GPSS/H

GPSS/H (General Purpose System Simulator) este un simulator proprietar (http://www.wolverinesoftware.com), destinat simularii sistemelor generale bazate pe evenimente (manufacturare, transport, distributie, comunicatii, spitale, calculatoare, logistica, etc.). El a fost lansat in octombrie 1961 de catre IBM si a cunoscut o continua dezvoltare, de la un instrument pentru utilizatori fara cunostinte de programare, care "desenau" sistemul catre un limbaj de programare adevarat.

GPSS/H a introdus paradigma modelarii prin flux de tranzactii, in care consumatori *activi* (tranzactii) calatoresc preintr-o diagrama de blocuri care reprezinta un sistem, concurand pentru folosirea unor resurse (obiecte *pasive*). Intr-un model GPSS/H o tranzactie poate reprezenta un pacient, un apel telefonic, un camion, un pachet de date, un student, nu proces sau orice identitate care are un comportament. Executarea unui model implica existenta mai multor tranzactii in sistem, fiecare avand un drum propriu prin blocurile definite. Tranzactiile lupta pentru ocuparea resurselor sistemului, fiind automat puse in cozi de asteptare daca resursele cerute sunt ocupate.

Varianta folosita este **Student GPSS/H**, o versiune complet functionala care permite realizarea unor modele de dimensiuni reduse, avand urmatoarele llimitari:

- Maxim 125 blocuri GPSS/H.
- Maxim 250 linii (blocuri si comenzi).
- Maxim 32720 octeti COMMON.
- Maxim 300 tranzactii simultante (mai putine insa daca se folosesc matrici mari).

Fiecare model GPSS/H este format din entitati, caracterizate de o serie de atribute care descriu starea la un moment dat. Majoritatea atributelor sunt adresabile doar intern, de catre limbaj. Exista insa si o serie de atribute adresabile de catre utilizator, numite atribute standard.

Entitati GPSS/H

Entitati de baza

Sunt blocurile si tranzactiile. Blocurile se caracterizeaza printr-un nume si o serie de argumente.

- Blocuri de actiune: SEIZE/RELEASE, PREEMPT, RETURN, ENTER/LEAVE, QUEUE/DEPART, LINK/UNLINK, SPLIT, ASSEMBLE, GATHER, MATH, ADVANCE, BUFFER, JOIN/REMOVE, SCAN.
- Blocuri pentru generarea si distrugerea tranzactiilor: GENERATE/TERMINATE.
- Blocuri de control deplasare: TEST, TRANSFER, GATE, EXAMINE, LOOP.
- Blocuri de modificare: INDEX, ALTER, ASSIGN, PRIORITY, INITIAL, LOGIC, MARK, SAVEVALUE/MSAVEVALUE, COUNT, SELECT, HELP.
- Blocuri pentru statistici: QUEUE, TABULATE, BTABULATE.
- Blocuri pentru listari: PRINT, TRACE, WRITE, SAVE/READ.

Fiecare tranzactie are o prioritate si poate avea maxim 100 parametri.

Entitati de tip echipament

Sunt resursele simple (facilitati FACILITIY), resursele multiple (STORAGE), comutatorii logici, caracterizate printr-o serie de atribute, care definesc gradul de ocupare.

Entitati de calcul

Sunt variabile numerice (VARIBLE, FVARIABLE), variabile logice (BVARIABLE), functii (FUNCTIONA), functii analitice.

Entitati statistice

Sunt cozile (QUEUE) si tabelele de frecvetna uni si bidimensionale.

Entitati de referinta

Sunt folosite pentru referinta valorilori nitiale (INITIAL).

Entiati de tip lant

Tranzactiile pot fi in urmatoarele stari:

- Asteptare.
- Intarziere.
- Blocate.
- Libere.

Toate tranzactiile in aceeasi stare formeaza lanturi de tranzactii.

Exista urmatoarele lanturi:

- Lantul evenimentelor curente.
- Lantul evenimentelor viitoare.
- Lantul tranzactiilor intrerupte (prioritate).
- Lantul de sincronizare.
- Lantul de intarziere.
- Lanturile utilizatorului (LINK/UNLINK).

Entitati de grup

GROUP, JOIN, REMOVE, EXAMINE, SCAN, ALTER.

Principalele blocuri GPSS/H

Blocuri care modelează activitatea tranzacțiilor

Sosirea (intrarea în sistem)

```
GENERATE      A,B,C,D,E,F,G,H,I
```

- A - interval mediu între sosiri
- B - abatere
- C - momentul sosirii primei tranzacții
- D - numărul maxim de tranzacții generate
- E - prioritate (valoare implicită = 0)
- F, G, H, I - numărul parametrilor din diferite categorii, ataşați tranzacțiilor generate si se reprezintă sub forma <număr><tip>, unde <tip> este:
  PH - întreg pe 2 octeți (Halfword)
  PF - întreg pe 4 octeți (Fullword)
  PB - întreg pe 1 octet (Byte)
  PL - real (fLoat)

La generare toți parametrii sunt inițializați la 0.

Exemple

a) sosire la intervale fixe de timp:

```
GENERATE        6           sosiri din 6 in 6 unitati de timp
```

b) sosiri la intervale de timp uniform distribuite între min şi max, A = (min + max) / 2 ; B = A – min:

```
GENERATE        15,4        min = 11, max = 19
```

echivalent cu

```
GENERATE        FRN(1)*8+11
```

c) sosiri la intervale fixe de timp A = 6 , începând de la momentul de timp C = 1:

```
GENERATE        6,,1
```

d) sosiri la intervale de timp uniform distribuite, cu media 15 şi abaterea 5, momentul sosirii primei tranzacții fiind impus:

```
GENERATE        15,5,2
```

În acest caz prima tranzacție soseşte la t = 2, iar următoarele tranzacții sosesc la intervale uniform distribuite între 10 şi 20 unități de timp.

e) se generează 25 tranzacții, care sunt introduse în sistem de la începutul simulării:

```
GENERATE        ,,,25
```

f) se generează cel mult 15 tranzacții, care sosesc, începând de la t = 50, la intervale de câte 30 unități de timp şi au prioritatea 100:

```
GENERATE        30,,50,15,100
```

g) se generează tranzacții cu doi parametri de tip Byte şi un parametru de tip real, care sosesc la intervale uniform distribuite între 6 şi 15 unități de timp, A = 10.5, B = 4.5:

```
GENERATE        10.5,4.5,,,,2PB,1PL
```

Secvente de numere aleatoare

RN(id)

- ca argument de funcție => valoare reală în domeniul (0, 1);
- în alt context => valoare întreagă în domeniul [0, 999]

FRN(id) - valoare reală în domeniul (0, 1)

Efectuarea unei activități cu o anumită durată, fixă sau variabilă

```
ADVANCE         A,B
```

- A - durată medie;
- B - abatere

Echivalent cu

```
ADVANCE         FRN(1)*(2*B)+A-B
```

Exemple

a) activitatea durează 15 unități de timp:

```
ADVANCE         15
```

b) activitatea durează între 7 şi 15 unități de timp:

```
ADVANCE     11,4
```

Părăsirea sistemului

```
TERMINATE   A
```

A - valoarea care se scade din cea a contorului de terminări, inițializat prin comanda START; în cazul în care valoarea contorului devine ≤ 0, simularea este oprită.

Pentru a modela **părăsirea sistemului fără modificarea contorului de terminări** nu se foloseşte operandul a, ca în exemplul următor:

```
TERMINATE
```

Modificarea priorității

```
PRIORITY    A
```

- A - expresie reprezentând noua valoare a priorității

Exemple

```
PRIORITY    50
PRIORITY    PR+1      prioritatea creste cu o unitate
```

Modificarea valorii unui parametru

```
ASSIGN      A[<op>],B,C
```

- A - identificatorul parametrului
- <op> - operația (+ sau -) efectuată între vechea valoare şi B
- B - expresia valorii modificatoare
- C - tipul parametrului (PH / PF / PB / PL)

Exemple

```
ASSIGN      COD,5,PB      COD = 5
ASSIGN      COD+,7,PB     COD = 12
ASSIGN      COD-,2,PB     COD = 10
```

Marcarea momentului de timp curent (necesară pentru a măsura timpul scurs între două evenimente)

```
MARK
```

Pentru a afla timpul scurs de la ultima marcare se utilizează atributul standard **M1**. Dacă tranzacția nu a trecut prin nici un bloc MARK valoarea atributului M1 reprezintă timpul scurs de la generare.

Blocuri care modelează interacțiuni tranzacții – resurse

```
SEIZE       A       ocupare resursă simplă
RELEASE     A       eliberare resursă simplă
```

- A - identificatorul resursei

```
ENTER       A[,B]   ocupare elemente resursă multiplă
LEAVE       A[,B]   eliberare elemente resursă multiplă
```

- A e identificatorul resursei.
- B e numărul de elemente ocupate / eliberate (implicit 1); este limitată superior de capacitatea resursei multiple, fixată printr-o declarație STORAGE cu structura.

```
STORAGE     S(Id1),c1[/S(Id2),c2/...]
```

- Id_i - identificator resursă.

- c_i - capacitate (număr unități).

Blocul SEIZE acceptă tranzacția numai dacă resursa simplă este liberă.

Blocul LEAVE acceptă tranzacția numai dacă resursa multiplă are suficiente elemente disponibile.

Blocuri care modelează interacțiuni tranzacții - comutatoare

```
LOGIC R       A    resetare
LOGIC S       A    setare
LOGIC I       A    inversare stare
```

- A - identificatorul comutatorului

Blocuri care controlează colectarea de statistici

```
QUEUE         A    marchează începutul colectării
DEPART        A    marchează sfârşitul colectării
```

- A - identificatorul entității statistice

Cel mai adesea se colectează statistici referitoare la cozile de aşteptare care se formează pentru utilizarea de resurse, dar pot să se colecteze şi statistici cu privire la timpul de tranzit între două puncte din model.

Blocuri care controlează fluxul tranzacțiilor

Transfer (salt) necondiționat

```
TRANSFER      ,B
```

- B - identificatorul blocului la care este trimisă tranzacția

Transfer probabilistic

```
TRANSFER      A,[B],C
```

- A - valoare pozitivă subunitară reprezentând probabilitatea ca tranzacția să fie trimisă la blocul C
- B, C - identificatori (etichete) de blocuri; dacă B lipseşte, atunci transferul se face la blocul imediat următor

Exemple:

a) situația în care 65% dintre tranzacțiile care trec printr-un punct trebuie transferate la blocul cu eticheta BETA, iar restul (35%) la blocul cu eticheta ALFA, se poate modela în două variante:

```
TRANSFER      .35,BETA,ALFA
```

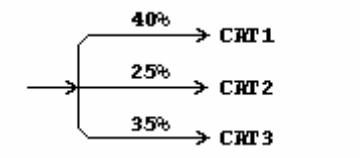
sau

```
TRANSFER      .65,ALFA,BETA
```

b) situația în care 75% dintre tranzacții trebuie transferate la blocul imediat următor, iar restul (25%) la blocul cu eticheta REST se modelează astfel:

```
TRANSFER      .25,,REST
```

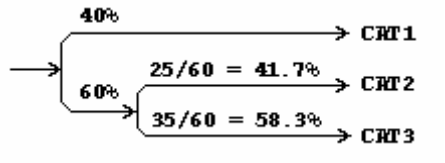
c) pentru a modela ramificarea



sunt necesare două blocuri TRANSFER.

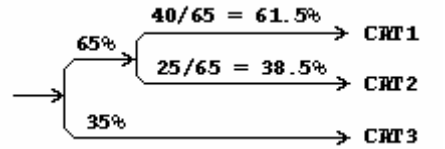
În cele ce urmează prezentăm două variante de modelare.

c.1)




```
TRANSFER    .40,,CAT1
TRANSFER    .583,CAT2,CAT3
```

c.2)




```
TRANSFER    .35,,CAT3
TRANSFER    .385,CAT1,CAT2
```

Salt condiționat de relația dintre două valori numerice

```
TEST r          A,B[,C]
```

- r - relația testată, poate fi:
  - E - egalitate
  - L - mai mic
  - G - mai mare
  - NE - inegalitate
  - LE - mai mic sau egal
  - GE - mai mare sau egal
- A, B - valorile între care se verifică relația - pot fi constante, atribute standard sau expresii; dacă relația este îndeplinită, atunci tranzacția trece la **blocul imediat următor**
- C- eticheta blocului la care este transferată tranzacția dacă valorile A şi B nu îndeplinesc relația **r**; în cazul în care C lipseşte, tranzacția aşteaptă îndeplinirea relației.

Exemple

a)

```
TEST L          Q(COADA),5,PLEC
```

Tranzacțiile îşi continuă deplasarea în flux normal numai dacă în COADA există mai puțin de 5 tranzacții; în caz contrar tranzacțiile sunt transferate la blocul cu eticheta PLEC.

b)

```
TEST GE         S(PACHETE),4
LEAVE           PACHETE,4
```

Tranzacțiile care eliberează câte 4 elemente ale resursei multiple PACHETE trebuie să aştepte ca numărul de elemente utilizate să fie de cel puțin 4.

Salt condiționat de starea unei entități - resursă simplă, resursă multiplă sau comutator

```
GATE x          A[,B]
```

- x - condiția testată privind starea entității, cu urmatoarele valori posibile:
  - resursă simplă:
    - U - utilizată
    - NU - neutilizată

I - întreruptă
NI - neîntreruptă

resursă multiplă:
SE - complet liberă (Empty)
SNE - utilizată (Not Empty)
SF - utilizată complet (Full)
SNF - are elemente disponibile (Not Full)

comutator:
LR - în starea Reset
LS - în starea Set

- A - identificatorul entității
- B - eticheta blocului la care este trimisă tranzacția dacă entitatea nu îndeplinește condiția testată; în cazul în care B lipsește tranzacția așteaptă îndeplinirea condiției.

Exemple

a)

```
        GATE LS       LIBER
```

Tranzacțiile așteaptă trecerea comutatorului LIBER în starea Set.

b)

```
        GATE SNE      LOCURI,XXX
```

Tranzacțiile care găsesc resursa LOCURI complet liberă sunt transferate la blocul XXX.

Funcții GPSS/H

O funcție GPSS/H este caracterizată prin:

- Identificator.
- argument - orice atribut standard numeric.
- tip:
  C - continuă,
  D - discretă,
  E - cu valori atribute standard numerice sau expresii.
- număr de puncte de definiție **n**.
- valoarea argumentului și valoarea funcției în fiecare punct de definiție.

Declarația unei funcții GPSS/H are structura

```
id_func   FUNCTION    <argument>,<tip><n>
a1,v1/..../an,vn
```

Exemple

```
INCEPUT  FUNCTION     RN(2),C2       valori uniform distribuite
0,10/1,20                            in domeniul (10,20)
MODIFICA FUNCTION     RN(3),D5       5 valori:  3  3.5  4  4.5  5
0.1,3/0.25,3.5/0.55,4/0.85,4.5/1,5   cu frecv: 10% 15% 30% 30% 15%
TIMP     FUNCTION     PB(NR),E2      argument : param.NR al tranzactiei
1,FN(INCEPUT)/2,FN(MODIFICA)         val.obtinuta prin eval.altei functii
DURATA   FUNCTION     C1,C5            argument : timpul relativ
60,4/150,5.5/200,5.5/240,4.75/480,6   valoare calculată prin interpolare
```

Apelul unei funcții are structura:

```
        FN(id_func)
```

Variabile globale

a) Variabile **simple** (SAVEVALUE) - nu trebuie declarate. Modificarea valorii se realizează prin intermediul blocului

```
        SAVEVALUE      A[<op>],B,C
```

- A - identificatorul variabilei
- <op> - operație (+ sau -) efectuată între vechea valoare şi B
- B - valoare modificatoare
- C - tipul variabilei (XB, XF, XH sau XL)

Exemple

```
        SAVEVALUE      TAXA,25,XF
        SAVEVALUE      TOTAL+,PB(NUMAR),XH
```

Valoarea unei astfel de variabile este adresată sub forma

```
        <tip>(id_var)
```

b) Variabile **bidimensionale *(MATRIX)*** - trebuie declarate tipul şi dimensiunile, folosind o instrucțiune de control de forma

```
id_mat  MATRIX         <tip>,<nr.linii>,<nr.coloane>
```

cu <tip> MB, MX, MH sau ML.

Exemple

```
SUME    MATRIX         MX,1,12
TIMP    MATRIX         ML,25,3
```

Actualizarea este realizată de blocul

```
        MSAVEVALUE     A[<op>],B,C,D,E
```

- A - identificatorul matricei
- <op> - operația (+ sau -) efectuată între vechea valoare şi D
- B - indice linie
- C - indice coloană
- D - expresia valorii modificatoare
- E - tipul : MB, MX, MH sau ML

Exemple

```
        MSAVEVALUE     TIMP,PB(IL),PB(IC),M1,ML
        MSAVEVALUE     SUME+,1,V(IORA),V(TAXA),MX
```

Valoarea unui element este adresată sub forma

```
        <tip>(id_mat,linie,coloană)
```

sau

```
        <tip>$id_mat(linie,coloană)
```

Variabile ale căror valori se obțin prin evaluarea unei expresii

In funcție de tipul valorii rezultatului (logic, întreg sau real) variabila se declară în unul din următoarele moduri:

```
id_var  BVARIABLE     <expresie logică>
id_var  VARIABLE      <expresie cu valoare de tip întreg>
id_var  FVARIABLE     <expresie cu valoare de tip real>
```

Operanzi: constante, atribute standard

- Operatori: aritmetici
  + adunare
  - scadere
  * înmultire
  / împărțire
  @ modulo
- relaționali
  'E' egal
  'NE' diferit
  'L' mai mic
  'LE' mai mic sau egal
  'G' mai mare
  'GE' mai mare sau egal
- logici
  + sau
  * şi

Exemple

```
COND     BVARIABLE     (C1'L'100)*SNF(RESM)
IORA     VARIABLE      (M1+59)/60
VALOARE  FVARIABLE     PL(PONDERE)*(2+RN(2)*5/999)-PL(DIFF)
```

Referințele la valorile acestor variabile au structura

```
        BV(id_var)
```

sau

```
        V(id_var)
```

după cum variabila id_var este de tip logic sau numeric.

Tabele

Se utilizează pentru a obține informații cu privire la frecvența cu care o mărime argument se încadrează în clasele specificate la declararea tabelei.

Tabele actualizate explicit

```
id_tab  TABLE        A,B,C,D,E
```

- A - argument :expresie sau atribut standard numeric
- IA - intervalul între tranzacții
- RT - numărul de tranzacții sosite în fiecare interval de E unități de timp
- B - limita primei clase
- C - lățimea unei clase
- D - numărul de clase

```
argument:        B       B+C      B+2C       B+(D-2)*C
          <------+-------+-------+--- .... ---+-------->
clasa:       1       2       3                    D
                                                (OVERFLOW)
```

Exemple

```
NORM     TABLE          PL(NORM),0,2,12
TRI      TABLE          M1,10,20,6
```

Actualizarea se realizează la trecerea unei tranzacții printr-un bloc

```
         TABULATE       id_tab
```

Tabele actualizate automat

Aceste tabele, care se referă la distribuția timpilor de aşteptare în cozi, se declară astfel:

```
id_tab   QTABLE         A,B,C,D
```

- A - identificator coadă
- B - limita primei clase
- C - lățimea unei clase
- D - numărul de clase

Utilizarea fişierelor de intrare/ieşire

```
*-- declarații
  . . . . . .
<IdInt>  FILEDEF       <IdExtern>
  . . . . . .

*-- citire / scriere inițială
         GETLIST       [FILE=<IdInt>,]<lista>
  . . . . . .
         PUTPIC        [FILE=<IdInt>,][LINES=numar_linii,]<lista>
<exact numar_linii care descriu formatul de scriere>

*-- model
  . . . . . .

*-- citire / scriere pe parcursul execuției experimentului de simulare
         BGETLIST      [FILE=<IdInt>,]<lista>
  . . . . . .
         BPUTPIC       [FILE=<IdInt>,][LINES=<numar_linii>,]<lista>
  . . . . . .

*-- execuție experimente de simulare
  . . . . . .

*-- citire / scriere informații la început/sfârşit experiment
         GETLIST       [FILE=<IdInt>,]<lista>
  . . . . . .
         PUTPIC        [FILE=<IdInt>,][LINES=<numar_linii>,]<lista>
<exact numar_linii care descriu formatul de scriere>
  . . . . . .

         END
```

<lista> poate fi de forma &var1[, ...] sau (<sublista>, &indice=<expr1>, <expr2>)

Introducere valori variabile globale în regim de dialog

```
  . . . . . .
*-- afisare mesaj
         PUTPIC
<textul mesajului>
         GETLIST      <lista>
  . . . . . .
```

Observatii

- Daca lipseste declaratia FILEDEF, atunci <IdExtern> este identic cu <IdInt>.
- In absenta operandului FILE se lucreaza cu fisierul de intrare/iesire standard.
- In absenta operandului LINES se scrie o singura linie.

Atribute standard

Observații:

a) Atributele definite de utilizator pot fi de patru tipuri:

- **B**yte [-128, 127]
- **F**ullword [-2147483648, 2147483647]
- **H**alfword [-32768, 32767]
- f**L**oating-point (rea**L**)

b) Atributele care se referă la starea unor entități pot avea numai două valori:

- 1 = adevărat
- 0 = fals

Sistem

- AC1 - timp curent absolut
- C1 - timp curent relativ

Tranzacții

Atribute implicite:

- M1 – timpul scurs de la ultima marcare, care are loc la generare sau la trecerea printr-un bloc MARK
- PR – prioritatea curentă
- XID1 – indice atribuit la generare

Parametri definiți de utilizator, de <tip> PB, PF, PH sau PL, referiti prin expresii de forma

```
        <tip>(id_par)
```

Blocuri

- W(id_bloc) - numărul curent de tranzacții aflate în bloc
- N(id_bloc) - numărul total de tranzacții care au intrat în bloc

Resurse simple (Facilities)

- F(id_res) - starea curentă: 0 = liberă, 1 = ocupată
- FC(id_res) - numărul de tranzacții care au ocupat resursa

- FT(id_res) - timpul mediu de ocupare de către o tranzacție
- FR(id_res) - rata utilizării resursei (între 0 şi 1000, deci 685 înseamnă 68.5%)

Resurse multiple (Storages)

- R(id_res) - numărul de unități disponibile
- S(id_res) - numărul de unități ocupate
- SE(id_res) - resursă neutilizată (Empty) ? => 1 | 0
- SF(id_res) - resursă complet ocupată (Full) ? => 1 | 0
- SA(id_res) - numărul mediu de unități utilizate
- SC(id_res) - numărul total de unități utilizate
- SM(id_res) - numărul maxim de unități utilizate
- SR(id_res) - rata utilizării
- ST(id_res) - timp mediu utilizare / unitate

Comutatoare (Logic switches)

- LR(id) - în starea Reset ? => 1 | 0
- LS(id) - în starea Set ? => 1 | 0

Cozi (Queues)

- Q(idq) - lungimea curentă
- QA(idq) - lungimea medie
- QM(idq) - lungimea maximă
- QC(idq) - numărul total de tranzacții care au trecut prin coadă
- QZ(idq) - numărul de tranzacții care nu au staționat în coadă
- QT(idq) - timpul mediu de staționare în coadă a tranzacțiilor (calculat considerând toate tranzacțiile)
- QX(idq) - timpul mediu de staționare **efectivă** în coadă (calculat pentru tranzacțiile cu timp de staționare > 0)

Generatoare de numere aleatoare

- RN(id)
  ca argument de funcție => valoare reală în domeniul (0, 1);
  în alt context => valoare întreagă în domeniul [0, 999]
- FRN(id) - valoare reală în domeniul (0, 1)

Entități de calcul

Variabile globale simple (Savevalue) de <tip> XB, XF, XH sau XL.

```
<tip>(id)
```

Variabile globale bidimensionale (Matrix) de <tip> MB, MX, MH sau ML

```
<tip>$id_mat(linie,coloană)
```

sau

```
<tip>(id_mat,linie,coloană)
```

Variabile a căror valoare se obține prin evaluarea unei expresii logice (BVARIABLE) sau aritmetice (VARIABLE, FVARIABLE).

```
        BV(id_var)
        V(id_var)
```

Funcţii definite de utilizator:

```
        FN(id_func)
```

Funcţii predefinite:

- matematice:
  ABS, ACOS, ASIN, ATAN, COS, EXP, LOG, SIN, SQRT, TAN
- pentru generare de distribuţii neuniforme:
  RVEXPO(nrg,medie),
  RVNORM(nrg,medie,abatere),
  RVTRI(nrg,min,m,max)

Controlul executiei experimentelor de simulare

In cazul unui model de simulare GPSS/H experimentul de simulare se execută numai dacă utilizatorul solicită acest lucru prin comanda **SIMULATE**.

Dacă se doreşte execuţia mai multor experimente de simulare consecutive se utilizează mai multe comenzi **START**, eventual separate prin comenzi care modifică contextul execuţiei. In cazul comenzilor **START** consecutive statisticile obţinute sunt cumulative.

O secventa de **m** comenzi

```
         START      n
```

este echivalenta cu

```
         START      m,,n
```

Dacă se doreşte ca în punctele intermediare să fie tipărite numai anumite statistic, atunci se pot utiliza comenzi **PRINT** referitoare la grupurile de statistici dorite, ca în exemplul următor:

```
        GENERATE        n
        PRINT           2,4,F       statistici pt.res.simple 2,3 si 4
        PRINT           ,,Q         statistici pentru toate cozile
        TERMINATE       1
*-- executie
        START           m-1,NP
        START           1
        END
```

Comanda CLEAR

- readuce toate statisticile la 0 şi elimină toate tranzacţiile din model;
- nu sunt afectate generatoarele de numere aleatoare şi nici &variabilele.

Se utilizează în cazul execuţiei mai multor experimente, pentru determinarea efectului schimbării secvenţelor de numere aleatoare.

Exemplu

```
        INTEGER         &I
        DO              &I=1,4
        START           200
        CLEAR
        ENDDO
```

Comanda RESET

- readuce la 0 statisticile;
- **nu** elimină tranzacţiile;
- actualizează contorul asociat fiecărui bloc la numărul de tranzacţii existente în bloc (N(b) = W(b));
- **nu** actualizează generatoarele de numere aleatoare şi nici &variabilele.

Se utilizează pentru eliminarea efectului regimului tranzitoriu.

Exemplu

```
        START           rtranz      executie pentru regimul tranzitoriu
        RESET
        START           rstat       executie pentru regimul stationar
```

Utilizarea fisierelor externe

```
         [B]GETLIST      [FILE=<IdInt>,]<lista>
         [B]PUTPIC       [FILE=<IdInt>,][LINES=numar_linii,]<lista>
<exact numar_linii care descriu formatul de scriere>
```

Conventii:

- in absenta FILE=<IdInt> se citeste de la consola / se afiseaza pe ecran
- <IdInt> - identificatorului intern al unui fisier coincide cu identificatorul extern, caz in care nu poate avea extensie, sau este declarat astfel

```
<IdInt>  FILEDEF      <IdExtern>
```

- <IdExtern> poate fi o constanta sau o variabila sir de caractere

Exemple

```
FDATE    FILEDEF      'DATE.DAT'
FREZ     FILEDEF      &FISREZ
```

Daca lipseste LINES=numar_linii, atunci se scrie o singura linie

- GETLIST / PUTPIC se folosesc in zonele de declaratii
- BGETLIST / BPUTPIC se folosesc in fluxul activitatilor

Afisare la consola

```
         [B]PUTPIC       [LINES=numar_linii,]<lista>
```

Exemple

a. Afisarea unui mesaj precedat de o linie goala

```
         PUTPIC          LINES=2
<textul mesajului>
```

b. Afisarea momentului sosirii unei tranzactii

```
         GENERATE        . . .
         BPUTPIC         C1
***.**  **
```

Citire de la consola, in regim de dialog:

```
         [B]PUTPIC
<text mesaj cerere>
         [B]GETLIST      <lista>
```

Exemple

a. Citire valori &variabile

```
         PUTPIC
Numar de studenti si de calculatoare =
         GETLIST      &NRS,&NRC
```

b. Citirea identificatorului extern al unui fisier

```
         VCHAR*20     &FISREZ
         PUTPIC
Numele fisierului de rezultate:
         GETLIST      &FISREZ
```

Algoritm GPSS/H

Opereaza cu:

- LAV - Lista Activitatilor Viitoare, LVF(timpRevenire)+FIFO
- LAC - Lista Activitatilor Curente (sau blocate), HVF(PRIO)+FIFO;
- CT - Contorul de Terminari.

```
- initializare CT;
- pentru fiecare bloc GENERATE :
     planifica primele tranzactii si insereaza in LAV;
CAT_TIMP LAV este nevida si CT > 0
{- avans timp: t = timpRevenire pentru prima tranzactie din LAV;
 - transfera din LAV in LAC toate tranzactiile cu timpRevenire = t;
 - initializeaza parcurgere LAC;
 CAT_TIMP mai exista tranzactii de tratat
 {- tranzactia curenta parcurge secventa de blocuri,
    pana cand ajunge la:
      a) ADVANCE => trece in LAV;
      b) TERMINATE => paraseste sistemul (eventual modificand
                                  contorul de terminari)
      c) un bloc care nu o accepta => ramane in LAC (blocata)
  - DACA a trecut printr-un bloc cu posibilitate de deblocare
      ATUNCI  revenire la inceputul LAC
      ALTFEL  avans la urmatoarea tranzactie din LAC
 }
}
```

Depanarea unui model GPSS/H

GPSS/H este dotat cu un depanator integrat, de tip ecran, care asista utilizatorul in verificarea si depanarea modelului. Acesta permite rularea pas cu pas, stabilirea unor puncte de oprire ale executiei, reveniri in stari anterioare, vizializarea valorilor curente. El se apeleaza in felul urmator:

```
GPSS/H      model TV
```

Initialele "TV" provin de la Test Video. Efectul acestei comenzi este deschiderea unui numar de trei ferestre pe ecran:

- Fereastra cu codul sursa al modelului.
- Fereastra de stare.
- Fereastra de dialog.

Depanatorul poate fi lansat si daca se tasteaza CTRL+C in momentul executie simularii, insa in acest caz depanatorul va fi orientat linie.

Fereastra sursa urmareste blocurile prin care trec tranzactiile ce aar in sistem, afisand si informatii despre "incarcarea" curenta si maximala a blocurilor.

Fereastra de stare afiseaza valorile curente ale principalelor marimi de stare (numarul tranzactiei XACT, momentul intrarii in sistem MARK TIME a acestei, aceasul absolut ABS CLOCK si relativ REL CLOCK, blocul curent CURBLK si cel urmator NEXTBLK, valoarea contorului de terminare TTG=Terminations To Go, prioritatea tranzactiei PRIORITY, etc.)

Fereastra de dialog este cea in care utilizzatorul introduce comenzile si in care se afiseaza raspunsurile sistemului.O serie de comenzi sunt atasate tastelor functionale F1-F12 si tastelor sageti. Descrierea acestora se face in fisierul "profile.hdb".

Prinicipalele comenzi de depanare

Toate comenzile depanatorului pot fi abreviate la prima litera.

Terminarea depanarii

QUIT - Termina depanarea, dar intreaba daca simularea in curs se anuleaza.

QQ (Quick Quit) - Termina depanarea indiferent daca e un model in executie.

Puncte de oprire (breakpoints)

BREAK numebloc - pune un punct de oprire la blocul "numebloc"

UNBREAK numebloc - elimina punctul de oprire.

Se pot declara mai multe breakpoint-uri cu aceeasi comanda, ele fiind separate prin spatii. Blocul poate fi specificat si prin identificatorul sau numeric, dar e preferabil sa se specifice numele asociat. Breakpoint-urile pot fi globale (folosesc comanda RUN) sau locale (CONTINUE). Cele globale exista pana sunt eliminate explicit, iar cele locale dispar dupa executia unei comenzi.

Situatia curenta a acestor puncet se poate afla folosind comanda

```
DISPLAY BREAKPOINTS
```

Conditii de interceptare

TRAP conditie - stabileste o conditie la a carei indeplinire simularea se intrerupe

UNTRAP conditie - elimina o conditie de oprire

Exista 4 conditii de interceptare

- CLOCK: legata de valoarea ceasului absolut
  TRAP CLOCK=moment
- XACT: legata de numarul tranzactiei curente
  TRAP XACT=numar
- NEXT: suspemda simulare si afiseaza informatii despre starea oricarei tranzactii care se "trezeste" si incearca sa ocupe urmatorul bloc
- SYSTEM: afiseaza informatii despre tranzactiile care trec in repaos

Comenzi de executie

RUN [numebloc] | [conditie] - executa sau reia executia modelului, simultan cu instalearea unui punct de oprire sau a unei conditii de interceptie

CONTINUE - similara cu comanda anterioara, doar ca instaleaza breakpoint-uri locale

STEP [nrpasi] - executa numarul precizat de pasi, implicit 1

Afisarea unor informatii

DISPLAY clasa_entitate[(menbru, ...)]

Se accepta urmatoarele clase: BLO, FAC, QUE, STO, AMP, TAB. Pentru o anume clasa se afiseaza implicit toate entitatile existente, insa se pot preciza si doar anumiti membri.

DISPLAY conditie

Conditiile care pot fi folosite sunt: BREAKPOINTS, CLOCKS, OUTPUT, STATUS

DISPLAY XACT=numar

Atasarea de actiuni cu breakpoint-uri

```
AT numebloc
       …
       lista de comenzi
       …
END
```

Breakpoint-ul introdus astfel se poate elimina cu UNBREAK.