sfirsit de fisier, cu fisiere continind programe BASIC sau tablouri genereaza erori.

5.6 SAVE, LOAD, VERIFY, MERGE

Aceste 4 instructiuni sint folosite in Tim-S Plus pentru a transfera pe unitatile de disc programe de pe benzile magnetice - concepute pentru modul de lucru Spectrum 48K -, unde urmeaza a fi utilizate, dar cu viteza de lucru mai mare.

Sintaxa :

abcdefghijk

unde:

a - SAVE, LOAD, VERIFY sau MERGE;
b - simbolul #;
c - o litera ce indica subsistemul utilizat;
daca este necesar:
d - separator (,) sau (;);
e - o expresie numerica;
daca este necesar:
f - separator (,) sau (;);
g - o expresie ce da un sir alfanumeric (nume);
si daca este necesar (BASIC):
h - LINE;
i - expresie numerica (program cu auto-run);
sau
h - DATA;
i - o variabila tablou (tablouri BASIC);
sau
h - CODE (pentru blocuri e date);
i - expresie numerica (adresa de start) - optionala;
j - separator (,);
k - expresie numerica (lungime bloc) - optionala;
sau
h - SCREEN$ (pentru imagini).

Simbolul "#" din cimpul b provoaca comutarea programelor de pe 48 BASIC pe extensia BASIC Interface 1. Deosebirea dintre forma cimpului argument (zona b-k) prezentata mai inainte si forma cimpului argument al acelorasi (4) functii in modul de lucru 48 BASIC este ca la prima nu mai sint intodeauna necesare nume. Exceptie face lucrul cu unitatile de discuri, in acest caz fiind necesare nume de fisiere.

Exemple:

1) SAVE # "m", 1, "FISIER"

programul BASIC din Tim-S Plus este salvat in fisierul cu numele cu FISIER pe unitatea 1.

2) SAVE #"n";2

programul BASIC din Tim-S Plus este trimis in retea spre statia spre 2.

3) SAVE # "b" SCREEN$

fisierul de afisare curent este trimis prin RS232 (SCREEN$ cu "T" va da eroare in executie deoarece este ilogic).

4) LOAD #"m", 2, "PROG"

fisierul cu numele PROG este incarcat de pe unitatea 2 si daca exista si este intradevar program BASIC va putea fi executat.

5) LOAD # "n"; 28

se incarca in calculator un program BASIC asteptat de la statia 1a

6) **LOAD * "b"**

se incarca in calculator un program BASIC de pe RS232.

E posibil ca datele receptionate de la retea sau de pe RS232 sa fie recunoscute fals ca programe BASIC, caz in care se va bloca in executie calculatorul fara a da mesaj de eroare:

Wrong file type.

7) **VERIFY * "M", 1; "PROG"**

compara programul din fisierul cu numele **PROG** al unitatii logice 1 cu programul aflat in memoria calculatorului.

8) **VERIFY * "N"; 33**

compara programul primit de la statia 33, prin retea, cu cel din memorie.

9) **VERIFY * "B" CODE 32000, 256**

compara 256 de octeti preluati de la locatii de memorie consecutive, incepind cu adresa 32000, cu informatia primita prin RS232.

10) **MERGE * "m", 1,"Casiopeea"**

programul BASIC aflat in fisierul cu numele **Casiopeea** pe unitatea 1 este alipit (MERGE obisnuit) la programul BASIC existent in memoria calculatorului.

11) **MERGE * "n", 53**

programul BASIC receptionat prin retea de la statia 53 este alipit la programul existent in memorie.

12) **MERGE * "b"**

programul BASIC preluat de la RS232 este alipit la programul din memorie.

De retinut ca MERGE se aplica doar la programe si variabile BASIC, altfel apare mesaj de eroare:

MERGE error.

Programele BASIC cu auto-run nu accepta MERGE.

3.7 CLS# & CLEAR#

CLS# provoaca actiunile efectuate de CLS simplu, in plus defineste urmatoarele optiuni de lucru cu ecranul video: INK 0, PAPER 7, BORDER 7, INVERSE 0, BRIGHT 0, OVER 0, FLASH 0.

CLEAR# sterge toate zonele de memoria tampon alocate canalelor de date. Toate canalele sint inchise (CLOSE). Canalele 0-3 primesc valori implicite, celelalte primesc valori nule. In contrast cu CLOSE#, CLEAR# sterge orice memorie tampon netransmisa si reduce zona de memorie tampon la configuratia

minima.

Exemple:

1) **CLS# : PAPER2 : CLS**

genereaza fond rosu pe TV.

2) **CLS#:SAVE *"n",0:PAPER2:CLS**

nu va genera fond rosu.

3) **CLS #:SAVE *"n",0:PRINT;:PAPER2:CLS**

va genera fond rosu deoarece comanda PRINT va reselecta canalul de afisare pe TV.

5.8 Reteaua locala omogena

Calculatorul Tim-S Plus se poate cupla intr-o retea locala omogena. Reteaua omogena este formata din calculatoare ale familiei Spectrum care prezinta compatibilitate **Interface I**, din punct de vedere al protocolului (adoptat la aceste calculatoare) de dialog prin intermediul unui singur fir purtator de informatie. Reteaua locala se refera la distributia statiilor care participa la retea. O retea de microcalculatoare permite cuplarea mai multor microcalculatoare intr- o anumita structura in scopul stabilirii unui canal de comunicatie comun mai multor statii, de mare viteza, pentru transferul datelor intre statii. Rata de transfer a datelor este de circa 5 Ko pe secunda, superioara legaturii seriale RS232.

Un microcalculator participant la retea este o statie a retelei care are un numar de statie. Este necesar ca statiile sa aibe numere diferite. Numarul unei statii poate fi schimbat la comanda, altfel se pastreaza valoarea implicita de initializare a numarului statiei, care este 1. Statie sursa este aceea care transmite un mesaj, iar statie destinatie este aceea care receptioneaza un mesaj.

In descrierea retelei vom denumi simbolic: **A** - (apropiata) statia la care lucreaza utilizatorul si **D** - (departata) statia cu care comunica utilizatorul. Comunicarea este posibila in sensul **A** spre **D** sau in sensul **D** spre **A**.

Reteaua este prezenta fizic prin 2 cuple pe carcasa, extensia Tim-S Plus si cablurile coaxiale (masa + semnal) care fac legatura intre statii. Calculatoarele care participa la retea este necesar sa fie cuplate intre ele fizic cu cabluri coaxiale. De exemplu, pentru o retea formata din 3 calculatoare - statia 1 (de tip **A**), statia 2 (de tip **D**) si statia 3 (de tip **D**) - , sint necesare 2 cabluri care pornesc de la 1 si merg la 2 respectiv la 3. (Utilizatorii retelei din 2 si 3 vor avea un punct de vedere similar cu al lui 1 in ceea ce priveste reteaua).

Din punct de vedere al comunicatiei ce se efectueaza, reteaua poate fi in repaus (cind nu se lucreaza pe ea) sau poate fi functionala (cind se lucreaza pe ea). Daca este functionala, reteaua poate fi in starea activa (cind se transmit semnale in acel moment) sau stare inactiva (cind exista o pauza intre trenurile de semnale transmise). Statia sursa are datoria sa faca reteaua functionala, pe cind statia destinatie are rolul de a citi reteaua si eventual, daca este necesar, de a raspunde. Protocolul de comunicatie intre statii corespunde unui format

strict definit.

In aceasta descriere vom veni in ajutorul utilizatorului care doreste sa programeze lucrul in retea folosind limbajul BASIC.

5.8.1 Schimbarea numarului statiei

Pentru un calculator alimentat si pornit, variabilele de sistem pentru modul de lucru in retea nu sint inserate in memorie. Este necesara folosirea unei instructiuni specifice extensiei de BASIC pentru a realiza inserarea. Dupa insertia variabilelor de extensie sistem statia primeste implicit numarul 1. Numarul statiei este retinut in variabila de extensie sistem numita simbolic NTSTAT de la locatia 23749. Utilizatorul poate schimba acest numar utilizind comanda:

FORMAT "N", numar statie

sau

POKE 23749, numar statie.

Numarul de statie trebuie sa fie cuprins intre 1 si 64. Restrictia nu se aplica pentru utilizatori care folosesc reteaua programind in cod Z80.

Numarul curent al statiei este copiat in zona de memorie tampon utilizata de canalele create pentru retea. Este posibil deci a se folosi canale de retea diferite pentru receptie si pentru emisie cu statiile D.

5.8.2 SAVE pentru retea

Folosind SAVE se pot transmite de la A catre D urmatoarele tipuri de informatie:

- programe BASIC;
- variabile tablou BASIC;
- blocuri de date in cod reprezentind cod sau imagine.

Exemple:

1) **SAVE *"n",2**

se trimite spre statia D cu numarul 2 programul BASIC din memoria statiei A.

2) **SAVE *"N"; 3 LINE 10**

se trimite spre statia D cu numarul 3 programul BASIC din memoria statiei A, programul fiind de tip auto-run.

3) **SAVE *"n", 0 SCREEN$**

se trimite spre statia D cu numarul 0 (orice numar de statie ce asteapta in receptie colectiva) imaginea TV curenta din statia A.

4) **SAVE *"N", 1 DATA A()**

trimite spre statia D cu numarul 1 tabloul A() din programul statiei A.

5) **SAVE *"n"; 10 CODE 30000,1000**

se trimit spre statia D cu numarul 10, 1000 de octeti incepind de la adresa 30000 a memoriei statiei A.

Realizarea comunicatiei cu SAVE de la A spre D este posibila doar daca si statia D asteapta sa primeasca acelasi tip de date. Daca in loc de numarul statiei D se pune 0 aceasta inseamna ca A transmite spre toate statiile (colectiv), iar toate statiile D care se asteapta la o emisie de tip colectiv vor receptiona datele transmise.

Realizarea unei legaturi de comunicatie de tip dedicat (spre o anumita statie D) implica un protocol strict de raspuns de la statia ce receptioneaza. La legaturile de comunicatie de tip colectiv statia ce emite nu asteapta raspuns. Datele trimise cu emisie colectiva sint transmise cu o viteza de circa 4 ori mai mica decit normal.

5.8.3 LOAD, MERGE si VERIFY pentru retea

Aceleasi categorii de date ca la SAVE pot fi receptionate de pe retea (de la statia D) utilizind LOAD, MERGE sau VERIFY.

Exemple:

1) **LOAD *"n"; 2 DATA A()**

se receptioneaza de la statia D cu numarul 2 un tablou numeric in variabila A().

2) **VERIFY *"N", 0**

programul BASIC din statia A este verificat prin comparatie cu programul BASIC emis colectiv de o statie D.

3) **MERGE *"N", 1**

programul primit de la statia D cu numarul 1 va realiza MERGE cu programul statiei A.

Este necesar ca statiile D sa fie pregatite sa transmita blocul de informatie asteptat de A.

Protocolul de comunicatie presupune minim 2 statii pe care le numerotam cu 1 si 2. Presupunem ca 1 vrea sa transmita. Este necesar sa dam numarul 1 statiei apoi sa comandam transmisia spre statia 2. Deci executam pe calculatorul statiei 1 instructiile:

```
FORMAT   "N",1
SAVE *"N", 2.
```

Pentru ca 2 sa receptioneze este necesar sa executam comenzi pe calculatorul care este statia 2, prin care numarul acestei statii este pus 2, iar numarul statiei de la care facem receptia este 1:

```
FORMAT "N",2
LOAD * "N";1
```

Daca una din statii ajunge la LOAD/SAVE inainte de a ajunge

cealalta la SAVE/LOAD, statia care ajunge prima trece in starea de asteptare. Instructiunile LOAD si SAVE din cele 2 programe diferite, de pe cele 2 calculatoare diferite, vor ajunge dupa un timp sa se sincronizeze (comunicatie dedicata). Dupa acest moment de sincronizare programul BASIC din statia 1 incepe sa fie transmis spre statia 2. Se face in prelabil o divizare a programului in blocuri de 255 de octeti, fiecare bloc nou obtinut fiind apoi transmis pe linie.

Pe perioada cit statiile se asteapta una pe cealalta, bordura imaginii video va face schimbarea de culoare intre culoarea initiala si culoarea din variabila IOBORD de la locatia 23750. In timpul comunicatiei pe BORDER vor apare dungi asemanatoare cu cele de la lucrul cu casetofonul.

Instructiile SAVE, LOAD, VERIFY, MERGE nu solicita de la utilizator deschiderea sau inchiderea unui canal, insa un canal pentru retea se deschide implicit (din interpretor). Acest canal este dat disponibil pentru alte scopuri de indata ce nu se mai lucreaza cu reteaua.

5.8.4 Transmisia de date prin retea

In acest scop sint folosite 3 comenzi BASIC: OPEN#,PRINT# si CLOSE#. Prin date se intelege in acest caz: o colectie de expresii PRINT-abile (tiparibile) si separate de caracterul CR (carr. return) sau o colectie de caractere. Se delimiteaza 3 etape:

A)Este necesar sa fie creat un canal de retea in scopul comunicatiei de la **A** spre **D**, aceasta realizindu-se cu instructiunea OPEN#.

Exemple:

1) **OPEN#4; "N"; 18**

asociaza canalul 4 pentru retea cind **A** comunica cu **D**=18.

2) **OPEN#5; "n", 0**

asociaza canalul 5 cu un nou canal de retea, care va lucra cu date in comunicatie colectiva.

Zona memoriei tampon alocata canalului de retea este de 255 octeti. Transmisia spre **D** se va face doar cind in aceasta zona sint colectati 255 de octeti.

B)Canalul de retea existent trebuie umplut cu datele necesare folosind o instructie PRINT#, cu numar de canal corespunzător.

Daca datele vor fi considerate ca expresii (la receptie), este necesar sa se aibe grija pentru asigurarea plasarii corespunzatoare a codului CR (ca si cum s-ar apasa ENTER pentru date preluate cu INPUT simplu).

Exemple:

1) **PRINT#4; 1**

in zona tampon alocata canalului 4 se vor incarca codul "1" si codul CR.

2) **PRINT#5; "UNU"**

in zona tampon alocata canalului 5 se vor incarca codurile "U", "N", "U" si codul CR.

3) **PRINT#6; A; B$**

in zona tampon alocata canalului 6 se vor incarca intii caracterele ce formeaza variabila A si codul CR, apoi caracterele ce formeaza variabila B$ si codul CR.

Un canal de retea permite incarcarea a 255 de octeti in zona tampon de memorie corespunzatoare. Cind se scrie al 256-lea octet cei 255 de octeti existenti sint transmisi pe retea iar ultimul (al 256-lea) devine primul in zona tampon (eliberata prin transmisie).

C)Este necesar sa se inchida canalul deschis utilizind instructiuna CLOSE#. Actiunea comenzii CLOSE# presupune urmatoarele faze: intii se transmite orice zona tampon corespunzatoare unui canal deschis pentru retea spre **D**, apoi elibereaza canalul de retea si inscrie 0 in locatia tampon respectiva (doar canalele 0...3 pastreaza valorile originale).

Exemplu:

1) **CLOSE#4**

inchide canalul 4.

5.8.5 Receptia de date din retea

Exista 4 instructiuni care permit receptia de date din retea: OPEN#, INPUT#, INKEY$#, CLOSE#. La fel ca la transmisie, prin date se poate intelege expresii (necesare a fi separate cu codul CR) sau o colectie de caractere. Sint necesare 3 etape:

A)Este necesar sa fie deschis un canal.

Exemple:

1) **OPEN#7, "N"; 1**

asociaza canalul 7 pentru un nou canal de retea, dedicat comunicatiei cu statia 1.

2) **OPEN#6; "n", 0**

asociaza canalul 6 pentru receptia din retea a datelor emise colectiv.

B)Se utilizeaza INPUT# si/sau INKEY$#. INKEY$# se foloseste cind se citeste de la tastatura doar un singur caracter.

Exemple:

1) **INPUT#7; A**

octetii receptionati, terminati cu un cod CR, sint atribuiti variabilei A (ceea ce se receptioneaza poate fi o expresie dar este necesar sa fie un argument valid pentru functia VAL).

2) **INPUT#7; A$**

octetii receptionati, terminati cu un cod CR, sint considerati ca un sir de caractere de lungime finita si sint atribuiti variabilei A$.

3) **INKEY$#7**

sirul dat de functie va fi urmatorul caracter receptionat.

In toate cazurile zona de memorie alocata canalului de retea este examinata pentru a determina daca exista date receptionate; daca nu exista se va prelua un nou bloc de date de la statia corespondenta. Daca ultimul bloc de date fost marcat ca fiind sfirsit de fisier atunci nu se mai preia alt bloc si se da mesaj de eroare "**end of file**" (sfirsit de fisier).

In cazul comunicatiei dedicate functia INKEY$# va da urmatorul octet receptionat (ca un sir de lungime 1). Daca D nu asteapta de la A o cerere de trimitere a urmatorului bloc de date, functia INKEY$# va da sirul nul (de cod 0). Aceasta serveste utilizatorului statiei A pentru a colecta date de la oricare statie pregatita sa transmita.

C)Este necesar sa se inchida canalul deschis anterior. Orice data receptionata dar nefolosita se pierde.

Exemplu:

CLOSE#7

elibereaza canalul 7 de retea si umple cu 0 octetii zonei tampon aferente acestui canal.

Protocolul de comunicatie cu date prin retea necesita minim 2 statii. De exemplu: fie statia 1 care vrea sa emita si statia 2 care receptioneaza date. In statia 1 va fi rulat programul:

```
10 FORMAT"N", 1
20 OPEN#5; "N", 2
30 FOR A=1 TO 240
40 PRINT#5; A
50 NEXT A
60 CLOSE#5
```

Programul da numar statiei (10), deschide canalul 5 de tip retea dedicata statiei 2 (20),transmite 240 de valori de la 1 la 240 (30,40,50), apoi inchide sirul 5, eliberind zona de memorie ocupata. Abia la executia liniei 60 se incheie transmisia propriuzisa prin transmisia unui bloc de date continind ultimele valori date ...239,240.

In statia 2, care receptioneaza, este necesar sa fie rulat programul:

```
10 FORMAT "N"; 2
20 OPEN#12; "N"; 1
30 INPUT#12; X
40 PRINT X
50 GOTO 30
```

Programul da numarul 2 pentru statie (10), deschide canalul 12 de tip retea dedicata statiei 1 (20), preia o expresie din zona de memorie tampon alocata canalului 12 (daca exista ceva receptionat acolo) si o atribuie variabilei X (30), afiseaza valoarea variabilei X pe ecran (40) si cicleaza la linia 30 (50).

Si in acest caz statiile sursa si destinatie se vor sincroniza prin introducerea unor perioade de asteptare (atunci cind este necesar).

Structura unei zone de memorie tampon (buffer) alocata unui canal de tip retea este:

Octet	Continut
0-1	Adresa #0008
2-3	Adresa #0008
4	Codul "N" (+#80 daca este canal folosit de SAVE, LOAD, VERIFY sau MERGE)
5-6	Adresa rutinei de transmisie
7-8	Adresa rutinei de receptie
9-10	Numarul #0114 (276 octeti in zona tampon)
	Urmatorii 8 octeti contin antetul:
11	Numarul statiei **D**
12	Numarul statiei **A**
13-14	Numarul blocului de date (>=0 si <65536)
15	Daca este cod "1" atunci marcheaza sfirsit de fisier; altfel este "0"
16	Lungimea zonei tampon necesara; pentru receptie =0; la emisie <256
17	Suma de control pentru date
18	Suma de control pentru anteriorii 7 octeti
	Urmatorii 2 octeti sint folositi in receptie
19	Pozitia ultimului caracter luat din zona tampon alocata canalului de retea
20	Numarul de octeti ce pot fi cititi din zona
	Urmatorii 255 octeti formeaza zona tampon de memorie alocata unui canal de retea pentru emisie sau receptie
21-275	255 octeti de date

Antetul este necesar stabilirii legaturii de comunicatie, dupa ce in prealabil retsaua a fost ocupata de statia **A** printr-un procedeu de sesizare a starii de repaus (CSMA). Dupa transmiterea antetului si primirea validarii receptiei de la statia **D**, transmisia continua cu blocuri de date de 255 octeti si validare a receptiei pina se termina sesiunea de lucru pe retea intre **A** si **D**.

Atentie:

1)Un bloc de date este marcat ca fiind sfirsit de fisier doar la inchiderea canalului de retea cu CLOSE# sau la executia unei instructiuni SAVE*.

2)O statie care executa in program o instructiune retea de tip LOAD, VERIFY, MERGE, INPUT va astepta oricit pina va receptiona corect ceea ce asteapta. In executia unui INKEY$# statia va astepta doar un anumit timp (scurt) dupa care va da ca valoare a functiei sirul nul.

3)La receptia blocurilor emise spre colectiv se poate crea confuzie deoarece intre blocurile de date pot apare durate destul de mari de stare inactiva, durate ce pot fi interpretate de alte statii ca stari de repaus. Aceste ultime statii vor incerca sa preia lucru pe retea emitind si eventual alterind ceea ce statia

care emite spre colectiv vrea sa transmita. Deci, atentie la transmisiile de durata in modul emisie colectiva.

4)La receptie, daca statia foloseste continutul zonei tampon alocata canalului de retea (cu INPUT# sau INKEY$#) atunci cind va ajunge sa foloseasca ultimul octet receptionat din zona, o citire de nou octet va genera (daca acest bloc de date este marcat ca sfirsit de fisier) un mesaj de eroare sau o cerere de nou bloc de date de la emitent (daca blocul nu este marcat ca sfirsit de fisier).

5)Sint 4 categorii de comunicatii pe retea:

- emisie dedicata (spre cineva);
- receptie dedicata (de la cineva);
- emisie colectiva (spre oricine);
- receptie colectiva (de la oricine);

6)Uneori este necesar ca utilizatorii retelei locale omogene sa mai utilizeze un mijloc de comunicare (exemplu: telefon sau vizual) pentru a-si sincroniza activitatile.

5.9 Legatura seriala RS232

Un al III-lea subsistem (optiune) oferit de extensia **Interface I** este legatura seriala RS 232, utilizabila prin cupla seriala.

RS232 asigura in principiu legatura seriala spre un sistem cu o interfata RS232, in protocolul definit. De exemplu o imprimanta sau un alt calculator.

RS232 asigura receptia sau transmisia datelor seriale pe un cablu fizic de 5 fire: 2 la receptie, 2 la emisie si un fir de masa (GND). Octetii sint transmisi unul dupa altul iar un octet anumit se transmite doar daca cealalta statie a semnalizat ca este pregatita sa-l recepţioneze.

Protocolul de emisie foloseste 2 fire: LDT si LCTS. Protocolul de receptie foloseste alte 2 fire: LDR si LDTR (vezi fig.8,b). Deci firele LDT, LCTS, LDR si LDTR de la o statie vor fi conectate cu firele LDR, LDTR, LDT, respectiv LCTS de la cealalta statie.

Octetul circula impachetat astfel: un bit de start, 8 biti ai octetului data si 2 biti de stop (fara bit de paritate). Total 11 biti.

Viteza de transmisie a datelor pe legatura seriala e luata implicit 9600 de unitati numite baud. Pentru a vedea citi octeti se transmit pe secunda se imparte aceasta viteza la 11.

Din cauza protocolului prin care emitatorul poate emite doar daca receptorul poate prelua octet, apar pauze diferite care fac viteza reala de transmisie mult mai mica decit aceea rezultata din calculul anterior.

5.9.1 Alegerea vitezei de transmisie

Viteza de transmisie se poate schimba de la valoare data implicit folosind instructiunea FORMAT.

Exemplu:

```
FORMAT "b";110
```

se alege viteza de 110 baud pentru RS232.

Vitezele standard programabile sint: 50, 110, 300, 600, 1200, 2400, 4800, 9600 si 19200. Daca valoarea vitezei nu este

aceiasi cu una din valorile standard, atunci valoarea adoptata se va trunchia la valoarea din tabel imediat mai mica; daca este <50 atunci se adopta valoarea 50; daca este >19200 si <65535 se adopta valoarea 19200.

Exemple:

1) **FORMAT "b"; 0**

pune viteza 50 baud.

2) **FORMAT "b"; 1500**

pune viteza 1200 baud.

In urma executiei comenzii FORMAT variabila de extensie sistem BAUD este actualizata cu valoarea rezultata. Utilizind instructia POKE sau prin programe in cod Z80 este posibil sa se aleaga orice viteza de transmisie nestandard, dupa formula:

(3500000/(26*viteza))-2

5.9.2 Comanda SAVE la RS232

Programele si variabilele tablou BASIC si blocurile de date (cod sau imagine) pot fi transmise prin RS232 spre alta statie, utilizind SAVE.

Exemple:

1) **SAVE *"b"**

programul si variabilele BASIC sint transmise prin RS232.

2) **SAVE *"b" SCREEN$**

o copie a imagini TV este transmisa prin RS232.

3) **SAVE *"b" CODE 30000,128**

128 octeti se preiau de la adrese consecutive din memorie - incepind cu adresa 30000 - si transmisi prin RS232.

4) **SAVE *"b" DATA B$( )**

variabila B$( ) este transmisa prin RS232.

5.9.3 Comenzile LOAD, MERGE si VERIFY

Acelasi categorii de date ca la SAVE pot fi receptionate de pe RS232 utilizind comenzile LOAD, MERGE si VERIFY.

Exemple:

1) **LOAD *"b"**

un program BASIC este incarcat de la RS232.

2) **MERGE *"b"**

un program BASIC luat prin RS232 este compus MERGE cu programul existent in calculator.

3) **VERIFY *"b" DATA 32000,100**

informatia unui bloc de 100 octeti luat de la RS232 este comparat cu informatia blocului de 100 octeti care incepe de la adresa 32000 din memoria calculatorului.

In toate cazurile cind folosim SAVE, LOAD, MERGE sau VERIFY si identificatorul de subsistem "B" sau "b" prin RS232 se vehiculeaza octeti in plaja de valori 0-255.

Pe RS232 se transmit date doar dupa ce receptorul semnalizeaza emitorul ce este gata sa primeasca date. Pe timpul pauzelor sau a transferurilor de date bordura imaginii TV sufera schimbari de culoare intre culoarea curenta si cea memorata in variabila de sistem IOBORD (la fel ca la retea).

5.9.4 Comunicarea de date prin RS232

Ca si la reteaua omogena transmiterea de date implica deschiderea unui canal (OPEN), care va fi asociat pentru RS232, inainte de folosirea instructiunii PRINT#.

Exemplu:

OPEN#4; "b"

canalul 4, deschis pentru RS232.

Un canal deschis poate fi folosit fie pentru transmiterea fie pentru receptia de date. Aceasta este posibil deoarece nu mai este necesara o zona de memorie tampon de lucru alocata RS232, aceasta zona reducindu-se acum la 1 octet.

Dupa executia unui OPEN# in zona de memorie alocata informatiei despre canale se vor genera 11 octeti:

1-2	Adresa #0008
3-4	Adresa #0008
5	Codul "T" (+#80 pentru canalele implicite deschise la SAVE, LOAD, MERGE, VERIFY)
6-7	Adresa rutinei de transmisie RS232
8-9	Adresa rutinei de receptie RS232
10-11	Codul #0B (11 octeti)

Pentru transmisie se foloseste PRINT#.

Exemple:

1) **PRINT#4; A**

valoarea PRINT-abila (tiparibila) a variabilei A va fi transmisa prin RS232 urmata de un cod CR.

2) **PRINT#4; "UN SIR"**

caracterele textului **"UN SIR"** vor fi transmise prin RS232 urmate de un cod CR.

Pentru receptie se foloseste INPUT# sau INKEY#.

Exemple:

1) **INPUT#4; A$**

caracterele receptionate prin RS232, pina la receptia unui cod CR, sint atribuite sirului A$.

LET A$ = INKEY$#4

urmatorul caracter receptionat va fi atribuit variabilei A$. Daca nu s-a receptionat nici un caracter, CODE A$ va fi zero.

O comanda CLOSE# (exemplu CLOSE#4) va face sa fie transmis un caracter LF (linie noua, caracterizat prin cod 10), inainte de a se elibera zona ocupata de cei 11 octeti alocati canalului de RS232. Daca nu este necesar la receptie acest ultim LF, atunci se va suprima, de exemplu, prin executia unei comenzi CLEAR#, care va elibera zona de memorie alocata canalului de RS232.

5.9.5 Modul TEXT prin RS232

Subsistemul RS232 poate fi apelat si cu identificatorul "T" sau "t", aceasta permitind vehicularea informatiei in cod ASCII si a cuvintelor BASIC (token).

Vom putea lista un program BASIC dind comenzile:

```
OPEN#4; "T"
LIST#4
CLOSE#4
```

In modul text codurile caracterelor sint modificate astfel:

- codurile de control 0-31 sint ignorate exceptind LF si CR (codurile 10 si 13);
- codurile semigrafice sint transformate in simbolul "?" (de cod 63);
- cuvintele BASIC (token-uri) sint expandate.

De exemplu, se poate cupla un Tim-S Plus cu o extensie la o imprimanta care are RS232. Este posibil lucrul in 3 moduri:

A)folosind canale de tip "B" prin care se permite transferul de octeti cu valori intre 0 si 255; utilizatorul are deplina libertate in utilizarea codurilor de comanda a caracterelor ASCII si a celorlalte coduri;

B)folosind canalul de tip "B" pentru transmiterea codurilor de comanda si canalul de tip "T" pentru transmiterea de texte ASCII combinate cu cuvinte BASIC.

Exemplu:

```
10 OPEN#4; "T": OPEN#5; "B"
20 PRINT#5; CHR$27+ CODE "K"
30 PRINT#4; "Textul propriu-zis"
```

C)folosind modul text "T", utilizatorul va putea sa foloseasca doar codurile de commanda CR, LF si intreg setul de simboluri simple si compuse pentru imprimare.

OBS. In cazul canalelor "b", "B" delimitatorul acceptat dupa OPEN#n este ";".

5.10 Utilizarea unitatilor de disc sub Interface I

Prezenta optiunilor hard/soft pentru modul de lucru **Interface** I la Tim-S Plus permite - printre altele - si utilizarea unitatilor de disc. In acest scop softul de **Interface** I prezinta urmatoarele instructii:

1. FORMAT
2. CAT
3. ERASE
4. SAVE*
5. LOAD*
6. MERGE*
7. VERIFY*
8. OPEN#
9. CLOSE#
10. PRINT#
11. INPUT#
12. INKEY$#
13. MOVE
14. CLS#
15. CLEAR#

5.10.1 FORMAT

Formateaza o discheta tip Interface I. Sintaxa:

FORMAT "m", n, "nume disc"

unde

m - diferentiaza discheta de retea si de RS 232C (m pentru disc);
n - numar de unitate de disc, 1 sau 2;
nume disc - numele dischetei.

O unitate de disc e identificata cu unul din numerele 1 sau 2 si pune la dispozitia utilizatorului 128 K. Unei unitati cu care se lucreaza i se rezerva circa 0.5 K din memoria principala a calculatorului (zona buffer, dupa variabilele de sistem).

OBS. Sintactic este corect ; in loc de ,. Observatia este valabila si in continuare.

Exemplu: FORMAT "m",1;"Princeton"

Executia instructiei FORMAT atrage dupa sine pierderea informatiei inscrise pe discheta. Discheta este impartita in piste si sectoare (9 sectoare/pista, 512 octeti/sector, dubla densitate). Informatia pe care o contine un sector dupa formatare este constanta **#E5**.

5.10.2 CAT

Catalogheaza o discheta (tipareste in canalul curent numele fisierelor existente pe discheta). Sintaxa:

CAT [#ns],n

unde **ns** este numarul unui canal logic (implicit 2 - ecranul superior).

Exemplu: **CAT 1**

Executie: in canalul specificat in instructie (2) se tipareste:

- numele dischetei;
- numele fisierelor de pe discheta;
- spatiul neocupat, in K.

5.10.3 ERASE

Sterge un fisier. Sintaxa:

ERASE "m",n,"nume fisier"

unde **nume fisier** este numele fisierului existent care va fi sters de pe discheta.

Exemplu: **ERASE "m",1,"Andromeda"**

Executie: fisierul cu numele dat in instructie este sters de pe discheta din unitatea 1.

5.10.4 SAVE*

Salveaza un fisier de tip program. Sintaxa:

SAVE *"m",n,"nume fisier" urmat de: **CODE adr,lung**
SCREEN$
LINE linstart
DATA x(i,j)

unde:

adr - adresa de inceput a programului in cod masina de salvat;
lung - lungimea programului in cod masina;
linstart - linia la care se face RUN dupa incarcarea programului BASIC;
x - numele matricii numerice sau alfanumerice;
i,j - dimensiunea matricii (pot sa lipseasca).

Exemplu: **SAVE *"m",2;"Pleiadele"CODE 0,16304**

Executie: Este creat pe discheta din unitatea 2 fisierul cu numele dat in instructie si tipul specificat, care este (in exemplu) de tip **cod-masina.** Celelalte tipuri de fisier sint:

- program BASIC (optiunea absenta);
- ecran (SCREEN$);
- date (DATA).

OBS. Salvarea pe discheta este asemanatoare din punctul de vedere al utilizatorului cu cea pe caseta. Toate optiunile sint similare. Observatia se mentine si la instructiunile urmatoare:

LOAD*, MERGE*, VERIFY*.

5.10.5 LOAD*

Incarca un fisier de tip program de la unitatea specificata. Sintaxa:

LOAD *"m",n,"nume fisier" urmat de: **CODE [adr],[lung]**
SCREEN$
DATA x(i,j)

Exemplu: **LOAD *"m",1,"Gizeh"**
LOAD *"m",3;"Tutankamon"CODE 32768,1000

Executie: Este incarcat fisierul daca tipul specificat in instructie coincide cu tipul cu care a fost salvat.

5.10.6 MERGE*

Adaugare program BASIC. Sintaxa:

MERGE *"m",n,"nume fisier"

Exemplu: **MERGE * "m",2,"Tungus"**

Executie: Programul specificat este adaugat programului existent deja in memoria calculatorului. Liniile vechi, al caror numar se suprapune peste un numar de linie nou, sint sterse.

5.10.7 VERIFY*

Verifica coincidenta a doua programe. Sintaxa:

VERIFY * "m",n,"nume fisier"

Exemplu: **VERIFY * "m",1;"Gemenii"**

Executie: Programul specificat in instructie este comparat, octet cu octet, cu cel din memorie. In cazul in care coincid, este tiparit un mesaj OK.

5.10.8 OPEN#

Deschide canal de date. Sintaxa:

OPEN # ns,"m",n,"nume fisier"

Exemplu: **10 OPEN # 4,"m",1,"Da Vinci"**

Executie: Este creat un canal de date cu numarul logic ns si asociat dischetei din unitatea 1. Fisierul "**nume fisier**" este deschis in citire daca numele exista pe discheta; daca nu, e deschis in scriere.
OBS. Daca dupa numar canal se foloseste drept separator "," (virgula), atunci se deschid canalele de asociate versiunii de lucru 48 BASIC (in cazul canalelor b, B, m si M de la Tim-S). Daca separatorul este "; se deschid canale de tip Interface I.

Exemplu: **OPEN #5,"m"** - imprimanta MIM-40;
OPEN #5;"m",1,"TEST" - canal floppy-disc.

5.10.9 PRINT#

Tipareste in canal. Sintaxa:
PRINT # ns,lista

unde **lista** accepta toate facilitatile de PRINT din BASIC-ul standard sint valabile si respectate.

Exemplu: **20 PRINT # 4,"Tiparire demonstrativa";AT20,10;1989**

Executie: Este creat un fisier de tip date pe discheta, facilitate care nu exista la fisierele pe caseta. Un fisier odata deschis cu OPEN # in scriere, se poate scrie in el cu PRINT# pina la inchiderea canalalului cu CLOSE#.

5.10.10 INPUT#

Citirea variabilei din canal. Sintaxa:

INPUT # ns;var

unde **var** este o variabila careia i se asociaza numarul sau sirul alfanumeric citit de pe discheta.

Exemplu: **30 INPUT #4;a$**

Executie: Variabila specificata - a$ - va avea valoarea citita de pe discheta. Variabilele trebuie citite in ordinea depunerii lor in canal.

5.10.11 INKEY$

Citeste cod din canal. Sintaxa:

INKEY$ # ns

Exemplu: **40 PRINT INKEY$ #4**

Executie: Similar cu functia INKEY$ din versiunea 48 BASIC, octetul este preluat, dar nu de la tastatura, ci din canal.

5.10.12 CLOSE#

Inchidere canal de date. Sintaxa:

CLOSE # ns

Exemplu: **CLOSE # 4**

Executie: Inchide canalul cu numarul logic ns, deschis in prealabil cu OPEN #. Daca a fost deschis in scriere este apelata o rutina de inregistrare, in cazul in care tamponul canalului nu este gol, iar daca a fost deschis in citire, eventualele date

necitite se pierd din memorie.

5.10.13 MOVE

Mutare fisier de tip date. Sintaxa:

```
          #nsd                          #nsr
MOVE                          TO
          "m",nd,"numfisd"              "m",nr,"numfis"
```

```
Exemplu: MOVE "m",1,"Fisier" TO "m",1,"Fismutat"
         MOVE "m",1,"Fisier" TO #2
         MOVE "m",1,"Fisier" TO "m",2,"FISIER"
```

Executie: Este creat un nou fisier (receptor) identic cu primul (debitor) cu nume schimbat (sau cu acelasi nume daca discheta este alta, in alta unitate). Este posibila mutarea fisierului intr-un canal logic deschis cu OPEN #, avind astfel posibilitatea de a completa fisiere de tip date.
OBS. Fisiere create cu SAVE * nu pot fi citite cu INPUT# sau INKEY$ si nici mutate cu MOVE. La fel fisiere create cu PRINT# nu pot fi citite cu LOAD *, MERGE * sau VERIFY *. Trebuie facuta deosebirea intre fisiere de tip program (similare celor de pe caseta) si fisiere de tip date.

5.10.14 CLS#

Sterge ecran si intializare atribute de culoare. Sintaxa:

```
CLS #
```

Executia: Instructia este similara comenzii dedicata stergerii ecranului (CLS) din 48 BASIC, in plus initializeaza atributele de culoare, dupa cum urmeaza:

```
PAPER 7 (alb)
BORDER 7 (alb)
INK 0  (negru)
```

5.10.15 CLEAR#

Anuleaza canalele de date. Sintaxa:

```
CLEAR #
```

Executia: Instructia aduce zona de canale de date la forma ei minima, stergind toate canalele deschise de utilizator.

5.11 Utilizarea rutinelor din Interface I

Din modul de lucru 48 BASIC se pot apela subrutine specifice din pagina ce contine softul extensiei de BASIC, Interface I (p11). Operatia este mult usurata de utilizarea asa-ziselor "hook-coduri". Un hook-cod este un octet (intre #1B si #32), prezent in programul utilizator dupa codul #CF, codul istructiei RST #08. De exemplu, sa incarcam in memorie, incepind cu adresa #A000, urmatoarea secventa de program:

```
A000 CF RST#08
A001 23 DEFB 35
A002 ...
```

Rezultatul executiei instructiei cu codul #CF (in modul de lucru 48 BASIC) este suspendarea executiei actualei secvente de program, lansarea in executie a rutinei din **Interface I** selectata pe baza hook-codului #35, urmata, in general, de revenirea la adresa #A002, de unde urmeaza a fi preluat urmatorul cod de instructie.

Apelul prin hook-coduri are avantajul ca ocupa doar 2 octeti si nu 3, cum ar fi in cazul apelului cu CALL, dar mai ales a fost necesar deoarece subrutinele apelate sint in softul **Interface I**, ceea ce inseamna ca este necesara o paginare preliminara, iar dupa executie o repaginare (se considera ca programele utilizator ruleaza sub controlul softului 48 BASIC).

In cazul apelarii prin hook-coduri, RST#08 produce comutarea accesului la memorie in cadrul primului sfert, de pe versiune 48 BASIC pe versiunea Interface I. Acest lucru se intimpla indiferent de codul de dupa #CF. Daca acest cod este intre #FF si #1A inseamna ca o eroare a fost depistata in programul BASIC si un mesaj de eroare va urma sa fie tiparit. Daca este cuprins intre #1B si #32, atunci este vorba de o instructie de tip extensie, care urmeaza a fi tratata in soft-ul Interface I, fara a se mai afisa mesajul de eroare (daca este corecta). Selectia instructiei se face chiar pe baza valorii hook-codului. Codurile intre #33 si #FE genereaza un mesaj de eroare "**Hook code error**".

De notat ca nici un registru nu este salvat. Utilizatorul trebuie, in unele cazuri, sa salveze H'L'.

5.11.1 Hook-codurile Interface I

Hook-codurile au urmatoarele semnificatii la Interface I:

#1B - CONSOLE INPUT
#1C - CONSOLE OUTPUT
#1D - RS232 INPUT
#1E - RS232 OUTPUT
#1F - ZX PRINTER OUTPUT
#20 - KEYBOARD TEST
#21 - SELECT DRIVE
#22 - OPEN FILE
#23 - CLOSE FILE
#24 - DELETE FILE
#25 - READ SEQUENTIAL
#26 - WRITE RECORD
#27 - READ RANDOM
#28 - READ SECTOR
#29 - READ NEXT
#2A - WRITE SECTOR
#2B - CREATE BUFFER
#2C - DELETE BUFFER
#2D - OPEN NETWORK CHANNEL
#2E - CLOSE NETWORK CHANNEL
#2F - GET PACKET
#30 - SEND PACKET
#31 - CREATE SYSTEM VARIABLES
#32 - CALLING Tim-S Plus

5.11.2 Detalii despre hook-coduri

#1B - CONSOLE INPUT

Subrutina asteapta ca o tasta sa fie apasata. Cind acest lucru s-a intimplat, codul caracterului este gasit si returnat in registrul A. Intreruperea mascabila trebuie sa fie validata inainte de apelare.

#1C - CONSOLE OUTPUT

Subrutina tipareste in canalul 2 (in mod normal - ecranul superior) caracterul al carui cod, este in registrul A. Acelasi efect l-ar avea:

```
PUSH AF
LD A,#02
CALL SELECT
POP AF
RST #10
```

SELECT este un subprogram din 48 BASIC care, in cazul variantei de apelare de mai sus, asociaza canalului 2 ecranul superior. De remarcat ca defilarea (scroll) este oprita.

#1D - RS232 INPUT

Subrutina preia octetul de date de la interfata seriala. Rata de transfer este controlata prin valoarea variabilei de sistem extins BAUD (#5CC3/4) iar valoare BORDER-ului prin variabila IOBORD (#5CC6).

Octetul primit prin legatura serie este returnat in A. prin intermediul variabilei de sistem extins SER-FL (#5CC7/8),care se recomanda a se anula inainte de primirea primului caracter.

#1E - RS232 OUTPUT

Este subrutina corespondenta precedentei (transfera octet de date). Nici un octet de date nu va fi transmis atita timp cit linia LDTR este la 1 logic.

Octetul ce urmeaza sa fie transmis se depune in registrul A.

#1F - PRINTER OUTPUT

Subrutina este identica cu CONSOLE OUTPUT, numai ca in locul canalului 2, este utilizat canalul 3 (in mod normal imprimanta).

#20 - KEYBOARD TEST

Subrutina foloseste urmatoarele 5 instructii pentru a testa apasarea unei taste.

```
AF XOR A
DBFE IN A,(#FE)
E61F AND #1F
D61F SUB #1F
C6FF ADD A,#FF
```

Executia subrutinei are ca efect pozitionarea fanionului CY, astfel: - CY=0; nici o tasta nu este apasata;
- CY=1; s-a apasat o tasta.

#21 - SELECT DRIVE

Hook-codul a fost pastrat pentru compatibilitate cu microdrive. Selectia unitatii de disc se face in modul de acces la disc daca in prealabil am incarcat la D-STR1 (#5CD6/7) numarul unitatii.

Acesta este primul hook-cod dintre cele 13 care se refera la discul flexibil.

#22 - OPEN FILE

Acest hook-cod serveste programatorului in cod masina pentru a crea un canal "ad-hoc" pentru floppy. Adresa de baza a canalului este returnata in registrul IX.

Hook-codul se utilizeaza astfel:

- se asigura ca variabilele de sistem extins sint inserate (la nevoie se utilizeaza hook-codul #31);
- intotdeauna se salveaza H'L';
- se introduce numarul unitatii de disc in D-STR1;
- se scrie numele fisierului in locatii succesive
- se introduce descriptorul numelui fisierului in N-STR1 in ordinea: 1) lungimea low;
  2) lungimea high;
  3) adresa de start low;
  4) adresa de start high;
- apeleaza OPEN FILE din Interface I;
- reface H'L' inaintea intoarcerii in 48 BASIC.

Hook-codul #22 se utilizeaza numai pentru a deschide fisiere de tip date si are acelasi efect ca si instructia OPEN#; daca numele fisierului este unul nou, atunci fisierul este deschis in scriere; altfel este deschis in citire si primul sector este incarcat in buffer. Odata canalul creat, fisierul poate fi manipulat in urmatorul mod:

- declararea canalului ca si canal curent astfel:

```
PUSH IX
POP HL
LD (CURCHL),HL;
```

- scriere in fisier utilizind RST#10 cu codul caracterului in A sau citire din fisier utilizind

```
CALL #15E6 (INPUT-AD).
```

OBS. Al 513-lea, 1025-lea... octet scris in canal va genera o noua inregistrare (sector scris) pe discheta.

Similar la citire.

#23 - CLOSE FILE

Actiunea acestei subrutine este similara acelei a instructiunii CLOSE#. Daca fisierul a carei adresa de baza este in IX este deschis pentru scriere, toate datele care nu au fost inca inregistrate pe discheta formeaza o inregistrare **"End of file"** si canalul este sters. Dar, atunci cind canalul a fost deschis pentru citire, toate datele din buffer sint pierdute si canalul este sters.

#24 - DELETE FILE

Subrutina poate fi folosita pentru a sterge un fisier de pe discheta. Parametrii fisierului trebuie sa fie pregatiti in D-STR1 si N-STR1. Cu acest hook-cod pot fi sterse si fisiere de tip program si tip date.

#25 - READ SEQUENTIAL

Acest hook-cod ajuta utilizatorul la incarcarea urmatoarei inregistrari (sector) a fisierului tip date in bufferul canalului. La intrarea in subrutina, IX trebuie sa contina adresa de baza a canalului, iar variabila de canal CHREC numarul sectorului curent. CHREC va fi incrementat.

#26 - WRITE RECORD

Hook-codul serveste la crearea unei inregistrari (sector). Ca de obicei IX trebuie sa contina adresa de baza a canalului care contine datele pentru inregistrare. Inregistrarea va fi creata in urmatorul sector liber al dischetei.

#27 - READ RANDOM

Subrutina este similara cu READ SEQUENTIAL, numai ca se citeste sectorul din fisier al carui numar este dat de continutul variabilei CHREQ.

#28 - READ SECTOR

Aceasta subrutina incarca inregistrarea din sectorul specificat de CHREQ. Nu conteaza carui fisier ii apartine aceasta inregistrare.

#29 - READ NEXT

Aceiasi functie cu READ SECTOR.

#2A - WRITE SECTOR

Subrutina efectueaza actiunea inversa lui READ SECTOR. Datele continute in bufferul canalului specificat de IX sint scrise in sectorul CHREQ al dischetei.

#2B - CREATE BUFFER

Subrutina creaza un canal si zona MAP (harta de fisier) necesara pentru fisierul specificat de D-STR1 si N-STR1 (la INTERFACE-1 executa aceiasi actiune ca si OPEN FILE).

#2C - DELETE BUFFER

Canalul si MAP-ul sau (numai in cazul in care nu mai este deschis alt canal pentru aceeasi unitate) sint sterse. Toate zonele relocatabile sint mutate mai in jos cu 627 octeti (spre adrese mai mici).

#2D - OPEN NETWORK CHANNEL

Subrutina creaza canal pentru reteaua locala de microcalculatoare. Statia destinatie trebuie sa fie in D-STR1 si numarul statiei proprii in NTSTAT. Ca de obicei, IX este returnat cu adresa de baza a canalului si se pot emite date cu RST#10 sau

receptiona cu CALL #15E6, dupa ce canalul a fost declarat curent.

#2E - CLOSE NETWORK

Subrutina utilizata la inchiderea uni canal de retea. Daca a fost deschis pentru scriere, datele netransmise se transmit, iar daca a fost deschis pentru citire, datele necitite se pierd si canalul se sterge.

#2F - GET PACKET

Subrutina ajuta la incarcarea unui pachet de date care formeaza pachetul transmis in retea. Variabilele canalului NCIRIS, NCSELF si NCNUMB trebuie sa aiba valorile necesare.

#30 - SEND PACKET

Opusa subrutinei precedente, trimite un pachet in retea. In intrare, A trebuie sa contina 0 pentru un bloc de date si 1 pentru un bloc "End of file". Acest "hook-cod" si precedentul fac ca NCNUMB sa avanseze daca executia se incheie fara erori.

#31 - CREATE SYSTEM VARIABLES

Este un hook-cod care nu are nici o actiune daca variabilele de sistem extensie au fost deja inserate. Daca nu, atunci le initializeaza.

#32 - CALLING Tim-S Plus

De fapt este un hook-cod care se deosebeste de toate celelalte prin faptul ca lasa softul extensiei operational, in cadrul primului sfert, la indemina utilizatorului si nu are o adresa anume de start a executiei. Cu ajutorul lui pot fi apelate subrutine din Interface I, punind mai intii in variabila de sistem HD-11 (#5CED/E) adresa subrutinei apelate. In incheierea acestei rutine, ce ruleaza in conditiile in care in cadrul primului sfert este accesibil softul extensiei BASIC, se recomanda revenirea in modul de lucru 48 BASIC, prin repaginare cu ajutorul instructiei JP #0700.

5.12 Variabilele de sistem extinse - Interface I

FLAGS3 - (#5CB6)

Fiecare bit al octetului adresat cu #5CB6 are o anumita semnificatie, dupa cum urmeaza:

Bit	Semnificatie
0	setat daca o comanda extensie a fost executata
1	setat daca s-a efectuat o instructie CLEAR#
2	setat daca variabila de sistem ERR SP a fost alterata de softul de extensie Interface I
3	setat pentru rutinele de retea
4	setat la LOAD*
5	setat pentru SAVE*
6	setat pentru MERGE*
7	setat pentru VERIFY.

VECTOR - (#5CB7/8)
Contine adresa la care se sare daca nu s-a gasit sintaxa corecta nici in soft-ul extensiei.

SBRT - zona (#5CB9-#5CC2)
In mod strict nu este o variabila de sistem, ci o subrutina scurta folosita de softul-ul extensie pentru a apela rutine din softul-ul principal (48 BASIC).

BAUD - (#5CC3/4)
Constituie rata de transfer pentru RS232.

NTSTAT - (#5CC5)
Este numarul statiei de retea curente atribuita microcalculatorului.

IOBORD - (#5CC6)
Urmareste culoarea BORDER-ului in timpul operatiilor de intrare/iesire, fiind incarcata cu codul acestei culori. Normal este 0 pentru negru, dar poate fi schimbat.

SER-FL - (#5CC7/8)
Este folosita la RS232 pentru receptie. Primul octet este fanion, resetat la inceputul unei subrutine de intrare si setat cind un octet a fost primit. Al doilea octet este octetul receptionat, la intoarcerea din subrutina de intrare.

SECTOR - (#5CC9/A)
Memoreaza sectorul header al fisierului.

CHADD - (#5CCB/C)
Este echivalenta variabilei de sistem CH-ADD din soft-ul principal.

NTRESP - (#5CCD)
Este codul raspuns dat in retea.

NTDEST - (#5CCE)
(Este) Contine statia destinatie in retea.

NTSRCE - (#5CCF)
Contine statia sursa in retea.

NTNUMB - (#5CD0/1)
Contine numarul blocului de retea curent transmis.

NTTYPE - (#5CD2)
Contine identificatorul pentru un bloc de retea: 0 pentru bloc normal si 1 pentru blocul final.

NTLEN - (#5CD3)
Contine lungimea blocului de retea transmis.

NTDCS - (#5CD4)
Contine suma de control a blocului de urmarit.

NTHCS - (#5CD5)
Contine suma de control a celor 7 primi octeti ai header-ului. Urmatorii opt octeti constituie identificatorul primului fisier.

D-STR1 - (#5CD6/7)
Folosita pentru operatii cu floppy-disc-ul, contine numarul unitatii de disc, pe 2 octeti. Pentru operatii in retea, contine numarul statiei destinatie. Pentru RS232 contine rata de transfer.

S-STR1 - (#5CD8)
Contine numarul canalului.

L-STR1 - (#5CD9)
Contine tipul canalului.

N-NSTR1 - (#5CDA/B)
Contine lungimea numelui fisierului.

T-STR1 - (#5CDC/D)
Contine adresa de inceput a numelui fisierului. Urmatorii 8 octeti sint folositi pentru LOAD si MOVE.

D-STR2 - (#5CDE/F) pina la... T-STR2 - (#5CE4/5)
Acesti 8 octeti sint aceiasi ca si cei 8 precedenti, care constituie identificatorul primului fisier.
Octetii urmatori sint echivalenti cu octetii de header din soft-ul principal, subrutinele lui, dar sint folositi de softul-ul de extensie.

HD-00 - (#5CE6)
Contine tipul fisierului: 0 program, 1 matrice numerica, 2 matrice, 3 cod-masina.

HD-0B - (#5CE7/8)
Contine lungimea datelor.

HD-0D - (#5CE9/A)
Contine adresa de inceput a datelor.

HD-0F - (#5CEB/C)
Contine numele matricii sau lungimea programului.

HD-11 - (#5CED/E)
Contine numarul liniei auto-start. Poate fi folosita si de hook-codul #32.

COPIES - (#5CEF)
Dicteaza numarul de copii facute la instructiunea SAVE*.

5.13 Structura unui canal de discheta

Octeti	Semnificatie
0 -1	Adresa 0008#
2 -3	Adresa 0008#
4	"M" ("M" + #80 daca este canal de tip "ad hoc")
5 -6	Adresa MWRCH (adresa de iesire a subrutinei)
7 -8	Adresa MDRCH (adresa de intrare a subrutinei)
9 -10	Numarul 595 (lungimea canalului de unitate logica)
11 -12	CHBYTE (contor pentru zona de date)
13	CHREC (numarul sectorului: 0÷255)

-23	14 -23	CHNAME (numele fisierului sau unitatii logice)
	24	CHFLAG (fanioane)
	25	CHDRIVE (numarul unitatii logice)
-27	26 -27	CHMAP (adresa MAP)
	28	HDNUMB (numarul sectorului)
-61	29	HDFLAG (fanioane)
30 -61	30 -61	MF (harta fisierului)
62 -62	62 -63	LENF (lungimea fisierului)
64 -61	64 -66	Neutilizati
67	67	RECFLG (fanioane)
	68	Neutilizat
	69 -70	RECLEN (lungimea inregistrarii)
	71 -81	Neutilizati
	82 -593	DATA AREA (zona de date)
	594	Neutilizat