

## PREFAȚĂ

Lucrarea “**Automate Programabile – Îndrumar de laborator**” reprezintă un important suport practic de concepere și proiectare a soluțiilor de conducere a proceselor industriale automate programabile bazate pe medii de programare evolute de tipul ISaGRAF sau Ladder Diagram.

Lucrarea este dedicată în primul rând studenților facultății de Automatică și Calculatoare din Universitatea Politehnica București, care audiază cursul și frecventează laboratorul de **Automate și Microprogramare**. De asemenea, este utilă studenților din facultățile cu profil de automatică aparținând universităților tehnice din țară.

Cartea este structurată în două părți, prima parte prezentând medii de programare logică ISaGRAF, utilizat de majoritatea producătorilor de automate programabile, în special din perspectiva modului de lucru cu interfața utilizator, util în rezolvarea lucrărilor de laborator la materia “**Automate și microprogramare**”:

De asemenea, sunt prezentate 14 aplicații corespunzătoare celor 14 laboratoare efectuate de studenții facultății de Automatică și Calculatoare din Universitatea Politehnica București.

Aplicațiile au fost alese astfel încât gradul de dificultate al lor să fie diferit, de la cele mai simple, rezolvabile printr-o unică diagramă logică, la procese industriale complexe, în care dezvoltarea unui proiect de control amplu necesită exercițiu și o foarte bună gândire logică.

La unele aplicații au fost descrise două variante de programare, pe lângă soluția în ISaGRAF fiind prezentată și rezolvarea în limbajul Ladder Diagram, în scopul înțelegerii algoritmului de conversie a unei diagrame logice secvențiale într-un program ciclic de tip diagramă în trepte.

Observațiile și propunerile existente la sfârșitul fiecărei aplicații ajută viitorul specialist în automatică să înțeleagă pe deplin soluția prezentată, având un punct de pornire solid pentru a concepe modificări ale programelor conform noilor situații ce pot apărea în cadrul unui proces.

Metodele și soluțiile prezentate la toate aplicațiile sunt absolut originale și au fost realizate de către autori atât în cadrul laboratorului de “**Automate și microprogramare**” cât și datorită numeroaselor contracte de cercetare la care autorii au participat de-a lungul anilor, în cadrul centrului de cercetare de excelență CIMR din cadrul Facultății de Automatică și Calculatoare din Universitatea POLITEHNICA București.

Autorii

## Cuprins:

|  |          |
|--|----------|
| <b>1. Mediul de programare ISaGRAF</b> | <b>9</b> |
|--|----------|

## Aplicații







|   |            |
|---|------------|
| <b>1. Problema 1 – Controlul unei macarale</b>                              | <b>25</b>  |
| <b>2. Problema 2 – Comanda mișcării oscilatorii a unui mobil</b>            | <b>32</b>  |
| <b>3. Problema 3 – Detecția și expulzarea automată a sticlelor fără dop</b> | <b>37</b>  |
| <b>4. Problema 4 – Stație automată de spălat autovehicule</b>               | <b>42</b>  |
| <b>5. Problema 5 – Elevator clasificator de pachete</b>                     | <b>49</b>  |
| <b>6. Problema 6 – Dozare și malaxare automată</b>                          | <b>57</b>  |
| <b>7. Problema 7 – Umplerea și astuparea automată a sticlelor</b>           | <b>68</b>  |
| <b>8. Problema 8 – Controlul unui lift</b>                                  | <b>74</b>  |
| <b>9. Problema 9 – Detectarea bagajelor care conțin metale</b>              | <b>79</b>  |
| <b>10. Problema 10 – Sortarea a patru tipuri de piese</b>                   | <b>84</b>  |
| <b>11. Problema 11 – Umplerea automată a unor containere</b>                | <b>93</b>  |
| <b>12. Problema 12 – Controlul unor uși automate</b>                        | <b>100</b> |
| <b>13. Problema 13 – Sortarea a două tipuri de piese</b>                    | <b>104</b> |
| <b>14. Problema 14 – Procesul de coacere al biscuiților</b>                 | <b>111</b> |

|                     |            |
|---------------------|------------|
| <b>Bibliografie</b> | <b>117</b> |
|---------------------|------------|

## 1. MEDIUL DE PROGRAMARE ISaGRAF

Programul ISaGRAF a fost dezvoltat de firma CJ International și reprezintă unul dintre cele mai structurate și prietenoase medii de programare pentru automatele programabile. Este compatibil cu standardul IEC 1131-3.

Un proiect în ISaGRAF este împărțit în mai multe unități numite *programe*. Un program este o unitate logică care descrie operațiile între variabilele și constantele unui proces. Acestea sunt legate împreună într-o structură arborescentă, având icoane diferite. Programele pot fi editate într-unul din limbajele grafice sau literale:

-  **Flow Chart (FC)**
-  **Ladder Diagram (LD)**
-  **Structured Text (ST)**
-  **Instruction List (IL)**
-  **Function Block Diagram (FBD)**
-  **Sequential Function Chart (SFC)**

Un program poate fi *ciclic* sau *secvențial*. Un program ciclic se execută în întregime la fiecare ciclu, iar execuția unui program secvențial urmează regulile dinamice ale limbajului SFC sau FC. Un program nu poate conține instrucțiuni din mai multe limbaje, cu excepția limbajelor LD și FBD, care pot fi combinate în cadrul aceluiași program.

Programele sunt considerate entități de nivel unu. Se pot scrie și subprograme, care sunt entități de nivel doi. Entitățile de nivel unu sunt lansate de sistemul de operare, în timp ce subprogramele, numite și programe fiu, sunt activate de programe, care se mai numesc și programe părinte.

Programele apar într-una din următoarele trei secțiuni:

- **BEGIN**: se execută la începutul ciclului;
- **SEQUENTIAL**: se execută după cele din secțiunea BEGIN și urmează legile dinamice ale limbajului SFC sau FC;
- **END**: se execută la sfârșitul ciclului.

Programele din secțiunea BEGIN se recomandă a fi utilizate pentru realizarea unor operații preliminare asupra variabilelor de intrare (ex.: filtrarea, citirea unei valori de la un traductor).

Programele din secțiunea SEQUENTIAL descriu operații secvențiale unde variabila timp sincronizează operațiile primare. Trebuie să existe cel puțin un program secvențial, restul doar dacă este nevoie.

Programele din secțiunea END se recomandă a fi utilizate pentru realizarea unor operații de protecție înainte de a trimite o variabilă către un dispozitiv de ieșire. Programele secțiunilor BEGIN și END nu pot fi descrise în SFC sau FC.

Fiecare program din secțiunea SEQUENTIAL poate controla alte programe SFC, programele fiu. Un program fiu este un program paralel, care poate fi pornit, oprit, suspendat sau repornit de către părintele său. Important este ca ambele programe, părinte și fiu, să fie scrise în SFC sau FC.

Atunci când un program părinte pornește un program fiu transmite un jeton fiecărui pas inițial al programului fiu. Această comandă este descrisă de declarația GSTART.

În cazul opririi unui program fiu de către părintele său, sunt eliminate toate jetoanele existente. Această comandă este descrisă de declarația GKILL.

Atunci când un program părinte îngheață un program fiu, șterge toate jetoanele din el, dar memorează poziția acestora pentru cazul în care programul fiu va fi repornit. Programul suspendat poate fi repornit folosind declarația GRST.

Și în cazul programelor FC ale secțiunii secvențiale pot exista subprograme FC fiu, dar un părinte FC este blocat în timpul execuției unui subprogram FC. Astfel nu sunt posibile operații simultane în programul FC părinte și programul fiu FC.

Obiectele din ISaGRAF sunt utilizate în orice program scris în FC, LD, ST, IL, FBD sau SFC. Aceste obiecte sunt: tipuri de bază, expresii constante, variabile și comentarii.

Principalele tipuri de variabile disponibile pentru programe sunt:

|          |  |
|----------|--|
| BOOLEAN: | valori binare de tipul true/ false;        |
| ANALOG:  | valori întregi (integer) sau reale (real); |
| TIMER:   | valori de tip timer;                       |
| MESSAGE: | șiruri de caractere.                       |

Constantele de tip întreg sunt reprezentate pe 32 de biți, având valorile cuprinse între  $-2^{32}$  și  $2^{32}$ . Pot fi exprimate în baza zecimală, în baza hexazecimală (valoarea trebuie precedată de 16#), în bază octală (valoarea trebuie precedată de 8#) și în binar (valoarea trebuie precedată de 2#).

Constantele analogice reale pot fi scrise fie printr-o reprezentare zecimală, fie printr-o reprezentare științifică. Punctul zecimal este folosit pentru a diferenția o constantă reală de una de tip întreg. Reprezentarea științifică folosește literele 'E' și 'F' pentru a separa mantisa de exponent. Partea exponențială a unei expresii științifice reale trebuie să fie o valoare întreagă cu semn între -37 și +37.

Constantele de tip timer sunt valori cuprinse între 0 și 23h59m59s999ms. Cea mai mică unitate permisă este milisecunda. Valoarea trebuie precedată de caracterele T# sau time#.

O constantă de tip mesaj reprezintă un șir de caractere cuprins între caracterele apostrof. Lungimea unei variabile nu poate depăși 255 caractere. Caracterul apostrof nu poate fi utilizat într-un string. Pentru utilizarea lui cât și pentru caracterele netipăribile se utilizează secvențe care încep cu caracterul \$.

Variabilele din ISaGRAF pot fi de două categorii: *locale* sau *globale*. Numele variabilelor trebuie să înceapă cu o literă și nu trebuie să depășească 16 caractere. Variabilele locale pot fi folosite doar de un program. Variabilele globale pot fi folosite de orice program al aplicației curente.

Variabilele pot avea unul dintre următoarele atribute:

INTERNAL: variabile actualizate de program;  
 INPUT: variabilă conectată la un dispozitiv de intrare;  
 OUTPUT: variabilă conectată la un dispozitiv de ieșire.

Comentariile pot fi introduse liber în limbajele literal acceptate și sunt precedate de caracterele (\* și succedate de caracterele \*).

## 1.1. Crearea unui proiect ISaGRAF

Pentru a crea un proiect se lansează programul ISaGRAF prin executarea unui dublu click pe iconița lui. Se va deschide fereastra Project Management (vezi fig.1), împărțită în două sub-fereestre. În sub-fereastra de sus, sunt listate proiectele deja existente, iar în sub-fereastra de jos sunt scrise detalii despre proiectul selectat.

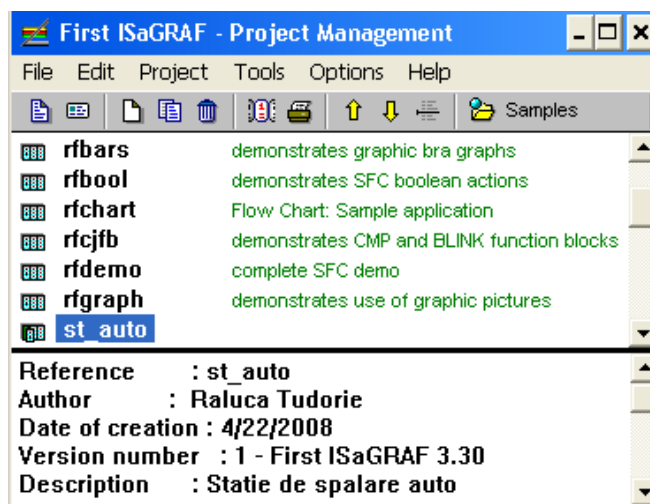


Fig.1. Fereastra ISaGRAF – Project Management

Acțiunile posibile în această etapă sunt: deschiderea/rularea unui proiect existent, modificarea lui sau crearea unui nou proiect.

Pentru crearea unui nou proiect se selectează *File* → *New*; se va deschide fereastra *Create* → *New Project* (vezi fig.2). Se introduce numele dorit al proiectului. Numele nu va depăși 8 caractere și începe cu o literă. Editarea descrierii proiectului se face prin alegerea opțiunii *Project* → *Project descriptor*, dar este indicat ca această operație să se facă la sfârșit pentru a cuprinde informații cât mai complete despre proiect.

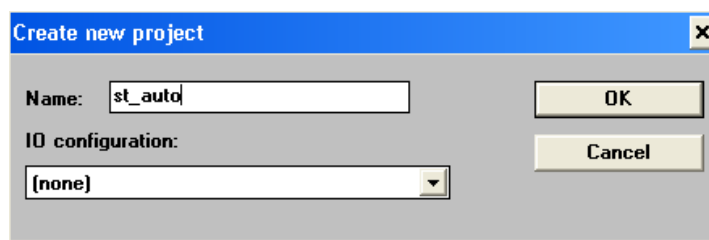


Fig.2. Fereastra Create new project

Pentru editarea programelor din cadrul unui proiect, se va da dublu click în fereastra Project Management pe proiectul dorit. Apare fereastra *Programs* (vezi fig.3).

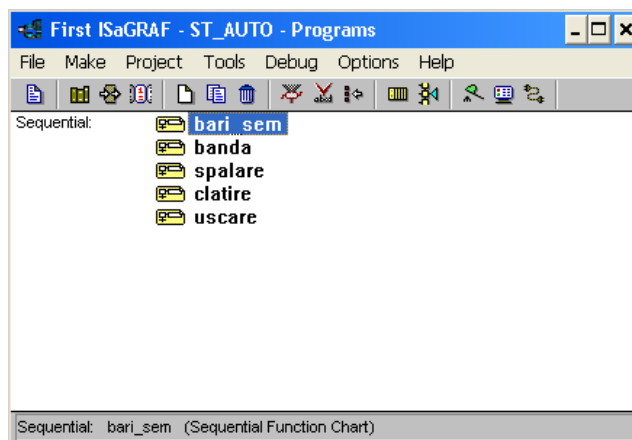


Fig.3. Fereastra Programs

Pentru introducerea unui nou program în proiectul existent, se alege din fereastra *Programs* opțiunea *File* → *New*. Se deschide fereastra *New Program* (vezi fig.4). Trebuie ales numele programului, limbajul în care se editează și locul său în ierarhia programelor (Begin, Sequential, End, Function, Function block sau Child of).

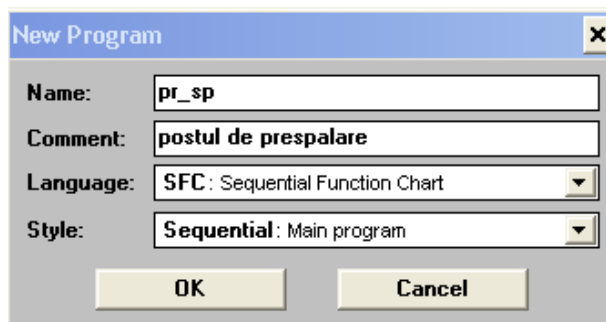


Fig.4. Fereastra New Program

După completarea celor patru câmpuri, se apasă OK, se închide fereastra iar programul apare în ierarhia de programe. Editarea programului se face prin dublu click pe numele acestuia. Înainte de editare, este indicat să se scrie variabilele programului. Editarea variabilelor se face pornind de la fereastra *Programs*, alegând opțiunea *File* → *Dictionary*. Se editează variabilele pentru fiecare categorie. În fig.5. sunt prezentate variabilele booleene. Trebuie ales numele variabilei și atributul.

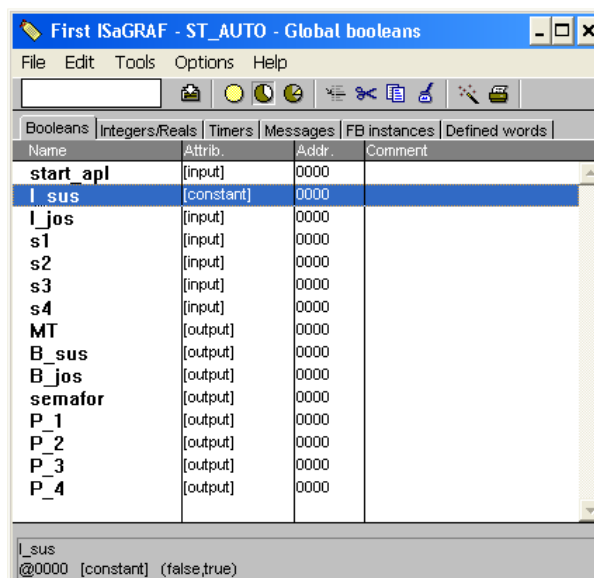


Fig.5. Fereastra Global booleans

În fig.6. este prezentată fereastra unei variabile booleene. În mod identic, arată și fereastra pentru variabilele analogice, doar că modul de afișare poate fi ales în mod diferit pentru cele întregi și reale.

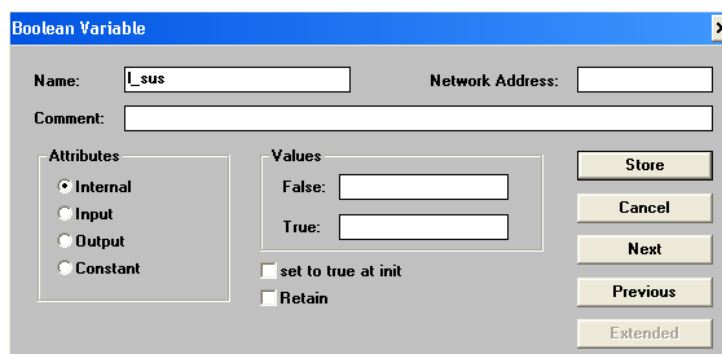


Fig.6. Fereastra Boolean Variable

După editarea variabilelor se trece la scrierea programelor. Lansarea unui editor se face prin dublu click pe program. Se deschide editorul ales la crearea programului SFC (Sequential Function Chart), FBD (Function Block Diagram), LD (Ladder Diagram), ST (Structured Text), IL (Instruction List) sau FC (Flow Chart).



## 1.2. Editarea în Sequential Function Chart

Sequential Function Chart combină editarea grafică cu cea sub formă de text. SFC permite editarea grafului de automatizare cu acțiunile și condițiile de parcurgere a tranzițiilor. Componentele grafice sunt prezentate în fig.7.

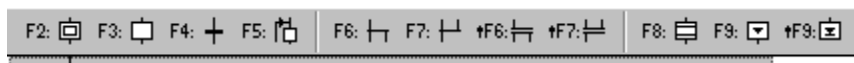


Fig.7. Componentele grafice ale editorului SFC

Componentele existente, în ordinea aparițiilor lor în fig.7 sunt: etapa inițială, etapă, tranziție, salt la etapă, începutul divergenței, sfârșitul divergenței, începutul paralelismului, sfârșitul paralelismului, macroetapă, prima etapă a unei macroetape, ultima etapă a unei macroetape.

Un program SFC este în mod uzual împărțit în două nivele diferite:

- nivelul 1: prezintă graful automatizării și numele etapelor și tranzițiilor
- nivelul 2: se scriu acțiunile și condițiile din tranziții în limbajele ST sau IL.

Pentru a plasa un element, programatorul trebuie să selecteze acel element din bara de componente grafice, apoi va selecta cu mouse-ul zona în care va dori să fie plasat elementul.

Fig.8. prezintă nivelul 1 al programului editat.

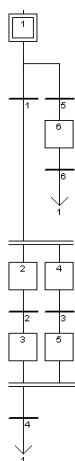


Fig.8. Nivelul 1 al programului SFC

Pentru a edita nivelul 2, se dă dublu-click pe etapă sau pe tranziție. Se va deschide în partea dreaptă a ecranului fereastra numită pentru o etapă *Step GSnnn*, unde nnn este numărul etapei, respectiv pentru o tranziție *Transition GTnnn*, unde nnn este numărul tranziției.

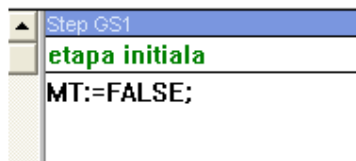


Fig. 9. Nivelul 2 al programului SFC pentru etapa 1

În editarea nivelului 2, mai ales la tranziții se pot folosi două variabile atașate fiecărei etape. Ele sunt actualizate în funcție de starea etapelor din graf. Aceste variabile sunt :

- GSnnn.x – variabilă booleană care reprezintă starea etapei, în timpul execuției programului (1 dacă etapa este activă și 0 în cazul în care etapa este inactivă)
- GSnnn.t – variabilă de tip timer; reprezintă timpul scurs de la activarea etapei.

Limbajul default pentru nivelul 2 de editare a programului SFC este ST.

Următoarele tipuri de acțiuni pot fi asociate unei etape:

- acțiuni booleene : atribuie unei variabile interne sau de ieșire o valoare în funcție de starea etapei căreia i s-a atașat acțiunea.

Sintaxa:

```
<variabila_booleana>;
atribuie valoarea atașată etapei variabilei booleene
/<variabila_booleana> ;
atribuie valoarea negată atasată etapei variabilei booleene
```

- acțiuni memorate: acțiuni de setare sau resetare. Sintaxa:

```
<variabila_booleana>(S);
setează variabila booleană la activarea etapei
/<variabila_booleana>(R);
resetează variabila booleană la activarea etapei
```

- acțiuni de tip puls: se execută o singură dată la activarea etapei, în primul ciclu automat. Sintaxa:

```
ACTION(P):
    (* Instrucțiuni ST *)
END_ACTION ;
```

- acțiuni de tip non-puls: se execută cât timp etapa este activă; dacă este o variabilă booleană va avea forma unor impulsuri scurte, care se repetă la fiecare ciclu. Sintaxa:

```

ACTION(N) :
    (* Instrucțiuni ST *)
END_ACTION ;

```

- acțiuni SFC: pornesc sau opresc un program fiu, în funcție de starea etapei.

```

ACTION(P) :
    GSTART(<Program_fiu>);
END_ACTION ;
ACTION(P) :
    GKILL(<Program_fiu>);
END_ACTION ;

```

### 1.3. Editarea în Function Block Diagram

FBD este un editor grafic. El combină capacitățile grafice cu cele de editare de text. Se pot construi funcții complexe, prin utilizarea blocurilor existente în biblioteca ISaGRAF și legarea acestora între ele.

FBD descrie o funcție între variabilele de intrare și variabilele de ieșire, funcția fiind descrisă de setul de blocuri elementare, care intra în componența diagramei.

Intrările pot fi constante, variabile (interne, de intrare sau de ieșire) sau ieșirile altor blocuri. Ieșirile pot fi variabile (interne sau de ieșire), intrările altor blocuri sau subprograme.

Operatorii sunt reprezentați prin blocuri funcționale dreptunghice. Intrările funcției sunt conectate în partea stângă a blocului în timp ce ieșirile sunt conectate în dreapta. Variabilele (intrările și ieșirile) sunt conectate la blocurile funcționale cu legături logice. Reprezentarea grafică a intrărilor și ieșirilor sunt dreptunghiuri cu colțurile rotunjite (vezi fig.10).

Etichetele și salturile sunt folosite pentru a controla execuția programului. Pentru introducerea etichetelor și a simbolului de salt există pe bara de instrumente obiecte speciale.

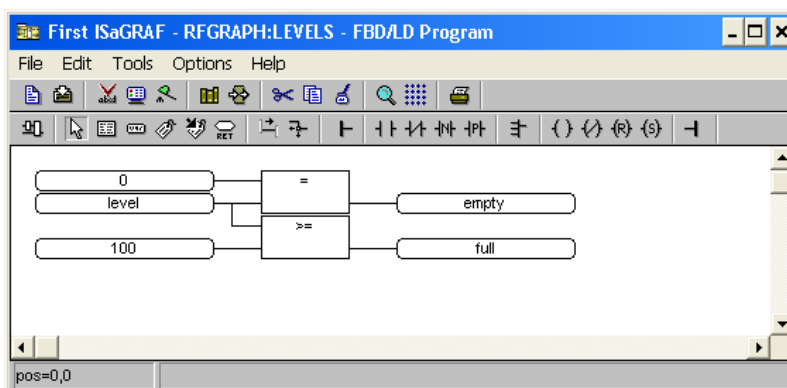


Fig.10. Editorul FBD

#### 1.4. Editarea în Ladder Diagram

Editorul Ladder Diagram este un editor grafic. El permite programatorului să reprezinte grafic ecuații booleene prin combinarea contactelor (variabile de intrare) cu bobine (variabile de ieșire). Modul de realizare a unui program este asemănător cu cel din FBD și nu sunt dificultăți în realizarea lui.

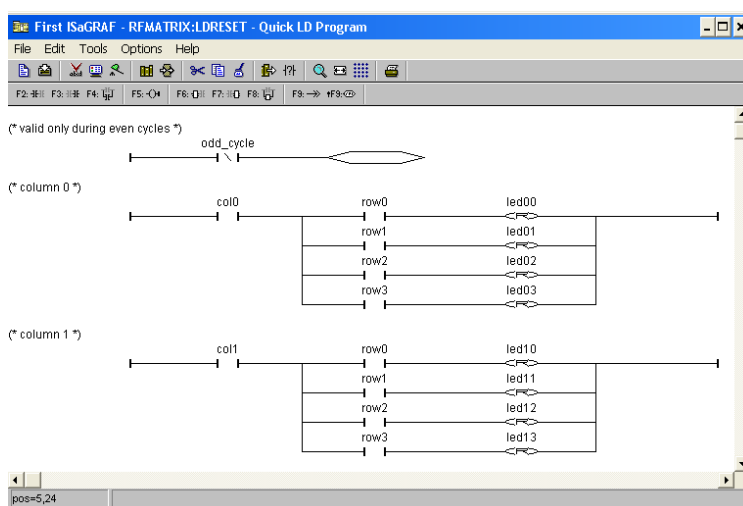


Fig.11. Editorul LD

### 1.5. Editarea în Structured Text și Instruction List

Aceste editoare sunt în mod text și sunt ușor de utilizat dacă este cunoscut limbajul.

În cadrul fiecărui editor există posibilitatea efectuării unei verificări a programelor. Pentru aceasta este necesar să se aleagă opțiunea *Files* → *Verify*. În cazul în care sunt erori acest lucru este semnalat într-o fereastră specială în care se indică codul erorii, coordonatele obiectului care a produs eroarea și un text explicativ. Fereastra trebuie închisă pentru a putea face o nouă verificare după corectarea erorii.

### 1.6. Utilizarea editorului de conexiuni

După elaborarea programului pentru automat este necesar să se stabilească o legătură logică între variabilele de I/O ale programului și canalele de pe automatul programabil. Pentru a realiza acest lucru utilizatorul trebuie să indice și să seteze modulele de pe automat și apoi să lege variabilele de canalele acestor module.

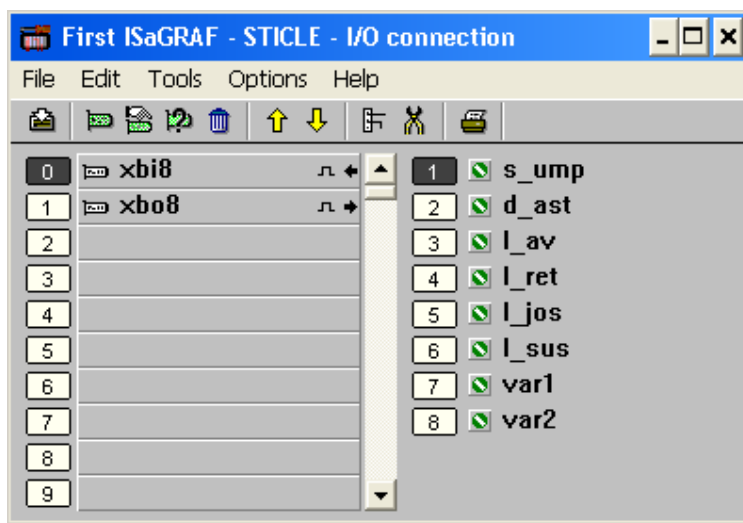


Fig. 12. Fereastra I/O connection

Alegerea, plasarea modulelor și conectarea variabilelor la canalele acestora se face din fereastra *Programs*, prin alegerea opțiunii *Project* → *I/O connection*.

La alegerea acestei opțiuni apare fereastra din fig.12, în care în partea stângă este rack-ul automatului programabil cu sloturile pentru 255 module. Un slot poate fi utilizat pentru introducerea unui modul. Numărul slotului este poziția în care se află față de CPU. Numerotoarea începe de la zero. Adresa logică a unui modul începe de la unu și trebuie setată într-o fereastră separată.

Prin realizarea unui dublu click asupra unui slot selectat al rack-ului se deschide fereastra *Select board/equipment* (fig.13), unde în caseta din stângă se pot alege modulele automatului din biblioteca de module.

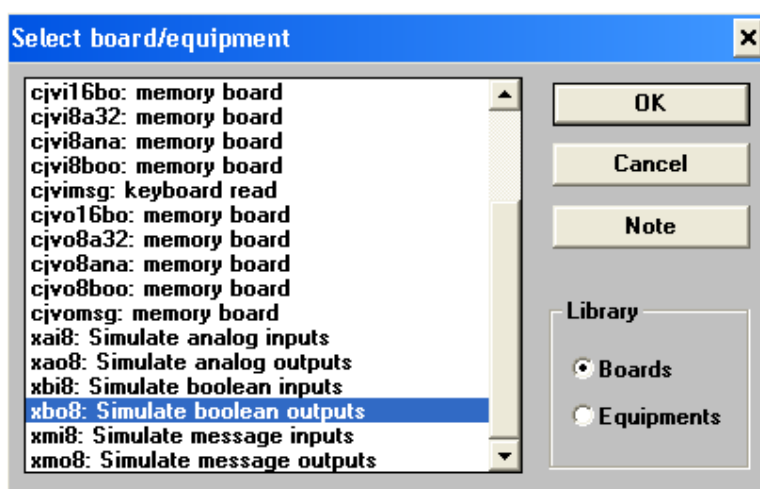


Fig.13. Fereastra *Select board/equipment*

După alegerea modulului corespunzător se apasă butonul OK, iar în fereastra *I/O Connection* va apare modulul respectiv având canalele cu care producătorul l-a echipat.

Urmează să se conecteze variabilele programului la aceste canale. Realizând un dublu click pe un canal va apare fereastra *Connect I/O channel #1* (fig.14), în care în caseta din stânga apar toate variabilele care pot fi conectate la canal.

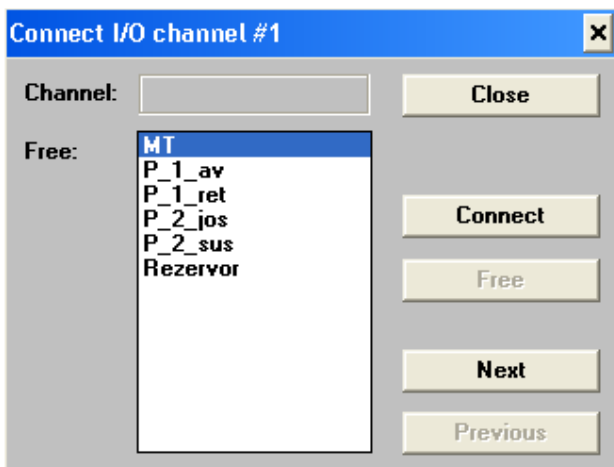


Fig. 14. Fereastra Connect I/O channel

Dacă se apasă butonul *Connect*, variabila respectivă va fi conectată la canalul selectat și de trece la următorul canal. Astfel se pot conecta toate variabilele și apoi se închide fereastra apăsând butonul *Close*. În fereastra *I/O Connection* operațiile făcute sunt reflectate prin plasarea variabilelor în dreptul canalelor alese. În această fereastră trebuie aleasă și adresa logică a modului.

## 1.7. Generarea codului aplicației

Generarea codului aplicației este următoarea etapă care trebuie parcursă înaintea testării și rulării în automat. Mai întâi, în funcție de automatul folosit, se aleg parametrii pentru generarea codului. Acest lucru se face pornind de la fereastra *Programs*, prin alegerea opțiunii *Make* → *Compiler options*. În fereastra care apare (fig.15) se pot alege parametrii pentru generarea codului. Se poate alege generarea de cod pentru simulare și pentru unități centrale cu procesor Motorola sau Intel. De asemenea, se pot bifa diferite opțiuni de optimizare a codului. De fiecare dată când se bifează mai multe opțiuni vor fi generate mai multe coduri.

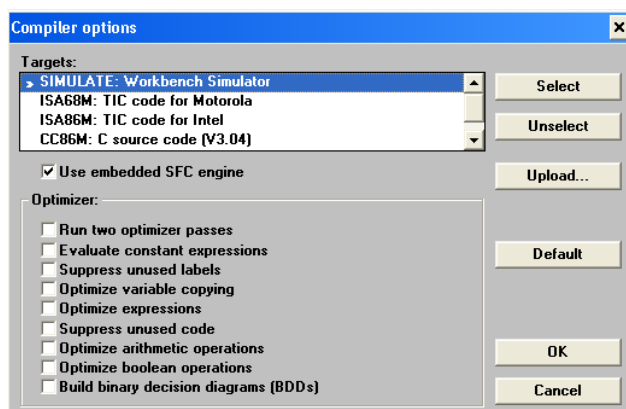


Fig. 15. Fereastra Compiler options

## 1.8. Verificarea prin simulare a programului

Pentru a realiza o verificare offline a programului elaborat se poate folosi opțiunea *Debug* → *Simulate* din fereastra *Programs*. Evident că trebuie ca, la generarea codului, să fie bifată și opțiunea de generare a codului pentru simulare.

Această comandă deschide debuggerul în modul simulare. Se rulează simultan un simulator complet al automatului (fig.16). Simulatorul nu poate fi pornit dacă nu a fost generat codul pentru simulare și dacă ferestrele de editare, de generare de cod și *I/O connection* sunt deschise.

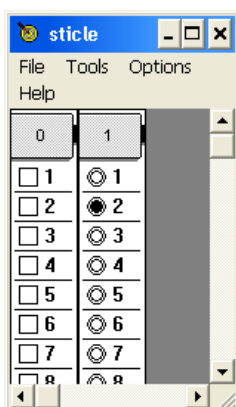


Fig.16. Fereastra de rulare a programului



Aceste ferestre pot fi deschise după pornirea simulatorului pentru a putea urmări evoluția programului. De exemplu în fereastra editorului SFC se poate urmări poziția jetonului în graf. Utilizarea simulatorului este ușoară și intuitivă. Intrările booleene sunt simulate prin butoane, care pot fi apășate, iar ieșirile booleene sunt simulate prin leduri care devin de culoare roșie dacă ieșirea respectivă este true. O intrare analogică este simulată printr-un câmp de text, în care se introduce valoarea intrării în zecimal sau hexazecimal. Dacă se execută un click pe acel câmp se poate introduce o valoare nouă. O ieșire analogică este un câmp numeric de ieșire în care valoarea variabilei este afișată. La fel sunt tratate de către simulator și variabilele de tip Messages.

Se poate seta simulatorul să afișeze, lângă variabila simulată și numele acesteia.

## 1.9. Setarea legăturii seriale

Pentru realizarea încărcării în automat a programului este nevoie, pe lângă cablul de legătură dintre PC și automat, care trebuie achiziționat odată cu automatul și de setarea portului serial al calculatorului pe care rulează ISaGRAF.

Pentru realizarea acestui lucru, din fereastra *Programs* se alege opțiunea *Debug* → *Link setup*. În fereastra care apare (fig.17) se pot seta portul de comunicație, viteza și paritatea.

Fig.17. Fereastra PC-PLC link parameters

### 1.10. Utilizarea debuggerului grafic

ISaGRAF include un debugger grafic și simbolic. Dacă din fereastra *Programs* se alege opțiunea *Debug* → *Debug* se lansează debuggerul, care controlează încărcarea aplicației în automat. Debuggerul comunică cu automatul prin intermediul legăturii seriale. Fereastra debuggerului conține toate comenzile și informațiile pentru controlul aplicației. În această fereastră se prezintă starea aplicației și informații asupra execuției acesteia.

Sunt posibile următoarele stări:

- *Logging...* - Stare în care debuggerul stabilește comunicația cu sistemul țintă
- *Disconnected* - Stare în care debuggerul nu poate comunica cu aplicația. În acest caz trebuie verificați parametrii legăturii seriale și cablul de legătură.
- *No target application* – Stare în care legătura este stabilită, dar aplicația nu este încărcată. Pentru încărcare se va folosi *Files* → *Download*.
- *Target application inactive* – Stare în care legătura este stabilită, aplicația este încărcată, dar nu este activată. Pentru activare se va folosi *Files* → *Activate PLC*.
- *Target application active* – Stare în care legătura este stabilită, aplicația este încărcată și ea este activată.
- *RUN* – Aplicația este în modul *Real Time*.
- *STOP* – Aplicația este în modul *Cycle to cycle*. Se așteaptă comenzi pentru executarea unui ciclu.
- *Fatal Error* – Aplicația este oprită din cauza unei erori fatale.

Când se încarcă aplicația trebuie realizată atât încărcarea codului cât și a simbolurilor aplicației.

O aplicație trebuie să se afle în modul *STOP* pentru a putea încărca o versiune modificată sau o altă aplicație.

## Problema 1: Controlul unei macarale

### 1. Descrierea sistemului și a procesului

Sistemul este format dintr-o macara acționată de două motoare, fiecare cu câte două senzuri de rotație și prevăzute cu patru limitatoare de cursă.

Aplicația constă în controlul acestei macarale care trebuie să realizeze cele 2 cicluri de mișcare reprezentate în figura 1.1. Inițial, macaraua se găsește în poziția de repaus 1. La apăsarea butonului de pornire, macaraua pornește și realizează ciclul 1 de mișcare, până ajunge în poziția de repaus 2, unde rămâne pentru un anumit timp cunoscut (3s), înainte de a porni ciclul 2; când ajunge în poziția de repaus 1, macaraua se va opri. Un nou ciclu va porni după reapăsarea butonului de pornire.

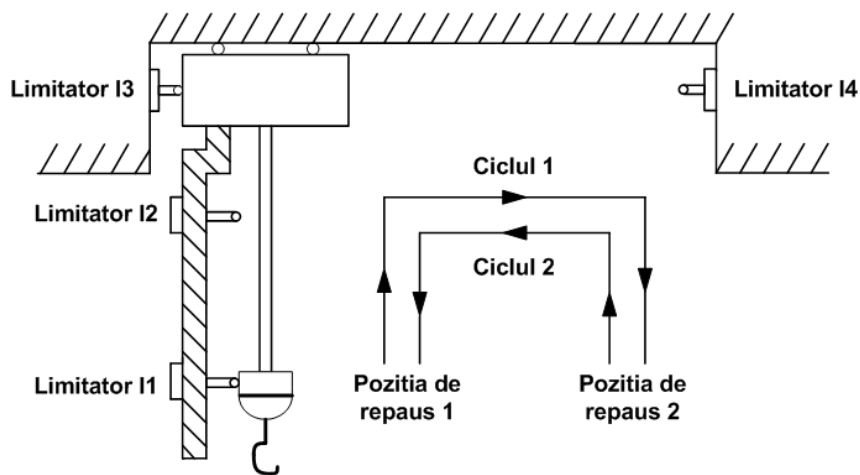


Fig.1.1. Controlul unei macarale

*Elemente de execuție:*

- două motoare cu două senzuri de rotație, unul pentru mișcarea orizontală și unul pentru cea verticală;

*Elemente de măsură:*

- patru limitatoare de cursă;
- un buton de pornire.

## 2. Soluția de automatizare

### □ Implementarea în mediul ISaGRAF

Prima soluție pentru controlul acestei aplicații o reprezintă un automat programabil de tip PEP Smart pentru care s-a dezvoltat un proiect ISaGRAF ce cuprinde un program principal secvențial.

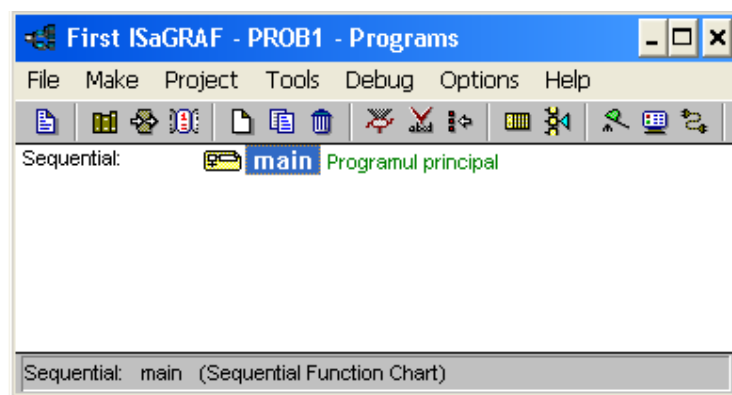


Fig.1.2. Structura proiectului ISaGRAF

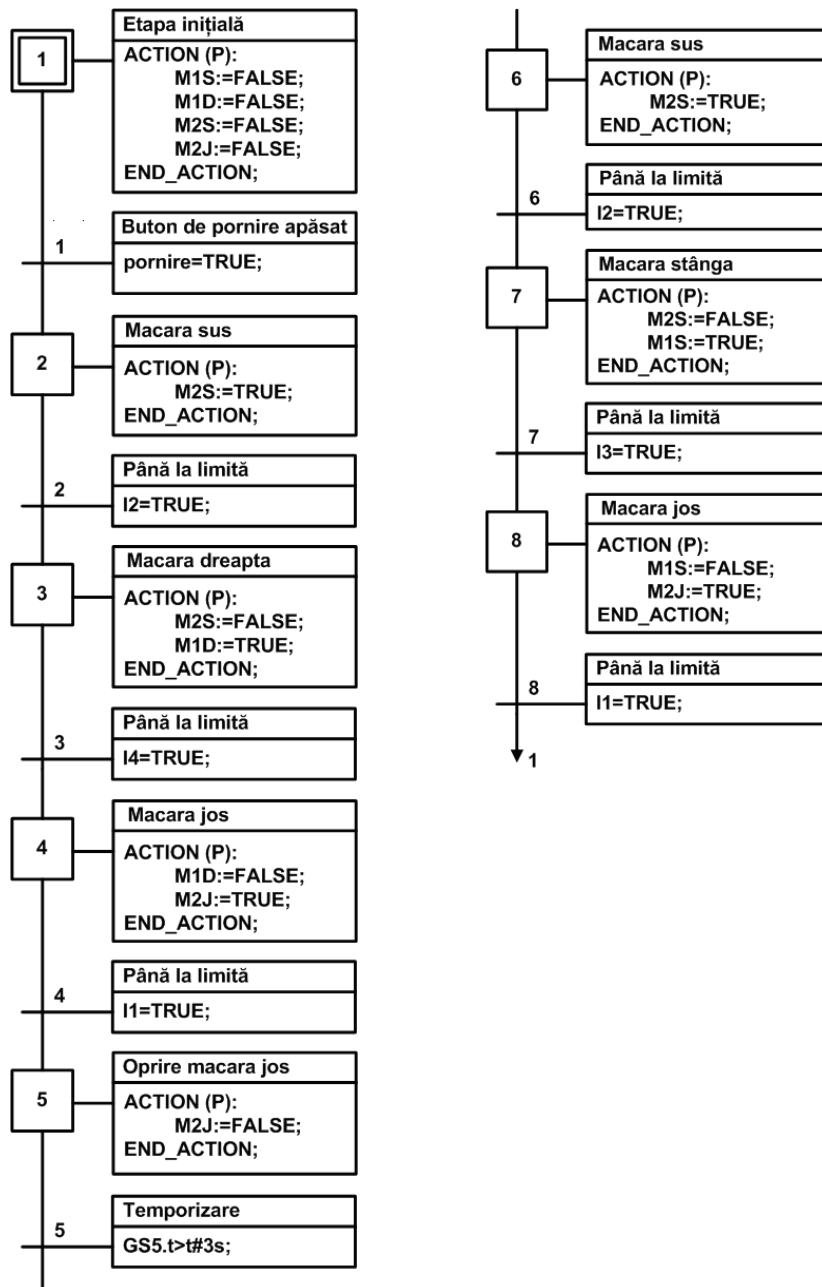
*Dicționarul de variabile globale:*

- *Variabile de intrare booleene:*
  - **pornire**: buton de pornit ciclul;
  - **I1**: limitator jos;
  - **I2**: limitator sus;
  - **I3**: limitator stânga;
  - **I4**: limitator dreapta.
- *Variabile de ieșire booleene:*
  - **M1S**: acționare motor 1 stânga;
  - **M1D**: acționare motor 1 dreapta;
  - **M2S**: acționare motor 2 sus;
  - **M2J**: acționare motor 2 jos.

27

Problema 1- Controlul unei macarale

Program main:



### □ Implementarea în limbajul Ladder Diagram

Pentru controlul acestei aplicații s-a ales un automat programabil de tip Allen Bradley. Programul de tip Ladder Diagram este construit pe baza diagramei logice din proiectul ISaGRAF prezentat anterior.

Asocierea intrărilor și ieșirilor fizice cu biți din regiștrii de intrare/ieșire este prezentată în tabelul 1.1:

*Tabelul 1.1.*

| Intrare fizică | Adresă internă | Ieșire fizică | Adresă internă |
|----------------|----------------|---------------|----------------|
| Pornire        | I:1/1          | M1S           | O:3/1          |
| I1             | I:1/2          | M1D           | O:3/2          |
| I2             | I:1/3          | M2S           | O:3/3          |
| I3             | I:1/4          | M2J           | O:3/4          |
| I4             | I:1/5          |               |                |

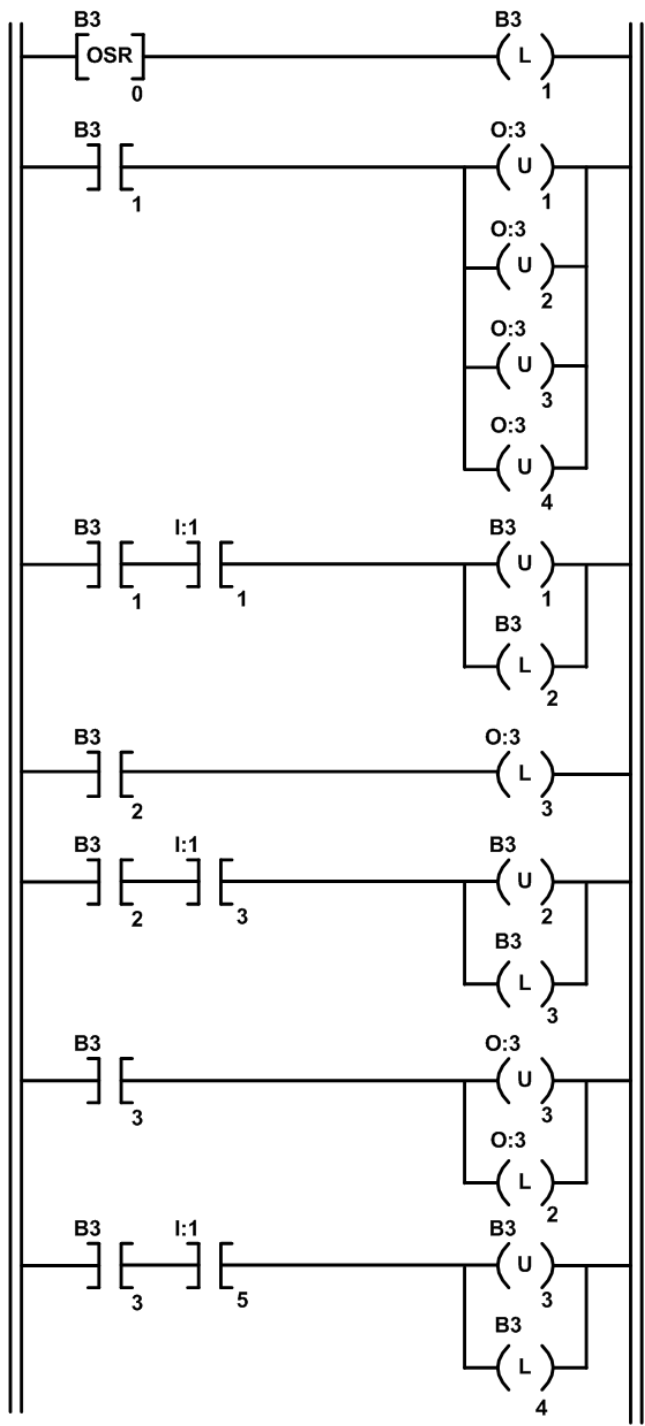
Asocierea etapelor cu biți din fișierul de bit B3 și alegerea fișierului de timer este prezentată în tabelul 1.2:

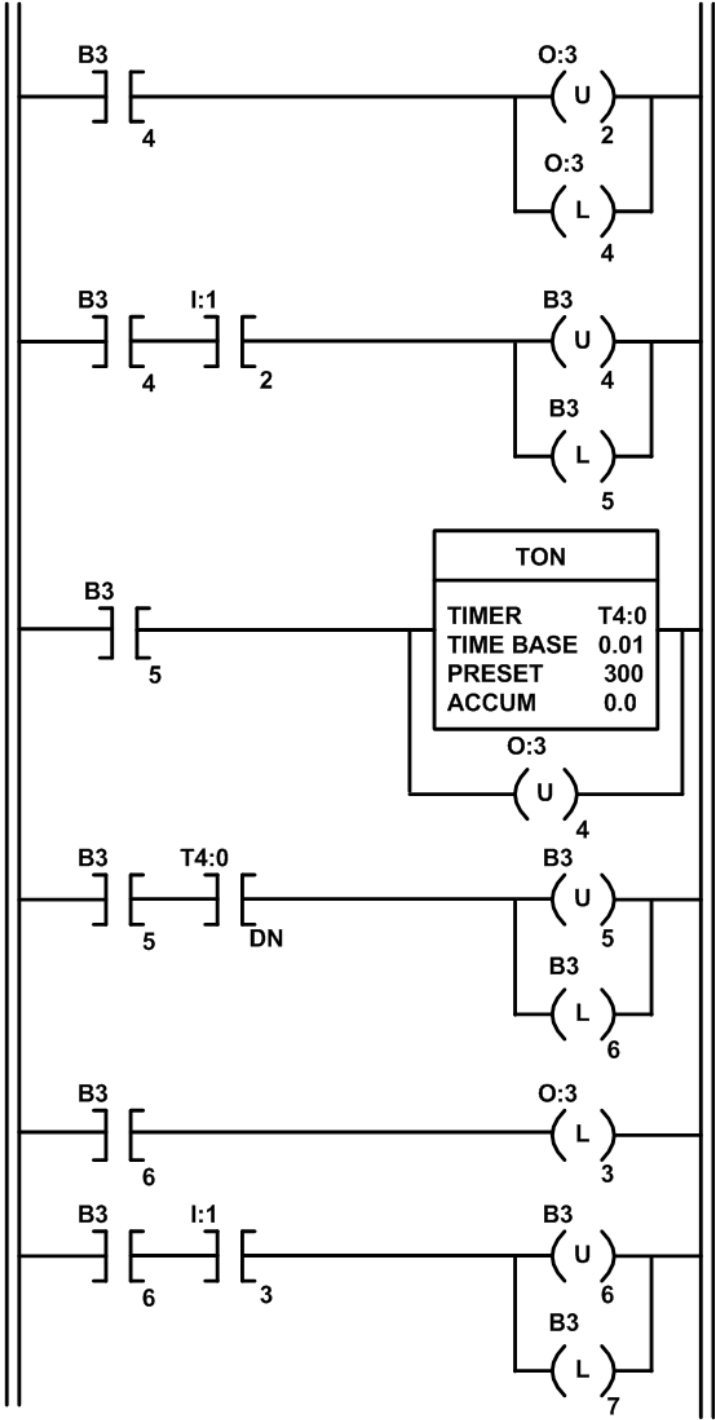
*Tabelul 1.2.*

| Etapă | Adresa bit | Temporizare   | Fișier de timer |
|-------|------------|---------------|-----------------|
| 1     | B3/1       | Temporizare 1 | T4:0            |
| 2     | B3/2       |               |                 |
| 3     | B3/3       |               |                 |
| 4     | B3/4       |               |                 |
| 5     | B3/5       |               |                 |
| 6     | B3/6       |               |                 |
| 7     | B3/7       |               |                 |
| 8     | B3/8       |               |                 |

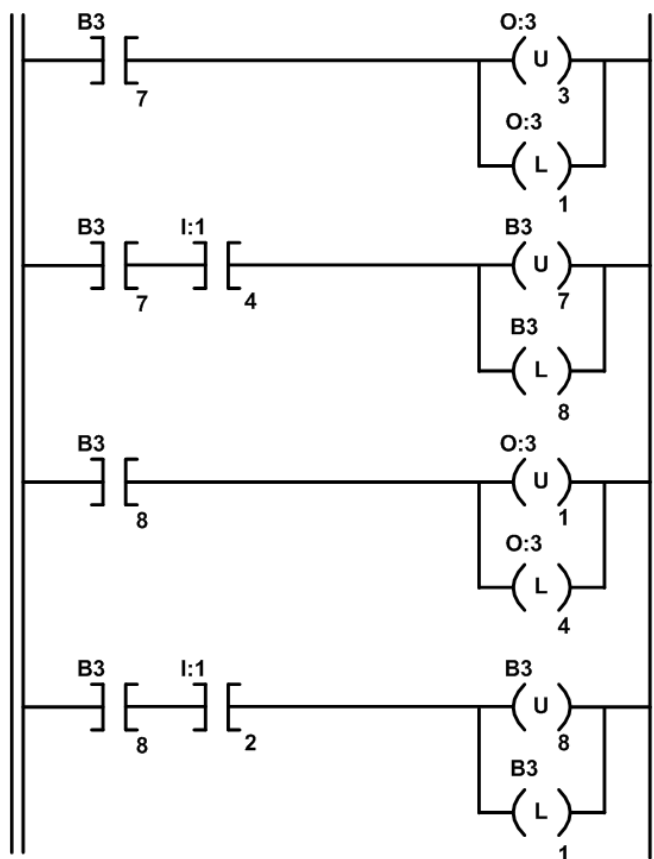
Diagrama Ladder este prezentată în continuare:

Problema 1- Controlul unei macarale









## Comentarii:

- În diagrama SFC se observă realizarea temporizării folosind variabila sistem GS5.t, care conține timpul scurs de la activarea etapei asociate. Această metodă de măsurare a timpului poate fi folosită doar în situația când o etapă trebuie să fie activă pentru o anumită perioadă de timp, în alte situații fiind necesară folosirea unor variabile globale de tip timer și gestionarea lor de către funcțiile TSTART și TSTOP.
- În diagrama Ladder se observă că ramurile de tip 1 și 2 asociate unei etape au fost scrise una după alta, pentru această aplicație programul funcționând corect, există însă situații când ordinea de scriere a ramurilor trebuie să fie diferită (de multe ori se scriu primele ramurile de tip 1 pentru fiecare etapă, apoi ramurile de tip 2).

## Problema 2: Comanda mișcării oscilatorii a unui mobil

### 1. Descrierea procesului

Un mobil alunecă pe un șurub mișcat de un motor acționat de două contactoare (MD – dreapta și MS – stânga), între 2 limitatoare de cursă. Mobilul trebuie să realizeze o mișcare oscilatorie continuă din momentul în care se primește comanda (impuls) de la butonul M. Un impuls de la butonul P trebuie să oprească motorul, dar nu imediat, ci la finalul mișcării începute. Un impuls de la butonul E produce o retragere imediată a mobilului în poziția de origine, iar sistemul se mai poate pune în mișcare doar apăsând butonul R.

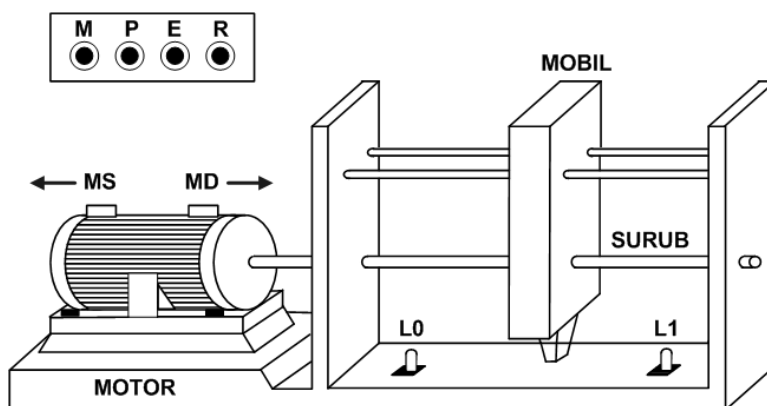


Fig. 2.1. Mișcarea oscilatorie a unui mobil

*Element de execuție:*

- un motor cu două sensuri de rotație;

*Elemente de măsură:*

- două limitatoare de cursă;
- 4 butoane.

### 2. Soluția de automatizare

Pentru controlul acestei aplicații se alege un automat programabil de tip PEP Smart PLC pentru care se dezvoltă un proiect ISaGRAF. Proiectul

conține două programe, un program principal și un program fiu, numit Osc. Programul Osc este pornit și oprit de către programul principal și este responsabil cu realizarea mișcării oscilatorii și sesizarea apăsării butoanelor, conform specificațiilor aplicației.

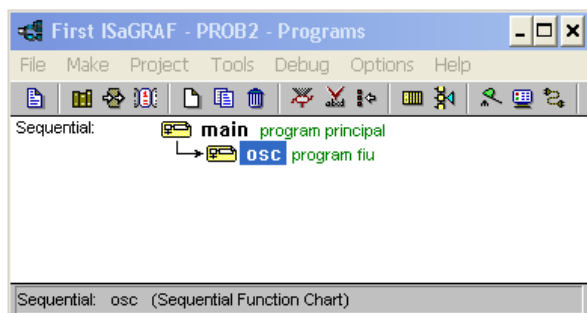


Fig. 2.2. Structura proiectului ISaGRAF

Dicționarul de variabile globale:

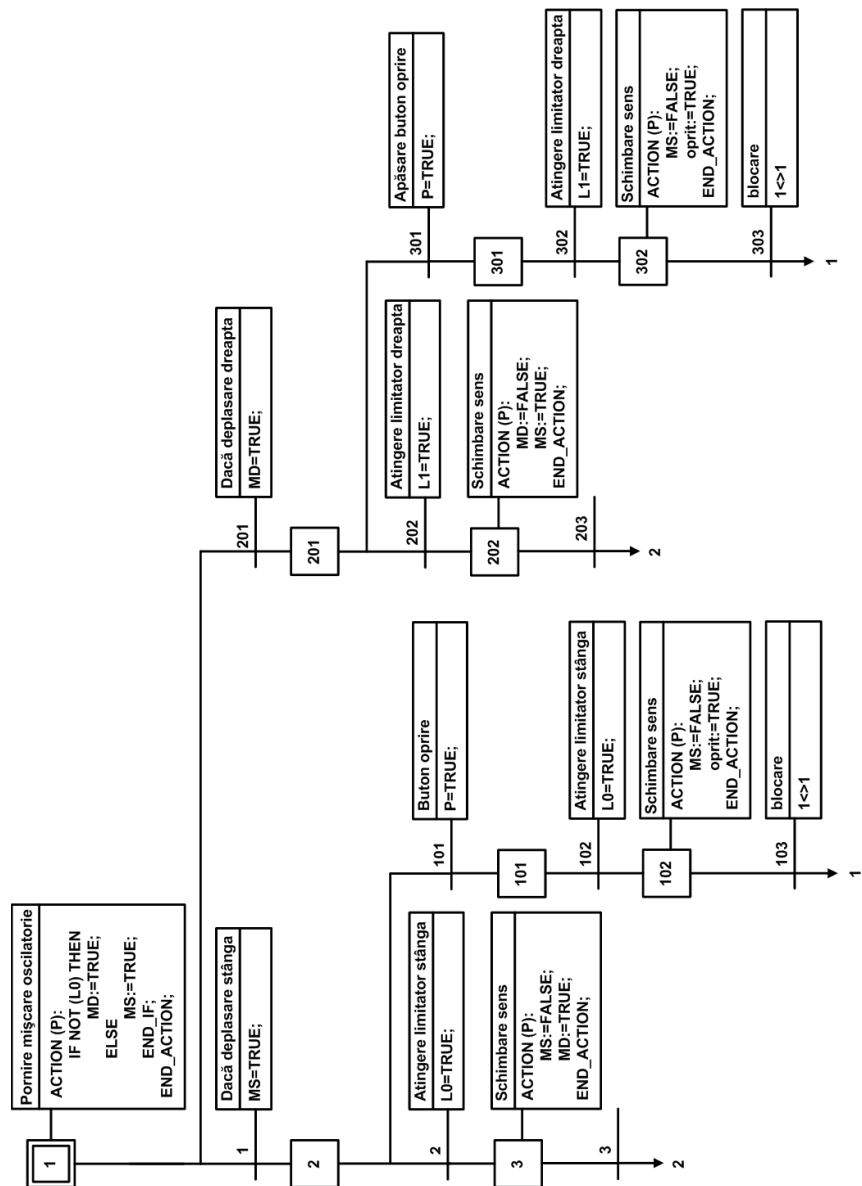
- *Variabile de intrare digitale:*
  - **M**: buton de pornire mișcare oscilatorie;
  - **P**: buton de oprire motor;
  - **E**: buton de retragere în poziția de origine;
  - **R**: buton de repornire;
  - **L0**: limitator stânga;
  - **L1**: limitator dreapta.
- *Variabile de ieșire digitale:*
  - **MS**: comandă motor stânga;
  - **MD**: comandă motor dreapta.
- *Variabile interne de tip boolean:*
  - **oprit**: are valoarea TRUE când mișcarea oscilatorie este oprită;
  - **v\_redgeM**: necesar funcției REDGE;
  - **v\_redgeR**: necesar funcției REDGE.



35

Problema 2 - Comanda mișcării oscilatorie a unui mobil

Program osc:



*Comentarii :*

- Funcția REDGE este folosită pentru detectarea impulsurilor produse prin apăsarea butoanelor;
- Comunicația între programe este realizată prin intermediul variabilei oprit;
- Programul principal oprește execuția programului fiu Osc atunci când variabila oprit are valoare TRUE sau mișcarea este întreruptă de apăsarea butonului E;
- Programul putea fi realizat și fără un program fiu, dar diagrama s-ar fi complicat foarte mult, din cauza necesității de a testa în fiecare etapă starea butonului E.

*Propunere:*

- Să se modifice programul în condițiile în care la apăsarea butonului de oprire, ciclul se încheie totdeauna când mobilul ajunge prima dată în partea dreaptă.

## Problema 3: Detecția și expulzarea automată a sticlelor fără dop

### 1. Descrierea procesului

Una dintre fazele de producție într-o linie de îmbuteliere constă în așezarea unui dop, ca urmare a încheierii secvenței de umplere. O dată astupate, sticlele se deplasează pe banda 1. Scopul aplicației este detectarea și evacuarea sticlelor care ies din faza de închidere fără dopul corespunzător. În plus, dacă sunt rejectate mai mult de 3 sticle consecutive, trebuie activată o alarmă. Repornirea ciclului se face prin apăsarea butonului Pc. Pentru detecția sticlei defecte se conjugă acțiunile unui senzor inductiv, care detectează prezența dopului și un echipament fotoelectric care semnalează prezența unei sticle.

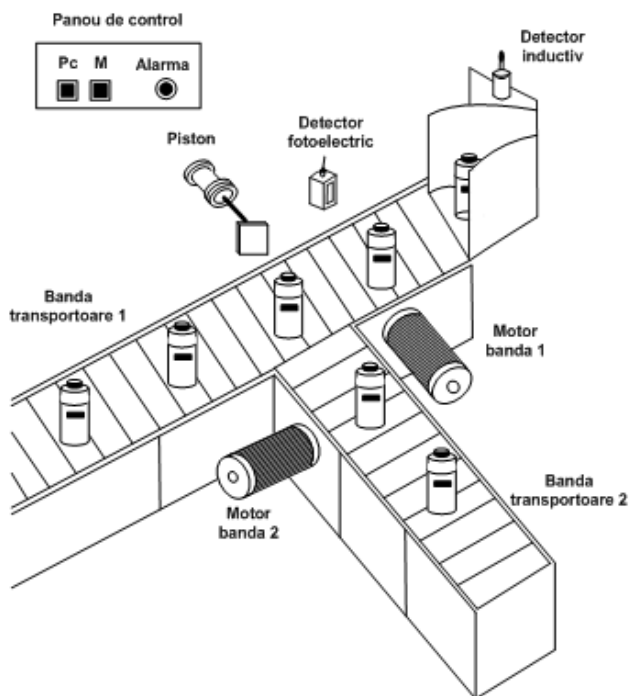


Fig.3.1. Detecția și expulzarea sticlelor fără dop

Procesul este inițiat prin apăsarea butonului M, care determină pornirea benzii transportoare 1. Atunci când se detectează o sticlă fără dop, este oprită banda 1 și este pornită banda transportoare 2 (dacă era oprită). În momentul în care sticla fără dop se găsește în zona de expulzare, este activat mecanismul de expulzare (un piston ce poate avansa și se poate retrage între două limite). Banda 1 va fi repornită în momentul în care sticla fără dop nu se mai găsește în zona de expulzare (practic semnalul transmis de detectorul fotoelectric are valoarea logică fals). Banda 2 va fi oprită după 5 secunde de la începerea expulzării ultimei sticle.

*Elemente de execuție:*

- două motoare care acționează două benzi transportoare;
- un piston folosit la expulzarea sticlelor fără dop.

*Elemente de măsură:*

- un detector inductiv pentru dopuri;
- un detector fotoelectric pentru sticle.

## 2. Soluția de automatizare

Pentru controlul acestei aplicații se alege un automat programabil de tip PEP Smart pentru care se dezvoltă un proiect ISaGRAF, cu un singur program principal dar cu două secțiuni programate: secțiunea secvențială și secțiunea de sfârșit ca în fig. 3.2.

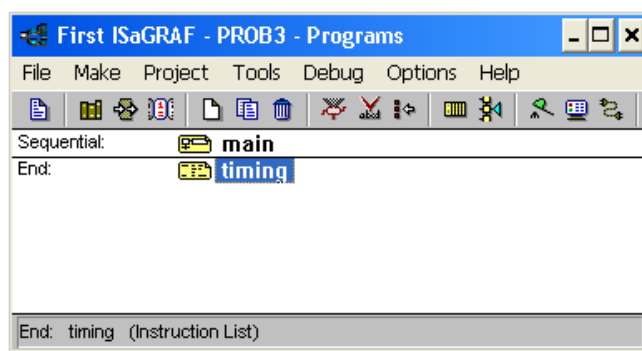


Fig.3.2. Structura proiectului ISaGRAF



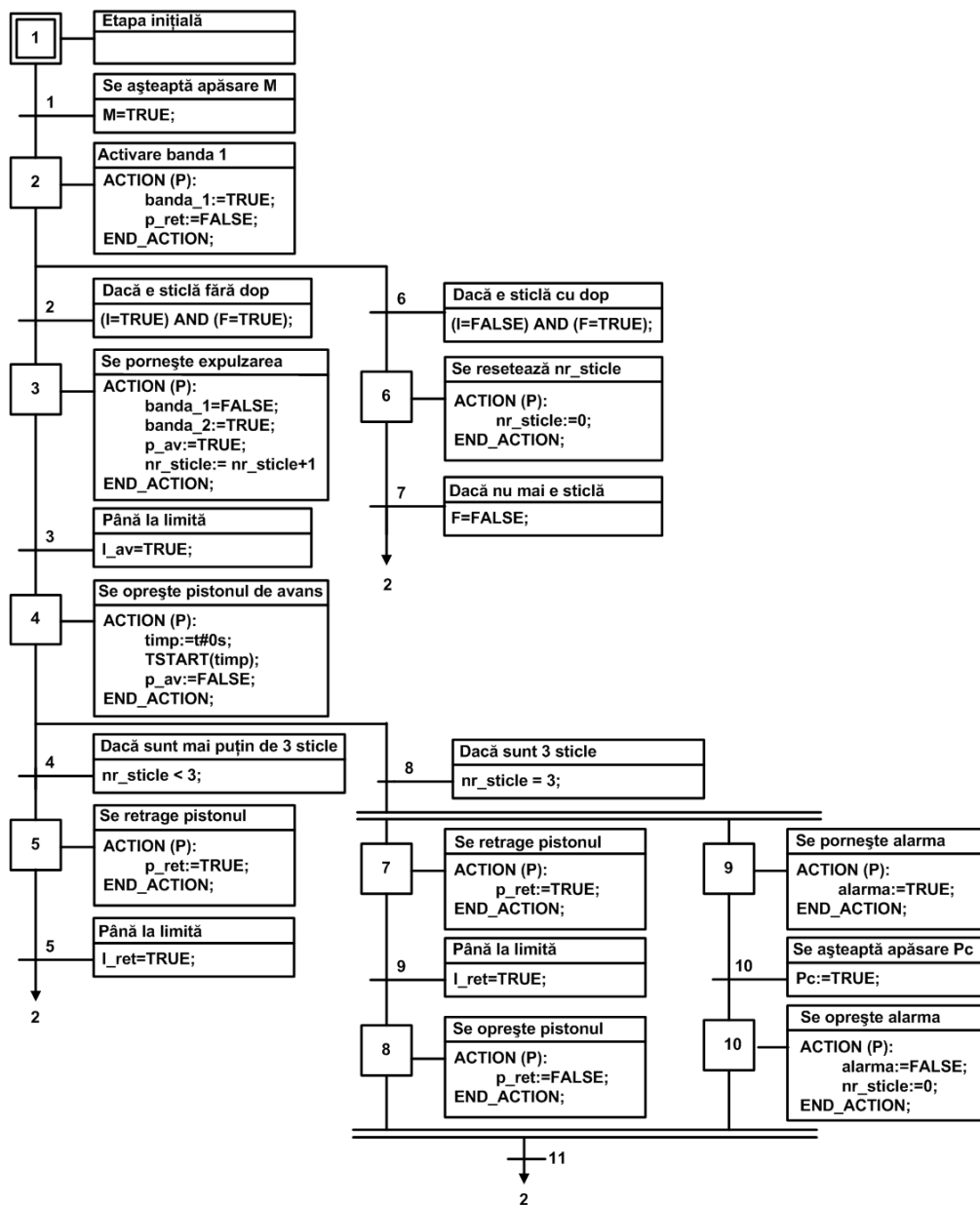
*Dicționarul de variabile globale:*

- *Variabile de intrare booleene:*
  - **M:**     buton de pornire;
  - **I:**     detector inductiv;
  - **F:**     detector fotoelectric;
  - **Pc:**    buton de reponire a ciclului;
  - **l\_av:**  limita de avans;
  - **l\_ret:**  limita de retragere.
- *Variabile de ieșire booleene:*
  - **banda\_1:**    comandă pornire/ oprire bandă 1;
  - **banda\_2:**    comandă pornire/ oprire bandă 2;
  - **alarmă:**     comandă alarma;
  - **p\_av:**        comandă avansul pistonului;
  - **p\_ret:**        comandă retragerea pistonului.
- *Variabile interne de tip întreg:*
  - **nr\_sticle:**    contorizează numărul de sticle fără dop expulzate consecutiv.
- *Variabile interne de tip Timer:*
  - **timp:**         folosit la contorizarea celor 5 secunde de activare a conveiorului 2.

### Programul Timing:

```
IF (timp>t#5s) THEN
    banda_2:=FALSE;
    TSTOP(timp);
    timp:=t#0s;
END_IF;
```

### Programul Main:



*Observații:*

- secțiunea de end (care conține programe ciclice) este necesară deoarece timpul trebuie verificat la fiecare ciclu automat astfel încât banda 2 să poată fi oprită în orice moment dacă perioada ei de activare a expirat;
- contorizarea timpului este făcută cu ajutorul funcțiilor TSTART și TSTOP;
- dacă o sticlă trebuie să fie expulzată în timp ce o alta se găsește pe conveiorul 2, timer-ul este resetat astfel încât contorizarea timpului repornește de la 0 pentru a asigura și evacuarea acestei ultime sticle;
- dacă numărul de sticle evacuate consecutiv este 3, atunci vor începe două secvențe de acțiuni ce trebuie să aibă loc simultan și anume:
  - retragerea pistonului
  - activarea – dezactivarea alarmei.

Abia după încheierea ambelor secvențe procesul poate reporni, situație reprezentată printr-un paralelism în cadrul diagramei.

*Propuneri:*

- Să se construiască o diagramă Ladder pentru un automat de tip Allen Bradley, care să controleze acest proces.
- Să se modifice proiectul ISaGRAF dacă aplicația se schimbă astfel:
  - Dacă pe bandă nu mai vin sticle timp de 2 minute, banda 1 va fi oprită, se activează alarma, repornirea procesului având loc la apăsarea butonului Pc.

## Problema 4: Stație automată de spălat autovehicule

### 1. Descrierea procesului

Scopul proiectării acestui sistem de control îl reprezintă automatizarea unei stații de spălat autovehicule. Stația are următoarea structură:

- bandă transportoare a mașinilor;
- patru posturi de lucru prevăzute cu senzori fotoelectrici de prezență a unei mașini și cu echipamente de acționare specifice postului respectiv (prespălare, spălare cu detergent, clătire și uscare);
- o barieră acționată de un motor cu două senzuri de rotație și două limitatoare de cursă;
- un semafor cu 2 culori, verde și roșu.

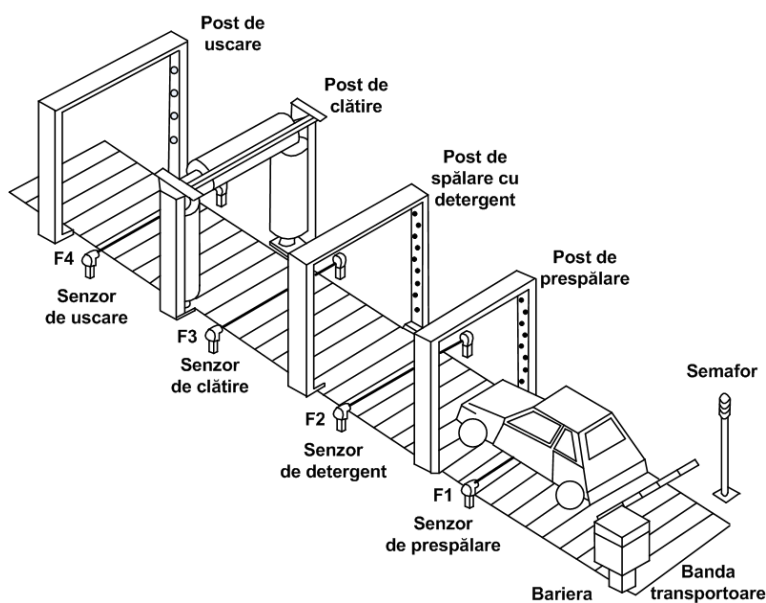


Fig.4.1. Stație automată de spălat autovehicule

Inițial banda este oprită, bariera ridicată și semaforul verde, neexistând mașini pe bandă. În momentul în care o mașină apare în dreptul postului 1, banda va trebui să pornească pentru a transporta mașinile prin

cele 4 posturi de lucru. Banda va fi oprită când nu mai există mașini pe bandă. Bariera va fi coborâtă iar semaforul va deveni roșu atâta timp cât postul 1 este ocupat. Cât timp o mașină trece printr-un post, echipamentul postului respectiv trebuie să fie activ.

*Elemente de execuție:*

- motorul benzii transportoare;
- motorul barierei cu două sensuri de rotație;
- un semafor;
- patru posturi de lucru.

*Elemente de măsură:*

- cinci celule fotoelectrice;
- două limitatoare de cursă ale barierei.

## 2. Soluția de automatizare

Pentru controlul acestei aplicații se alege un automat programabil de tip PEP Smart pentru care se dezvoltă un proiect ISaGRAF, proiect ce constă din șase programe editate în SFC care rulează în paralel. Structura proiectului ISaGRAF este prezentată în fig.4.2.

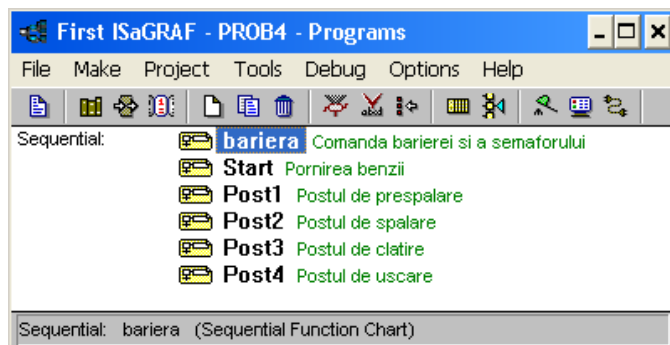


Fig.4.2.Structura proiectului ISaGRAF

*Dicționarul de variabile:*

- *Variabile de intrare booleene:*
  - **limita\_sus**: limitator de cursă sus pentru barieră;
  - **limita\_jos**: limitator de cursă jos pentru barieră;
  - **foto\_1**: fotocelula postului 1;

- **foto\_2:** fototelula postului 2;
- **foto\_3:** fototelula postului 3;
- **foto\_4:** fototelula postului 4.

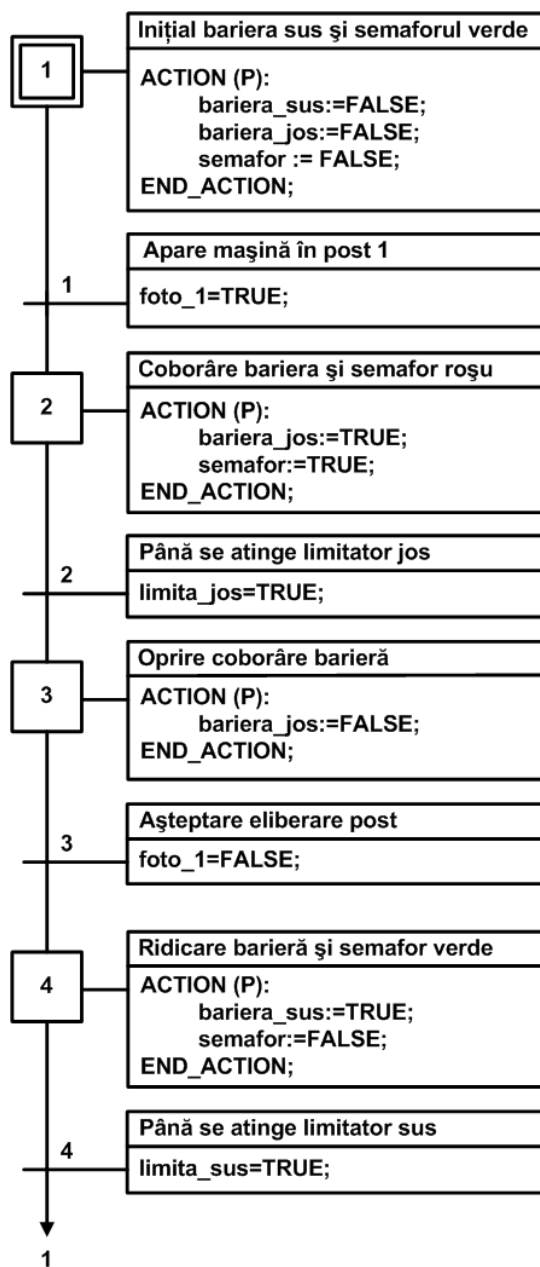
▪ *Variabile de ieșire booleene*

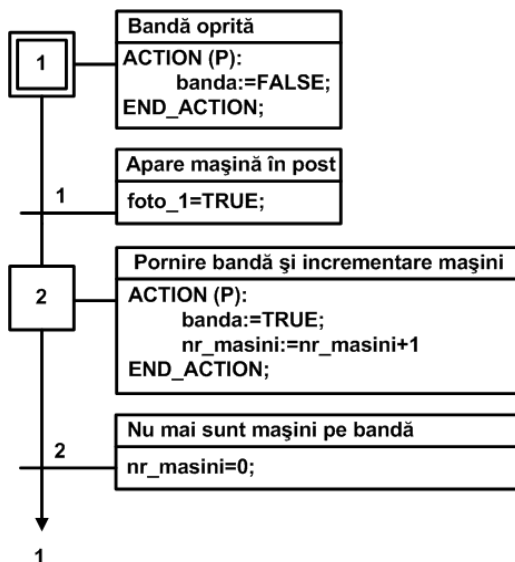
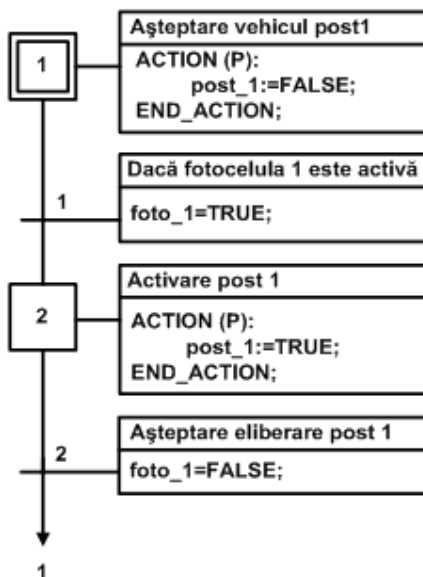
- **banda:** comandă pornire/oprire bandă;
- **bariera\_sus:** comandă ridicare barieră;
- **bariera\_jos:** comandă coborâre barieră;
- **semafor:** comandă activare/ dezactivare semafor;
- **post\_1:** comandă activare/ dezactivare post prespălare;
- **post\_2:** comandă activare/ dezactivare post spălare;
- **post\_3:** comandă activare/ dezactivare post clătire;
- **post\_4:** comandă activare/ dezactivare post uscare.

▪ *Variabile interne integer*

- **nr\_masini:** numărul de mașini existente în stația de spălare.

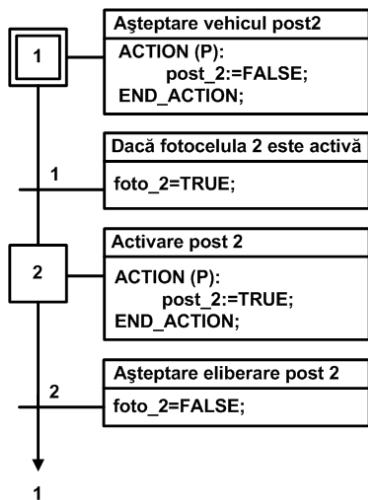
Program bariera:



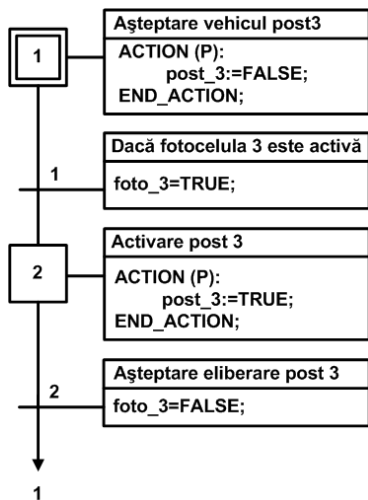
**Program start:****Program Post1:**



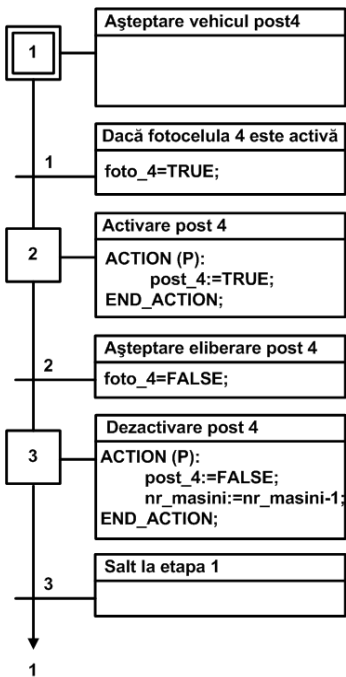
**Program Post2:**



**Program Post3:**



**Program Post4:**



*Observații:*

- Toate programele dezvoltate sunt independente și rulează în paralel, fapt ce ușurează foarte mult înțelegerea și realizarea programului.
- Determinarea faptului că mai există sau nu mașini pe bandă s-a făcut cu ajutorul unei variabile de tip integer, **nr\_mașini**, o altă abordare de genul testarea tuturor senzorilor de prezență fiind incorectă, putând exista mașini pe bandă dar între posturi, deci toți senzorii de prezență putând fi inactivi la un moment dat, deși există mașini pe bandă.

*Propunere:*

- Să se modifice proiectul corespunzător situației în care pentru un timp *Tasteptare* nu mai urcă nicio mașină pe bandă, situație în care bariera va fi coborâtă iar reluarea procesului se poate face numai prin apăsarea unui buton, introdus suplimentar în sistem.

## Problema 5: Elevator clasificator de pachete

### 1. Descrierea procesului

Pe o bandă transportoare vin două tipuri de pachete (mic și mare). Tipul pachetului este determinat de un cântar, în final pachetele fiind sortate și transportate în direcții diferite, în funcție de tipul pachetului. Componentele sistemului sunt:

- patru benzi transportoare acționate de motoare cu un singur sens de rotație;
- un cântar al cărui traductor generează un semnal analogic între 4 și 20 mA, proporțional cu o gamă de valori ale greutății între 0 și 100 kg;
- un piston (cilindru) C dotat cu un plan elevator pentru ridicarea pachetelor, poate avansa și se poate retrage între 2 limite;
- două pistoane A și B pentru evacuarea pachetelor, pot avansa și se pot retrage între 2 limite;
- doi senzori de prezență pachet F0 și F1 poziționați ca în fig. 5.1.

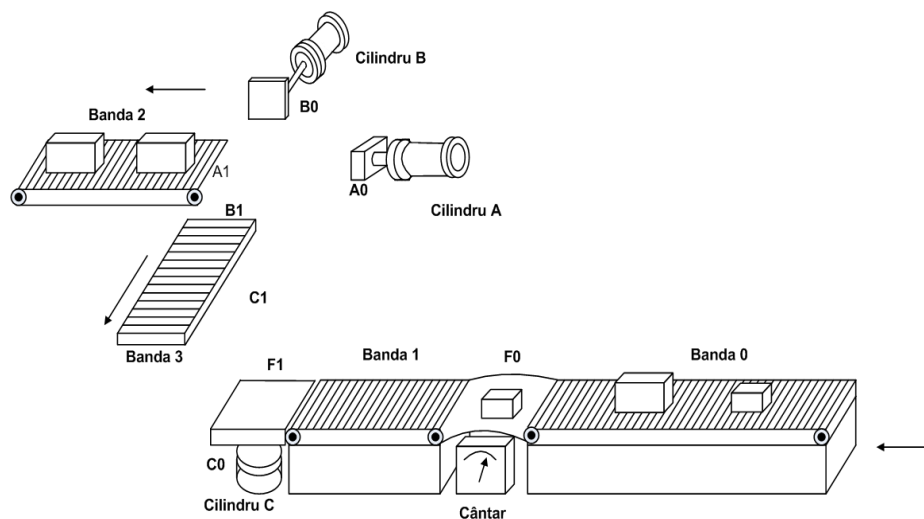


Fig.5.1. Elevator clasificator de pachete

Procesul pornește cu transportul unui pachet către cântar. Când pachetul ajunge pe cântar, banda 0 va fi oprită. Aici pachetul este cântărit fiind astfel identificat în funcție de greutatea citită. În cazul în care cilindrul C este în repaus și nu are pachet deasupra, pachetul de pe cântar este transportat de banda 1 până la planul elevator. Când banda 1 este oprită și nu există pachet pe cântar, banda 0 poate reporni. Cilindrul C ridică pachetele. Apoi pachetele sunt transportate diferit: pachetele mici sunt plasate pe banda 2 de către cilindrul A, iar pachetele mari sunt așezate pe banda 3 de către cilindrul B. Cilindrul elevator C se retrage doar când cilindrul A, respectiv B au atins poziția de avans. Benzile 2 și 3 se opresc când cilindrul A, respectiv B ajunge la limita de retragere.

*Elemente de execuție:*

- trei cilindri cu dublu efect (A,B,C);
- patru benzi transportoare.

*Elemente de măsură:*

- șase limitatoare de cursă;
- două detectoare de prezență.

## 2. Soluția de automatizare

Pentru controlul acestei aplicații se alege un automat programabil de tip PEP Smart pentru care se dezvoltă un proiect ISaGRAF, ce cuprinde cinci programe secvențiale ce rulează în paralel și un program în secțiunea begin, program ce se execută la începutul fiecărui ciclu automat. Modulul de intrare analogică al automatului acceptă un semnal între 0 și 20 mA, conversia analog-numerică făcându-se pe 12 biți. Structura proiectului ISaGRAF este prezentată în fig.5.2.

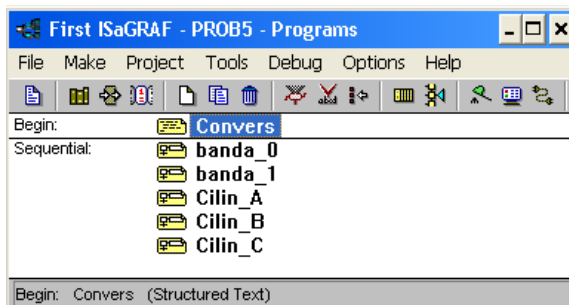
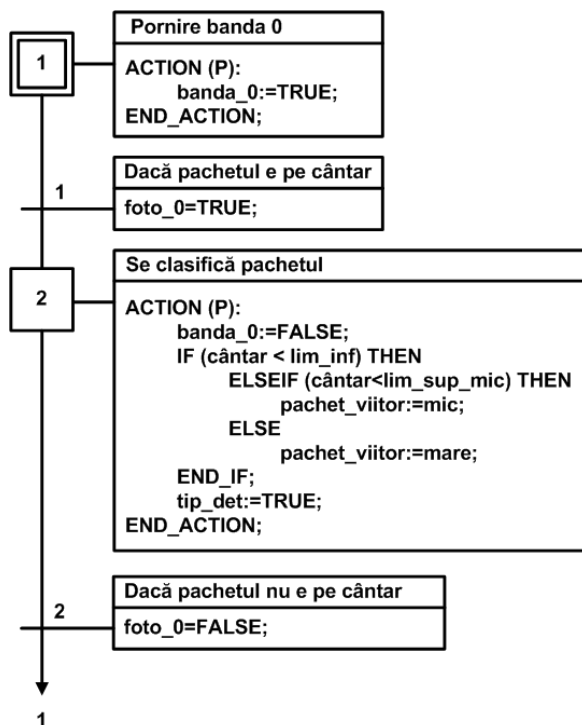


Fig.5.2.Structura proiectului ISaGRAF

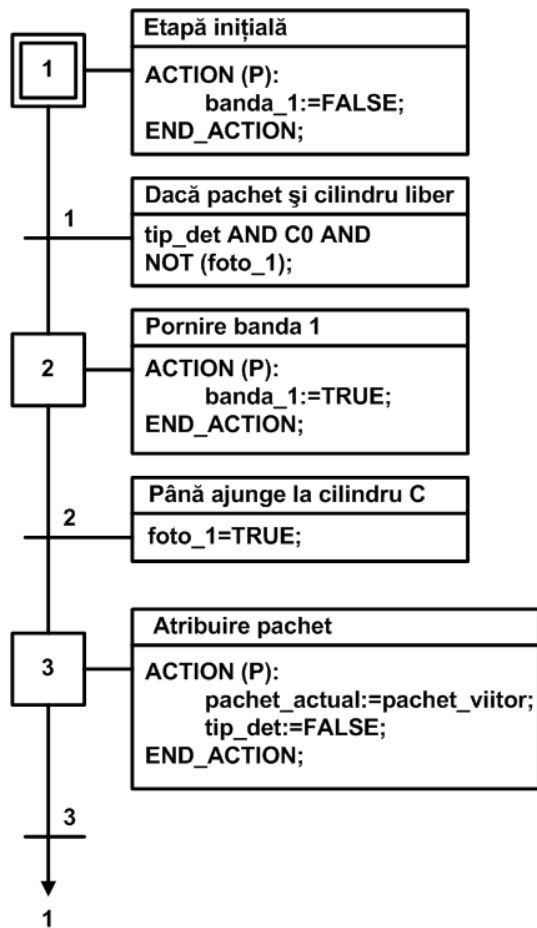
*Dicționarul de variabile globale:*

- *Variabile de intrare booleene:*
  - **foto\_0:** fotocelula 0;
  - **foto\_1:** fotocelula 1;
  - **A0:** limită retragere cilindru A;
  - **A1:** limită avans cilindru A;
  - **B0:** limită retragere cilindru B;
  - **B1:** limită avans cilindru B;
  - **C0:** limită coborâre cilindru C;
  - **C1:** limită ridicare cilindru C.
  
- *Variabile de ieșire booleene:*
  - **banda\_0:** activare/ dezactivare banda 0;
  - **banda\_1:** activare/ dezactivare banda 1;
  - **banda\_2:** activare/ dezactivare banda 2;
  - **banda\_3:** activare/ dezactivare banda 3;
  - **A\_retras:** retragere cilindru A;
  - **A\_avans:** avans cilindru A;
  - **B\_retras:** retragere cilindru B;
  - **B\_avans:** avans cilindru B;
  - **C\_retras:** retragere cilindru C;
  - **C\_ridicare:** ridicare cilindru C.
  
- *Variabile interne booleene:*
  - **tip\_det:** variabila de sincronizare între cântărire și banda transportoare 1;
  
- *Variabile globale analogice:*
  - **cântar:** variabilă internă, reprezintă valoarea reală a greutății de pe cântar (integer);
  - **traductor\_cântar:** variabilă de intrare, valoarea primită de la traductorul cântarului (integer, între 0 - 4096);
  - **pachet\_actual:** variabilă internă, în care se memorează tipul pachetului actual ce urmează a fi transportat;
  - **pachet\_viitor:** variabilă internă, în care se memorează tipul pachetului de pe cântar (următorul ce va fi transportat);
  - **lim\_inf:** constantă, greutatea minimă a pachetului mic;
  - **lim\_sup\_mic:** greutatea maximă a pachetului mic;
  - **mic:** constantă cu valoarea 1;
  - **mare:** constantă cu valoarea 2.

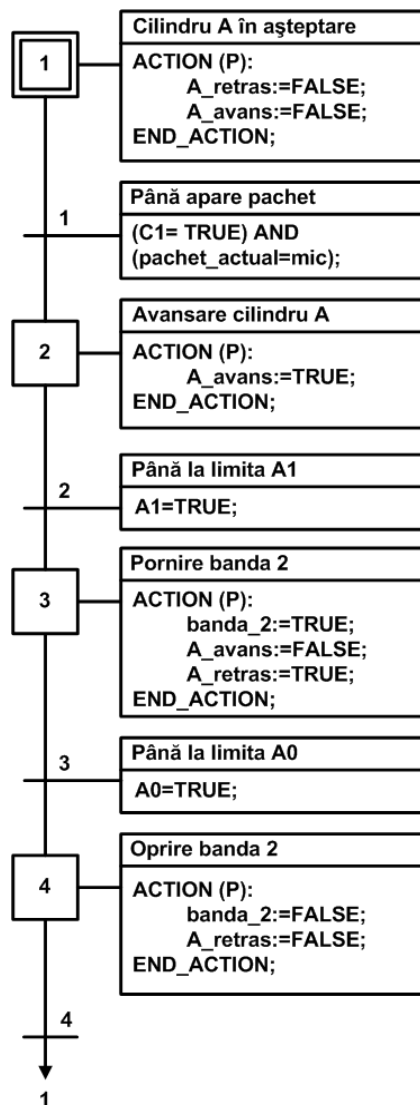
**Program banda\_0:**



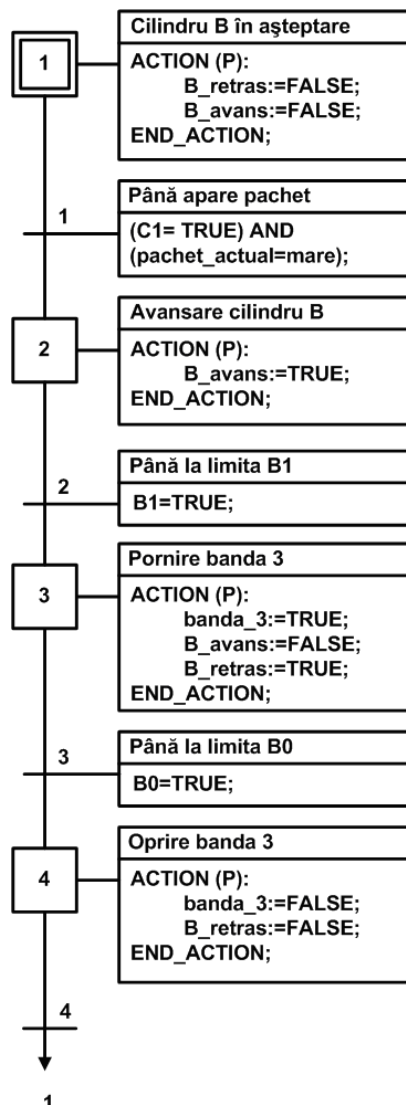
Program banda\_1:



### Program Cilin\_A

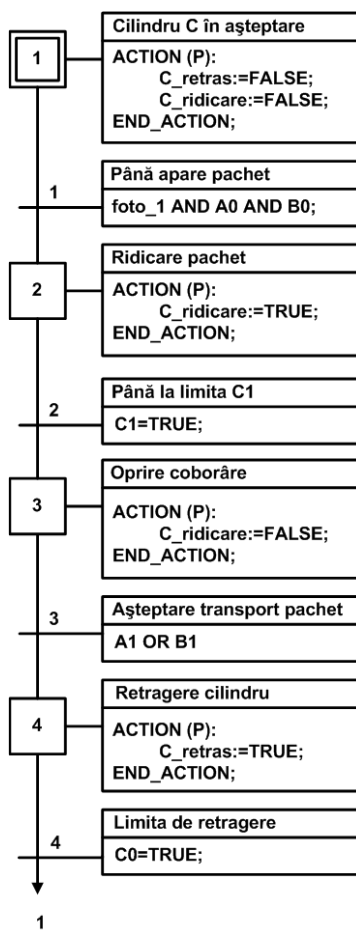


### Program Cilin\_B





Program Cilin\_C:



**Programul Convers:**

Programul “*convers*” realizează conversia din unități CAN (Convertor Analog Numeric) în valori exprimate în unități de măsură inginerești. Intrarea analogică a modului de intrare lucrează pe 12 biți și măsoară un curent de 0-20 mA, dar traductorul de la cântar generează un curent de 4-20 mA. În această situație trebuie făcută o translație de scală. Se observă că la valoarea minimă a domeniului de măsură traductorul generează 4mA, corespunzătoare valorii 819 citită de automat, valoare pentru care automatul trebuie să indice valoarea minimă a mărimii măsurate. Astfel formula de conversie este:

$$\text{val\_ing} = (\text{val\_cit} - 819) * (\text{Ds} - \text{Di}) / (4095 - 819)$$

unde:

val\_ing – valoarea în unități inginerești  
 val\_cit – valoarea citită în unități CAN  
 Di – domeniul inferior de măsură  
 Ds – domeniul superior de măsură

În cazul nostru: Ds = 100, Di = 0, astfel încât instrucțiunea în programul *convers* se scrie:

```
cantar := INT((( REAL(traductor_cantar ) - 819)*100.0)
/ (3276) ;
```

*Observații:*

- s-au folosit 2 variabile pentru memorarea tipului pachetelor deoarece la un moment dat în instalație pot exista 2 pachete neevacuate, unul pe cântar sau pe banda 1 și altul în curs de evacuare;
- comunicația între programul banda\_0 care determină tipul pachetului și programul banda\_1 se face prin intermediul variabile *tip\_det*, cu valoarea TRUE din momentul în care tipul unui pachet a fost determinat până când pachetul a ajuns deasupra planului elevator.

*Propunere:*

- Să se modifice programul în cazul în care nu există un senzor de prezență pachet în dreptul cântarului, aceasta determinându-se prin creșterea greutateii de pe cântar peste o valoare numită *marja*. Identificarea tipului pachetului se va face după un timp numit *Tstabilizare* de la detectarea prezenței pachetului deasupra cântarului.

## Problema 6: Dozare și malaxare automată

### 1. Descrierea instalației și a procesului

Sistemul conține:

- un malaxor pivotant care se poate roti în jurul axei sale și poate pivota dreapta/ stânga;
- două containere pline cu substanțele A, respectiv B, prevăzute cu robineți pentru golire;
- o bandă transportoare pe care vin brichete solubile;
- un senzor de detecție brichetă poziționat sub banda transportoare, activ atâta timp cât prin dreptul său trece o brichetă ( $d$ );
- un cântar prevăzut cu un robinet pentru golire, care transmite două semnale digitale A și B, corespunzătoare anumitor valori ale greutății, respectiv semnalul Z, atunci când cântarul este gol.

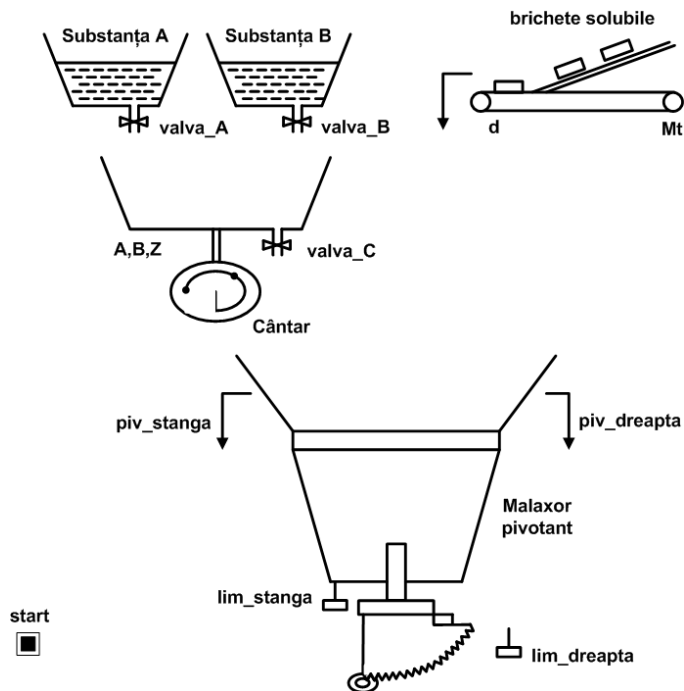


Fig.6.1. Dozare și malaxare automată

Utilitatea acestei instalații este de a amesteca anumite cantități din cele două substanțe cu câte două brichete solubile pentru a obține un produs finit. O șarjă pornește la acționarea butonului de alimentare, moment în care se pornește cântărirea și alimentarea produselor în următorul mod:

- cântărirea produsului A prin deschiderea valvei A, până la referința A;
- cântărirea produsului B prin deschiderea valvei B, până la referința B;
- apoi, golirea cântarului în malaxor prin deschiderea valvei C până la referința zero;
- simultan cu precedentele operații are loc alimentarea malaxorului cu două brichete solubile.

După ce aceste secvențe s-au încheiat, malaxorul se va roti în jurul axei sale timp de 2 minute, apoi, fără a opri rotația, va pivota către dreapta pentru a evacua conținutul. Apoi va pivota înapoi către poziția sa de repaus. După ce a atins această poziție un nou ciclu de producție poate porni la apăsarea butonului de alimentare.

*Elemente de execuție:*

- trei electrovalve (valva\_A, valva\_B, valva\_C);
- motorul benzii transportoare cu un singur sens de rotație;
- motorul de rotație al malaxorului;
- motorul de pivotare al malaxorului, cu două sensuri de rotație.

*Elemente de măsură:*

- trei senzori de greutate pentru referințele A, B și zero;
- două limitatoare de cursă;
- un detector de trecere.

## 2. Soluția de automatizare

### □ Implementarea în mediul ISaGRAF

Pentru controlul acestei aplicații se alege un automat programabil de tip PEP Smart pentru care se dezvoltă un proiect ISaGRAF, ce cuprinde un program principal. Structura proiectului ISaGRAF este prezentată în fig.6.2.

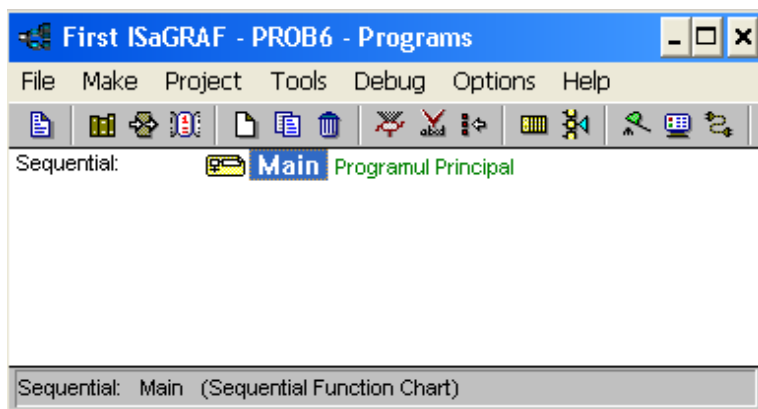
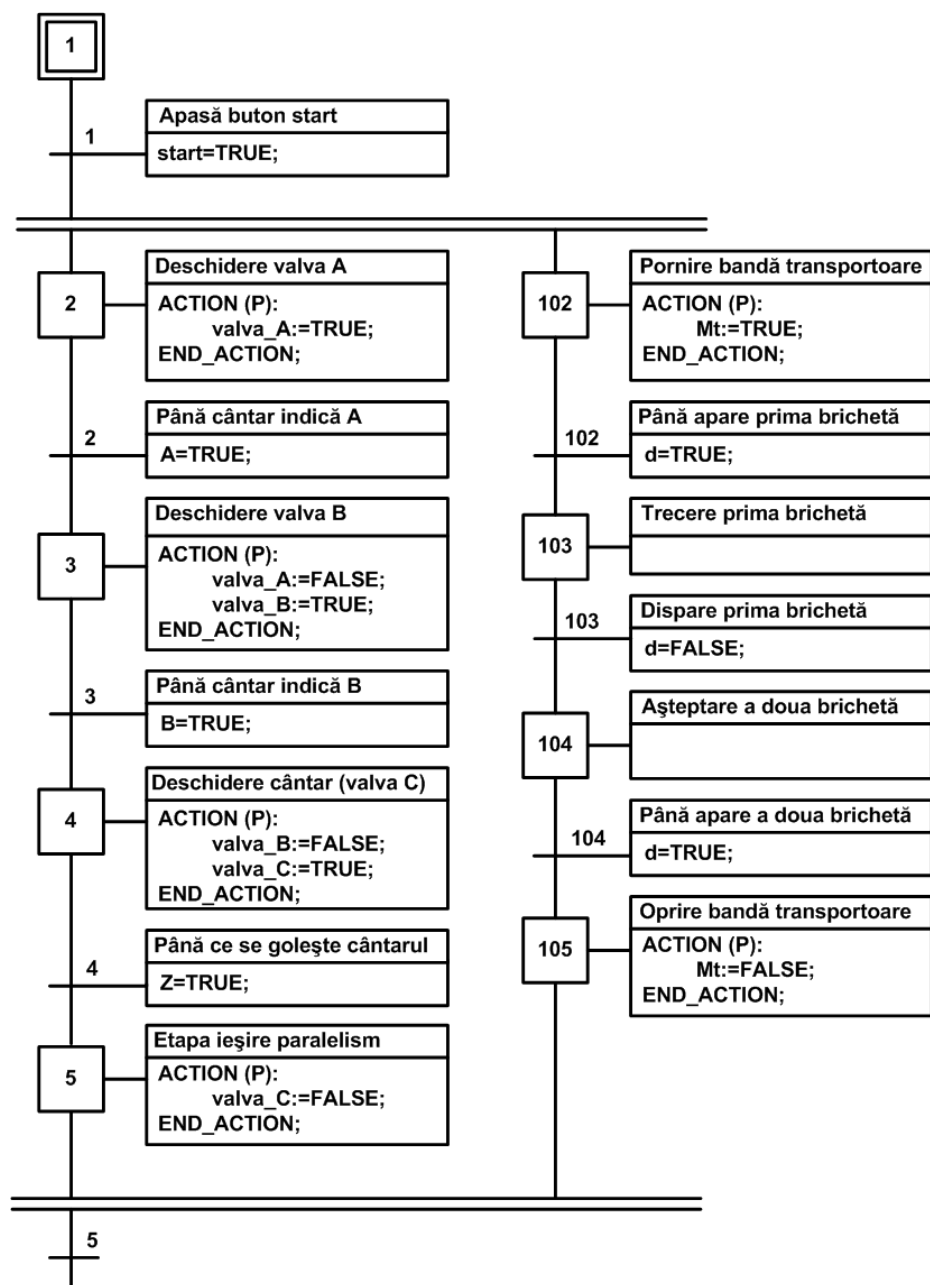


Fig.6.2. Structura proiectului ISaGRAF

Dicționarul de variabile globale:

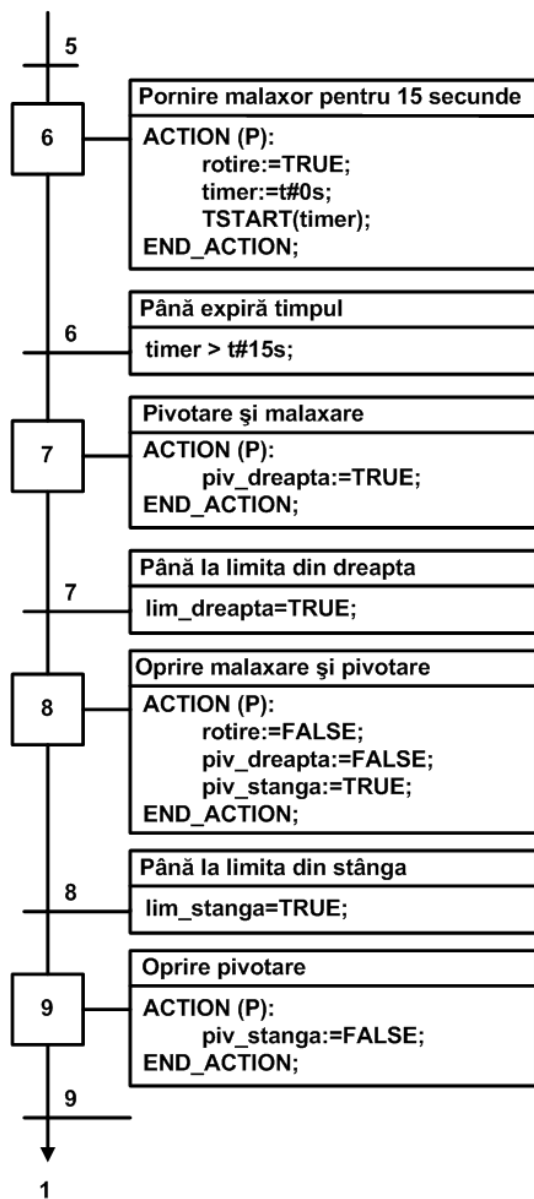
- *Variabile de intrare booleene:*
  - **start**: buton de pornire;
  - **d**: detector de trecere brichete solubile;
  - **A**: detectorul greutății produsului A;
  - **B**: detectorul greutății produselor A + B;
  - **Z**: detectorul golirii cântarului;
  - **lim\_stanga**: limitator stânga malaxor;
  - **lim\_dreapta**: limitator dreapta malaxor.
- *Variabile de ieșire booleene:*
  - **valva\_A**: comandă deschiderea/ închiderea valvei A;
  - **valva\_B**: comandă deschiderea/ închiderea valvei B;
  - **valva\_C**: comandă deschiderea/ închiderea valvei C;
  - **Mt**: comandă motorul benzii transportoare;
  - **rotire**: comandă rotația malaxorului;
  - **piv\_dreapta**: comandă pivotarea către dreapta a malaxorului;
  - **piv\_stanga**: comandă pivotarea către stânga a malaxorului.
- *Variabile globale de tip timer:*
  - **timer**: temporizare folosită la rotația malaxorului.

**Programul Main:**



61

Problema 6 – Dozare și malaxare automată



### □ Implementarea în limbajul Ladder Diagram

Pentru controlul acestei aplicații s-a ales un automat programabil de tip Allen Bradley. Programul de tip Ladder Diagram este construit pe baza diagramei logice din proiectul IsaGraf prezentat anterior.

Asocierea intrărilor și ieșirilor fizice cu biți din regiștrii de intrare/ieșire este prezentată în tabelul 6.1:

**Tabelul 1.1.**

| Intrare fizică | Adresă internă | Ieșire fizică | Adresă internă |
|----------------|----------------|---------------|----------------|
| START          | I:1/1          | Valva_A       | O:3/1          |
| A              | I:1/2          | Valva_B       | O:3/2          |
| B              | I:1/3          | Valva_C       | O:3/3          |
| Z              | I:1/4          | Mt            | O:3/4          |
| lim_dreapta    | I:1/5          | Rotire        | O:3/5          |
| d              | I:1/6          | Pit_dreapta   | O:3/6          |
| lim_stanga     | I:1/7          | Pit_stanga    | O:3/7          |

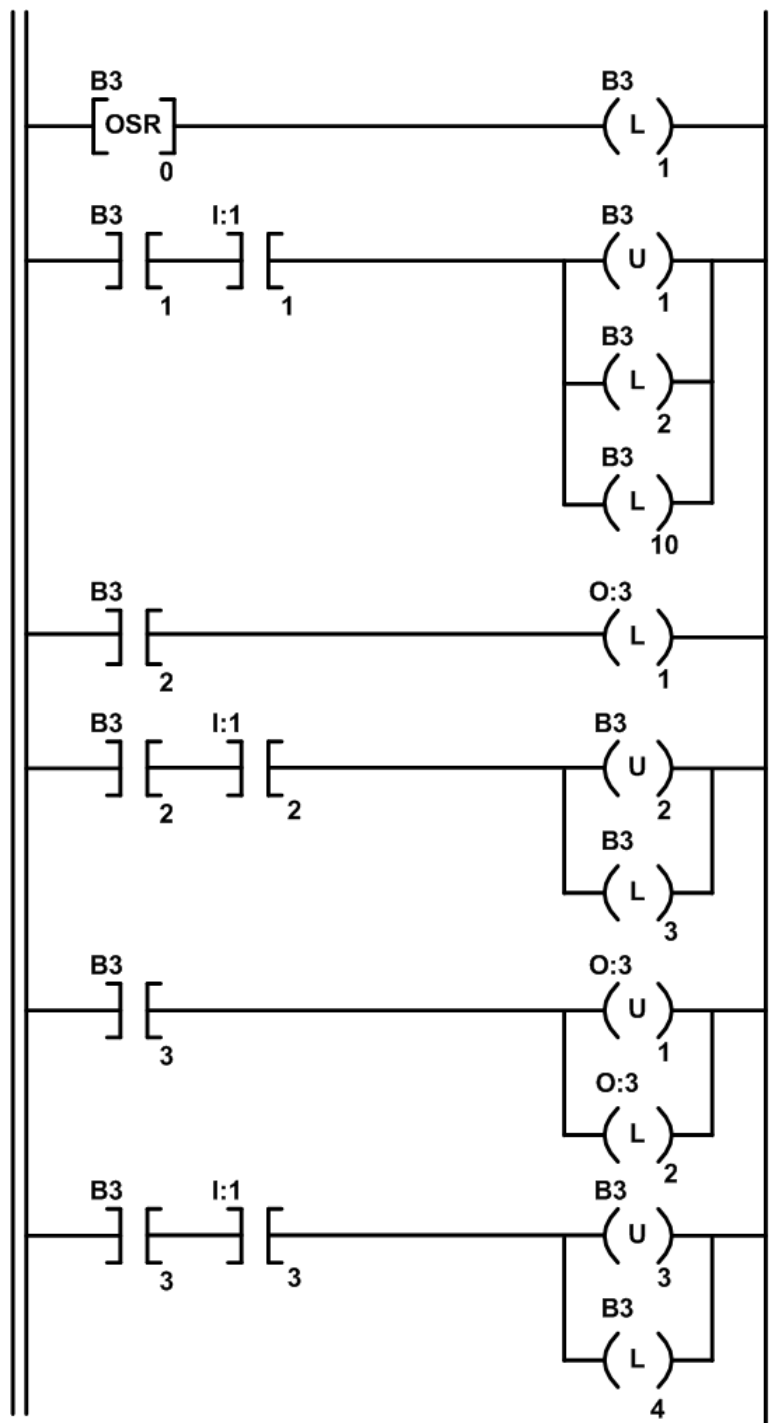
Asocierea etapelor cu biți din fișierul de bit B3 și alegerea fișierului de timer este prezentată în tabelul 6.2:

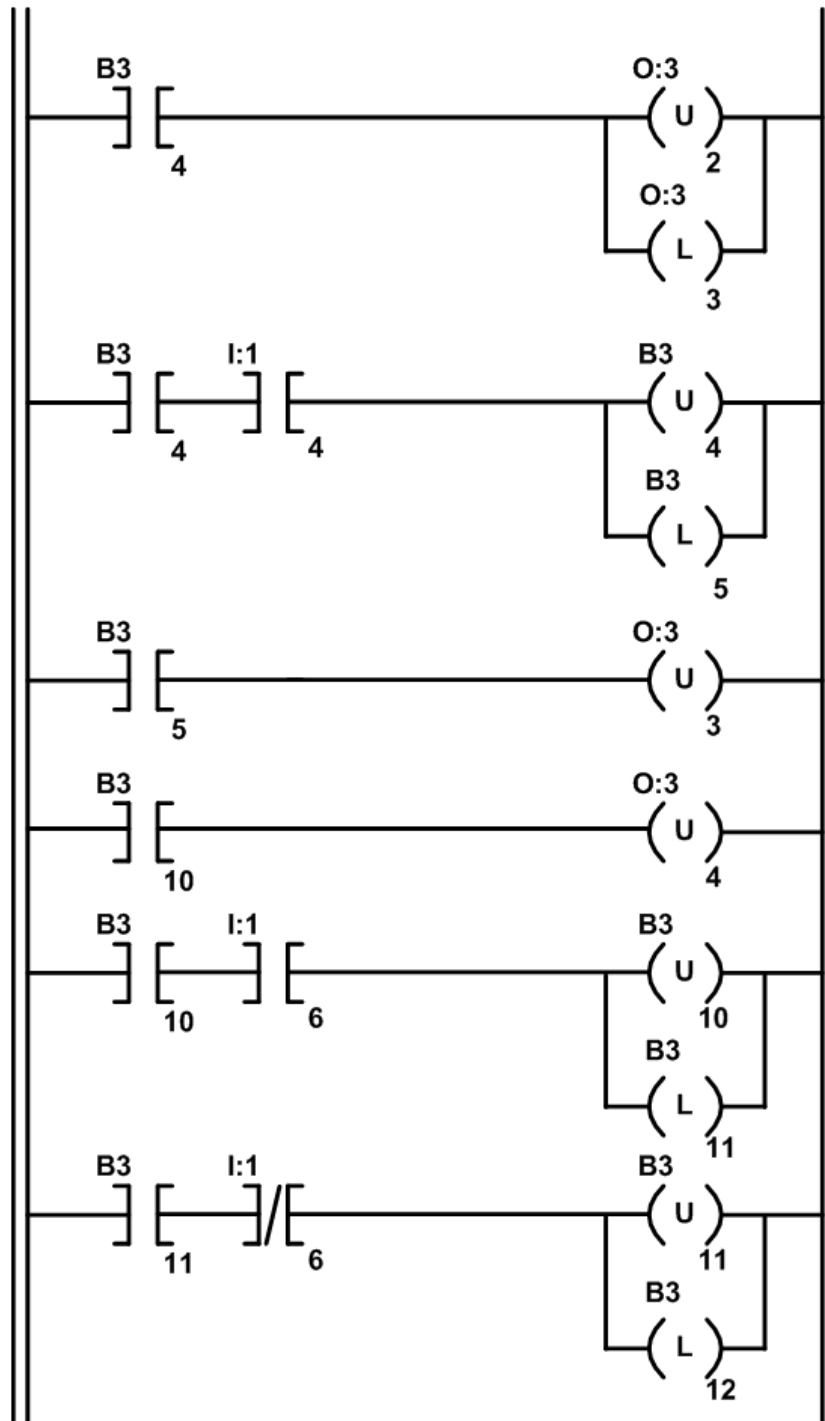
**Tabelul 1.2.**

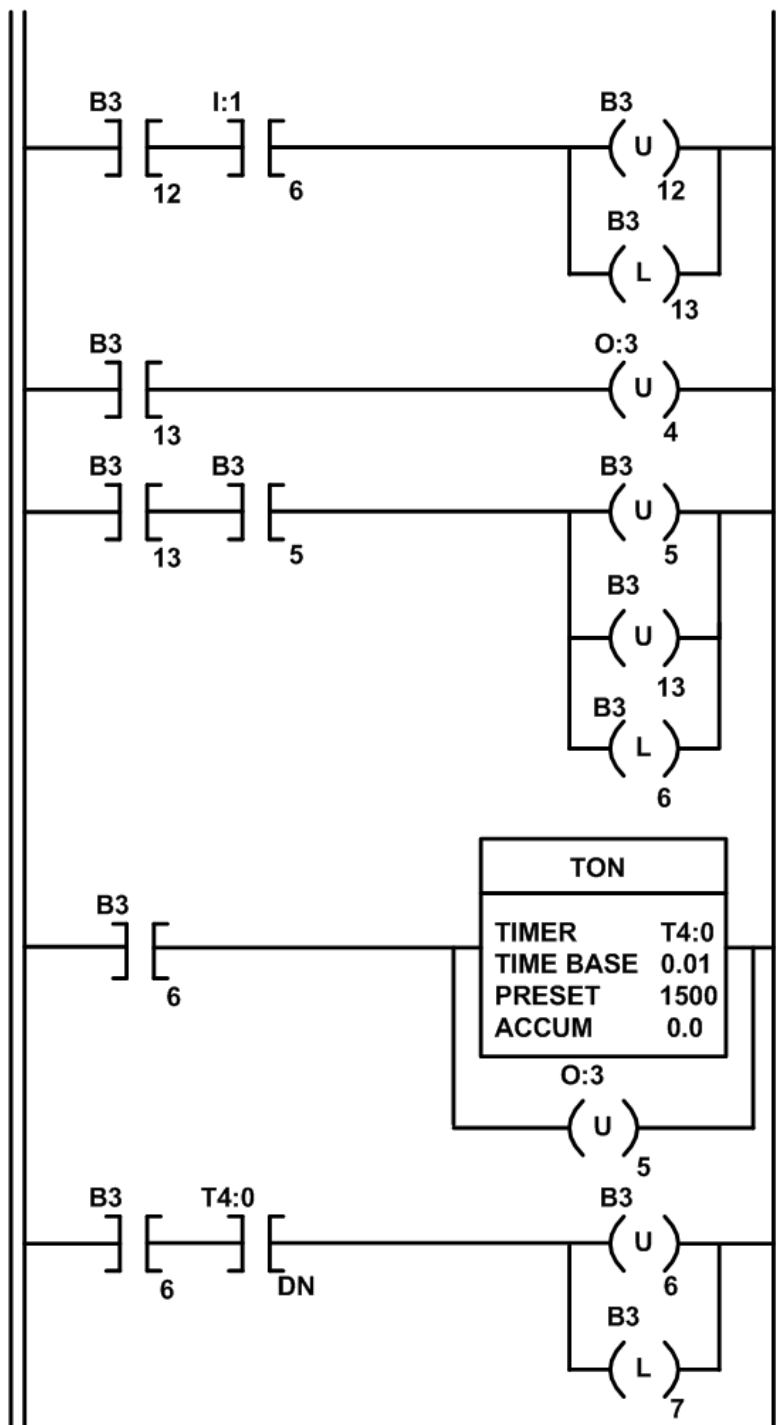
| Etapă | Adresa bit | Temporizare   | Fișier de timer |
|-------|------------|---------------|-----------------|
| 1     | B3/1       | Temporizare 1 | T4:0            |
| 2     | B3/2       |               |                 |
| 3     | B3/3       |               |                 |
| 4     | B3/4       |               |                 |
| 5     | B3/5       |               |                 |
| 6     | B3/6       |               |                 |
| 7     | B3/7       |               |                 |
| 8     | B3/8       |               |                 |
| 9     | B3/9       |               |                 |
| 102   | B3/10      |               |                 |
| 103   | B3/11      |               |                 |
| 104   | B3/12      |               |                 |
| 105   | B3/13      |               |                 |

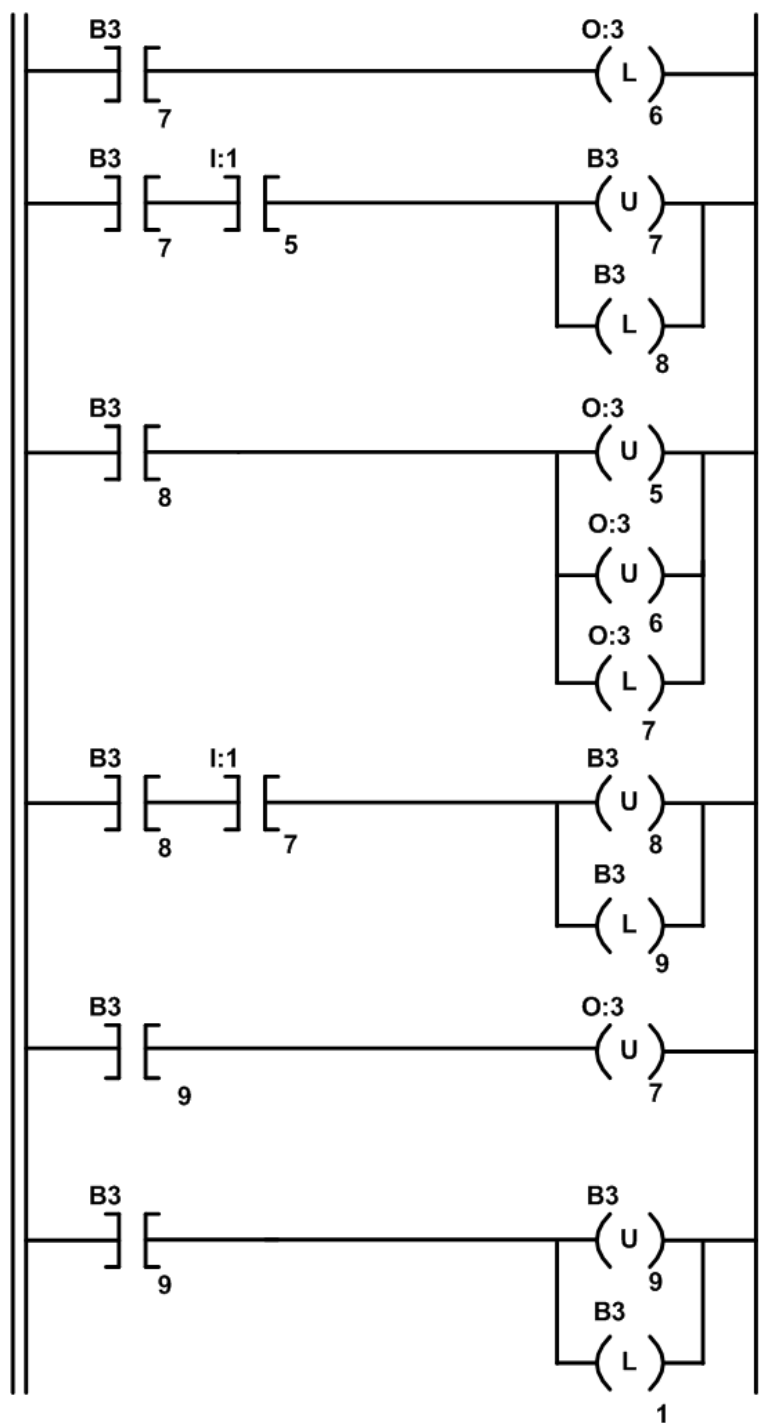
Diagrama Ladder este prezentată în continuare:











*Observații:*

- În diagrama SFC se pot observa cele două secvențe paralele, aducerea lichidelor și aducerea brichetelor în malaxor. Ieșirea din paralelism se face atunci când ambele secvențe s-au încheiat.
- Pentru exemplificare, temporizarea a fost realizată cu instrucțiunea TSTART deși în acest caz putea fi folosită variabila globală GS006.t. Nu s-a mai folosit funcția TSTOP pentru că în etapa 7 variabila timer este resetată, deci incrementarea ei va reîncepe de la 0 din acel moment, în alte etape sau tranziții variabila nefiind folosită.

*Propunere:*

- Să se modifice programul pentru cazul în care aducerea brichetelor începe odată cu golirea cântarului iar numărul de brichete aduse este oarecare (egal cu n).

## Problema 7: Umplerea și astuparea automată a sticlelor

### 1. Descrierea instalației și a procesului

Scopul acestei aplicații îl constituie controlul unui sistem de umplere și astupare a unor sticle.

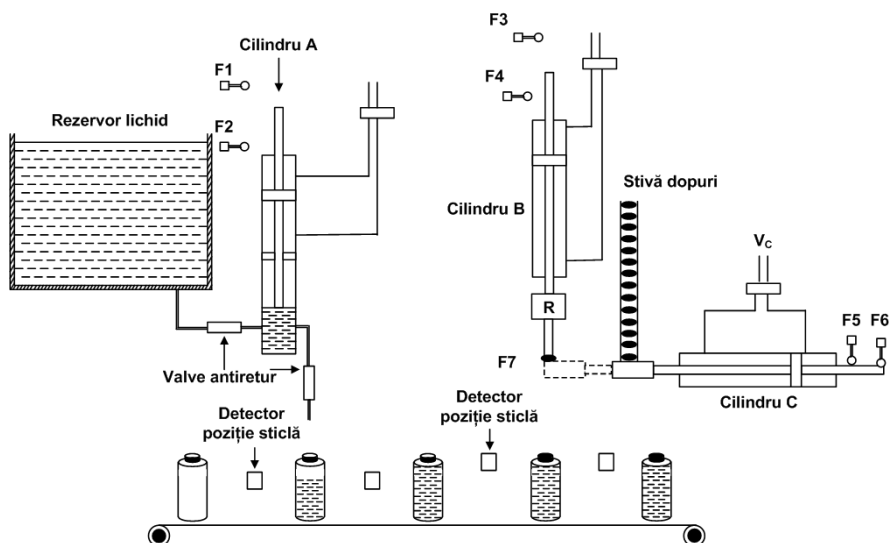


Fig.7.1. Umplerea și astuparea automată a sticlelor

Componentele instalației sunt:

- o bandă transportoare pe care circulă sticle, acționată de un motor cu un singur sens de rotație;
- un rezervor cu lichid folosit la umplerea sticlelor;
- un cilindru cu dublu efect (A în fig. 7.1) cu ajutorul căruia sunt umplute sticlele;
- un cilindru cu dublu efect (B în fig. 7.1) care aduce dopurile pentru astuparea sticlelor;
- un cilindru cu dublu efect (C în fig. 7.1) care preia dopurile din stiva de dopuri;
- un mecanism rotativ care efectuează efectiv astuparea sticlelor, a cărui mișcare este limitată de către un limitator;

- doi senzori detectori de prezența unei sticle în pozițiile de umplere, respectiv de astupare;
- un detector al poziției de preluare dop de către cilindrul B.

Inițial, pentru simplificarea aplicației, se presupune că există deja o sticlă umplută între cele două poziții principale de pe bandă (umplere și astupare).

La inițializarea aplicației se pornește motorul benzii transportoare. Acesta se va opri când există sticle atât în poziția de umplere cât și în cea de astupare. Se presupune că sticlele vin pe bandă la distanțe egale astfel încât umplerea și astuparea să aibă loc simultan. Acțiunea de umplere are loc prin avansul și retragerea cilindrului A între limitatoarele F1 și F2. Retragera cilindrului A trebuie să aibă loc abia când cilindrul B a pus un dop deasupra unei sticle. Secvența de astupare este următoarea: Cilindrul C aduce un dop din stiva de dopuri, cilindrul B avansează până în dreptul detectorului F7, cilindrul C se retrage apoi cilindrul B avansează până la limita sa de avans, împingând dopul deasupra sticlei. Din acel moment încep două secvențe de acțiuni care au loc în paralel. Prima constă în înșurubarea dopului de către mecanismul rotativ, urmată de retragerea cilindrului B, cea de-a doua fiind retragerea cilindrului A. Când ambele secvențe s-au încheiat, banda transportoare poate fi repornită, ciclul reluându-se când un nou lot de sticle apar în cele două poziții semnificative de pe bandă.

*Elemente de execuție:*

- un cilindru A ce reglează dozatorul volumetric;
- un cilindru de avans B cu trei poziții;
- un cilindru C ce reprezintă mecanismul de transfer al capacelor;
- motorul benzii transportoare;
- un mecanism de înșurubare dopuri.

*Elemente de măsură:*

- șase limitatoare de cursă;
- un detector de poziție;
- o fotocelulă pentru detecție sticlă de umplut;
- o fotocelulă pentru detecție sticlă plină.

## **2. Soluția de automatizare**

Pentru controlul acestei aplicații s-a ales un automat programabil de tip PEP Smart pentru care s-a dezvoltat un proiect ISaGRAF ce cuprinde un program principal, numit main.

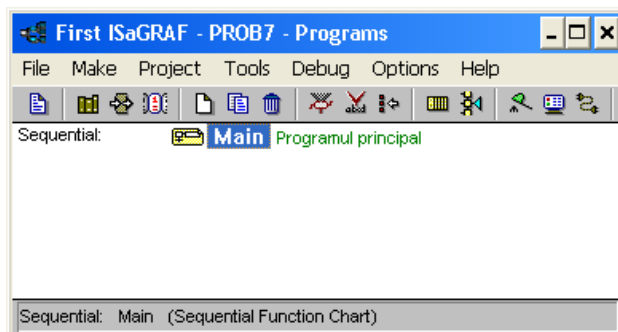


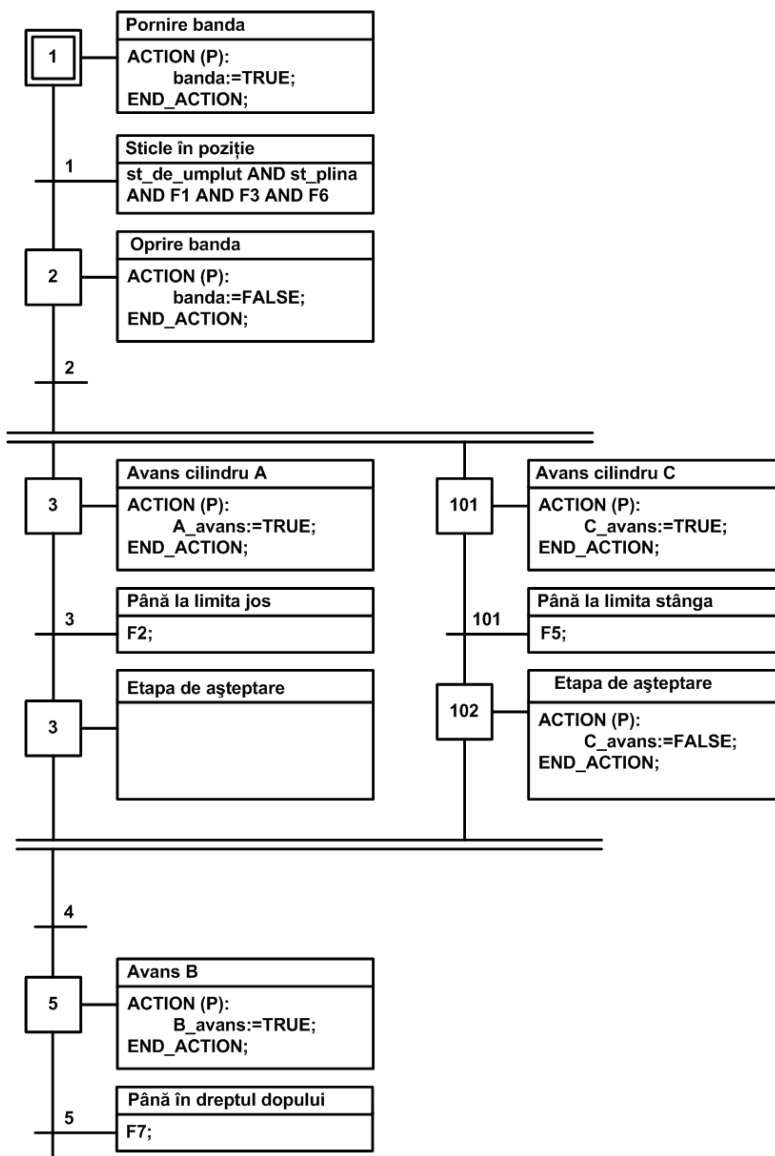
Fig.7.2. Structura proiectului ISaGRAF

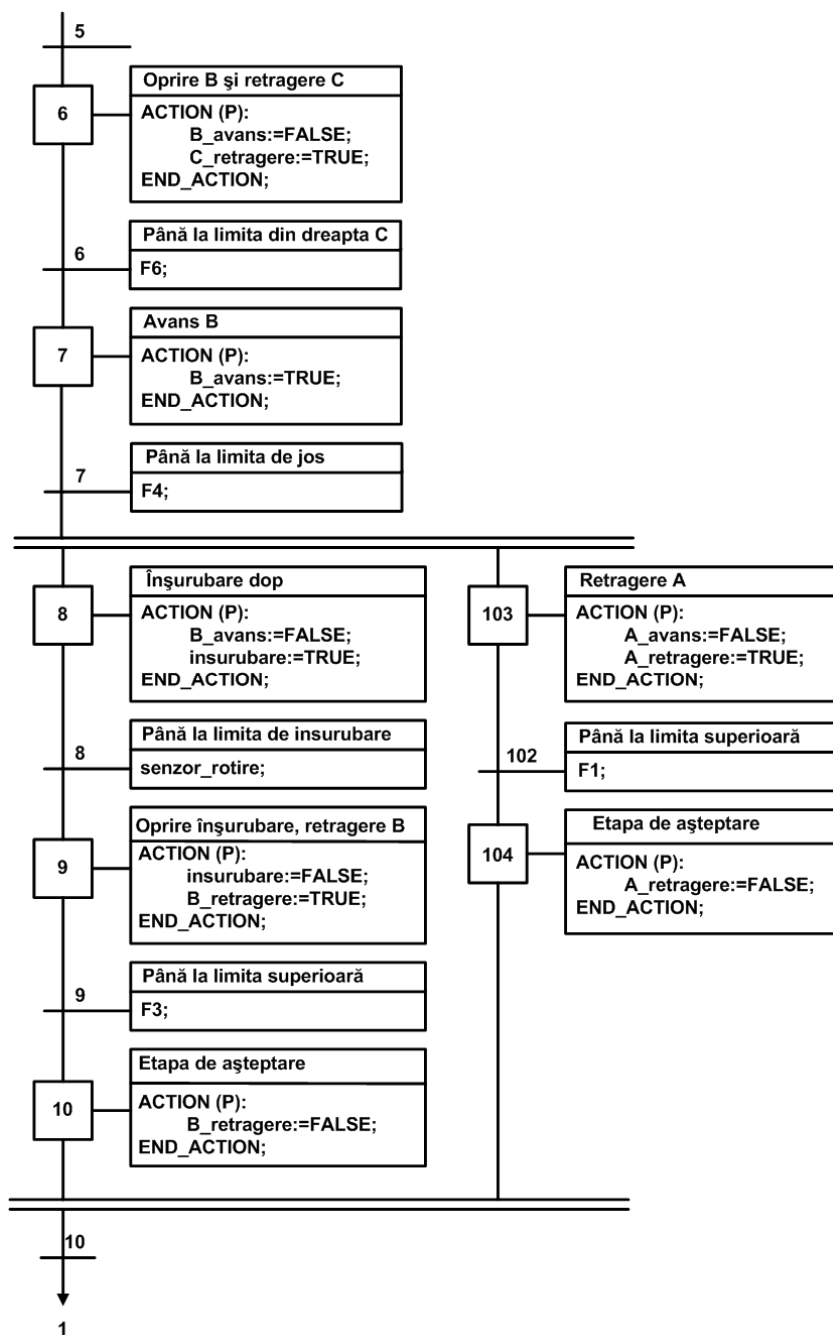
*Dicționarul de variabile globale:*

- *Variabile de intrare booleene:*
  - **F1:** limitator sus cilindru A
  - **F2:** limitator jos cilindru A
  - **F3:** limitator sus cilindru B
  - **F4:** limitator jos cilindru B
  - **F5:** limitator stânga cilindru C
  - **F6:** limitator dreapta cilindru C
  - **F7:** Detector poziție de preluat dop
  - **senzor\_rotire:** limitator rotire dispozitiv de înșurubare
  - **st\_de\_umplut:** senzor detecție sticlă de umplut
  - **st\_plină:** senzor detecție sticlă plină
- *Variabile de ieșire booleene:*
  - **banda:** comandă pornire/ oprire bandă
  - **insurubare:** comanda de înșurubare a dopului
  - **A\_avans:** comandă avans cilindru A
  - **A\_retragere:** comandă retragere cilindru A
  - **B\_avans:** comandă avans cilindru B
  - **B\_retragere:** comandă retragere cilindru B
  - **C\_avans :** comandă avans cilindru C
  - **C\_retragere:** comandă retragere cilindru C



### Programul main:





*Observații:*

- În program au fost folosite 2 paralelisme, fiecare având câte 2 secvențe ce se execută simultan.
- A doua secvență paralelă implementează condiția că o dată cu umplerea unei sticle, alta să fie astupată.

*Propunere:*

- Să se modifice programul în situația în care sticlele vin aleator pe banda transportoare, iar umplerea și astuparea nu mai au loc simultan.
- Să se modifice programul ținându-se cont de situația în care nu apar sticle simultan în cele 2 poziții semnificative. În acest caz să fie declanșată o alarmă care să fie oprită și ciclul reluat abia când nu mai există sticle în niciuna din cele 2 poziții de pe bandă (sticla este evacuată manual).

## Problema 8: Controlul unui lift

### 1. Descrierea sistemului și a procesului

Componentele sistemului sunt:

- 10 butoane interioare;
- 10 butoane exterioare;
- un senzor prezență persoană în interiorul liftului;
- 10 senzori prezență lift la fiecare etaj;
- un contact de ușă închisă a cabinei liftului;
- un motor cu două sensuri de rotație.

Funcționarea dorită a liftului este următoarea: inițial liftul se află la parter, gol. Dacă o persoană se află în lift și apasă unul din cele 10 butoane interne de comandă, liftul va porni către nivelul destinație, oprindu-se când se activează senzorul de prezență lift, de la nivelul respectiv. Pornirea liftului are loc numai dacă ușa de interior este închisă. Dacă liftul este gol el poate fi chemat de la alt nivel, cu ajutorul butoanelor externe.

*Elemente de execuție:*

- motorul liftului

*Elemente de măsură:*

- 10 senzori prezență lift;
- 20 butoane;
- un senzor prezență persoană;
- un contact de ușă închisă.

### 2. Soluția de automatizare

Pentru controlul acestei aplicații se alege un automat programabil de tip PEP Smart pentru care se dezvoltă un proiect ISaGRAF, ce cuprinde un program în secțiunea begin și un program în secțiunea sequential, așa cum este prezentat și în figura 8.1.

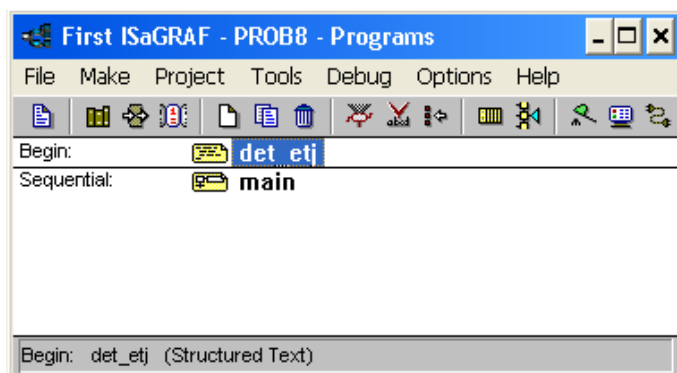


Fig.8.1. Structura proiectului ISaGRAF

Dicționarul de variabile globale:

- Variabile de intrare booleene:
  - **bint1 ... bint10**
  - **bext1 ... bext 10**
  - **usa\_cabina**
  - **etaj1 ... etaj 10**
  - **prez\_persoana**
- Variabile de ieșire booleene:
  - **Msus, Mjos**
- Variabile interne de tip întreg:
  - **but\_int**
  - **but\_ext**
  - **etaj\_curent**
  - **etaj\_dorit**

## Program det\_etj

```

but_int:=0;
but_ext:=0;
etaj_curent:=0;

IF bint1 = TRUE THEN
    but_int = 1;
END_IF;
IF bint2 = TRUE THEN
    but_int = 2;
END_IF;
IF bint3 = TRUE THEN
    but_int = 3;
END_IF;
IF bint4 = TRUE THEN
    but_int = 4;
END_IF;
IF bint5 = TRUE THEN
    but_int = 5;
END_IF;
IF bint6 = TRUE THEN
    but_int = 6;
END_IF;
IF bint7 = TRUE THEN
    but_int = 7;
END_IF;
IF bint8 = TRUE THEN
    but_int = 8;
END_IF;
IF bint9 = TRUE THEN
    but_int = 9;
END_IF;
IF bint10 = TRUE THEN
    but_int = 10;
END_IF;

IF bext1 = TRUE THEN
    but_ext = 1;
END_IF;
IF bext2 = TRUE THEN
    but_ext = 2;
END_IF;
IF bext3 = TRUE THEN
    but_ext = 3;
END_IF;
IF bext4 = TRUE THEN
    but_ext = 4;
END_IF;
IF bext5 = TRUE THEN
    but_ext = 5;
END_IF;
IF bext6 = TRUE THEN
    but_ext = 6;
END_IF;
IF bext7 = TRUE THEN
    but_ext = 7;
END_IF;
IF bext9 = TRUE THEN
    but_ext = 9;
END_IF;
IF bext10 = TRUE THEN
    but_ext = 10;
END_IF;

IF etaj1 = TRUE THEN
    etaj_curent = 1;
END_IF;
IF etaj2 = TRUE THEN
    etaj_curent = 2;
END_IF;
IF etaj3 = TRUE THEN
    etaj_curent = 3;
END_IF;
IF etaj4 = TRUE THEN
    etaj_curent = 4;
END_IF;
IF etaj5 = TRUE THEN
    etaj_curent = 5;
END_IF;
IF etaj6 = TRUE THEN
    etaj_curent = 6;
END_IF;
IF etaj7 = TRUE THEN
    etaj_curent = 7;
END_IF;
IF etaj8 = TRUE THEN
    etaj_curent = 8;

```

77

## Problema 8 – Controlul unui lift

```

END_IF;
IF etaj9 = TRUE THEN
    etaj_curent = 9;
END_IF;

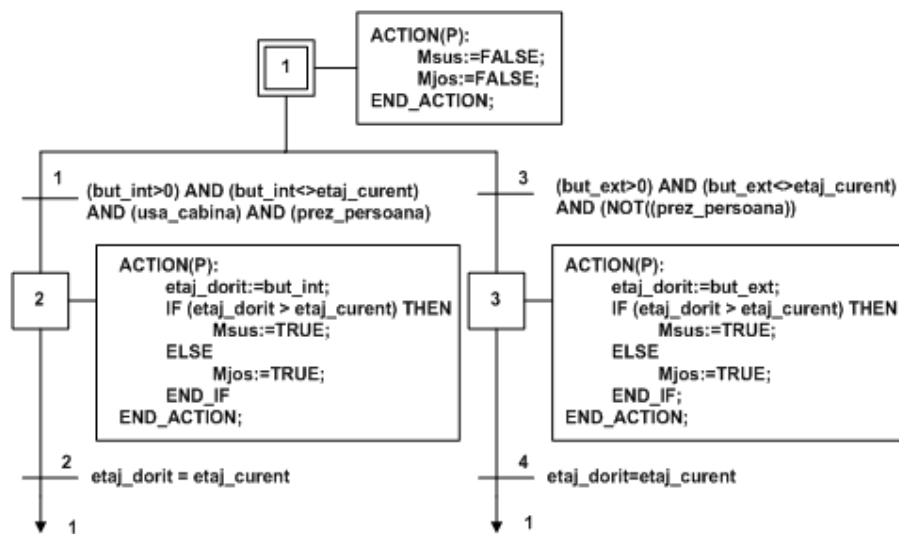
```

```

IF etaj10 = TRUE THEN
    etaj_curent = 10;
END_IF;

```

## Program Principal



## Observații:

- S-a ales soluția folosirii unui program ciclic în secțiunea Begin deoarece starea butoanelor și etajul unde se află liftul trebuie actualizate la fiecare ciclu automat pentru a putea controla corect liftul.
- Încercarea de a nu folosi un program ciclic și de a testa starea butoanelor și a senzorilor într-un program SFC ar duce la crearea unei diagrame foarte mari și stufoase, în mod practic nici nu ar fi posibil la un număr mare de butoane și senzori întrucât mediile de programare logică permit un număr limitat de condiții de tranziție atașate unei etape.

*Propunere:*

- În cazul în care ar exista un buton de urgență în interiorul liftului, să se modifice programul astfel încât liftul să se oprească la apăsarea acestuia, mișcarea fiind reluată la apăsarea oricărui buton de comandă mișcare.



## Problema 9: Detectarea bagajelor care conțin metale

### 1. Descrierea procesului

Elementele sistemului de condus sunt:

- o bandă transportoare acționată de un motor ( $MT$ ) cu un singur sens de rotație pe care sunt transportate bagaje într-un aeroport;
- o barieră comandată de un motor cu două sensuri de rotație  $Msus$ ,  $Mjos$ , având limitatoare de cursă  $lsus$  și  $ljjos$ ;
- un senzor de prezență bagaje  $s$ , activ cât timp în dreptul său se găsește un bagaj;
- un detector de metal  $d$ , activ dacă bagajul din dreptul senzorului  $s$  conține obiecte de metal.

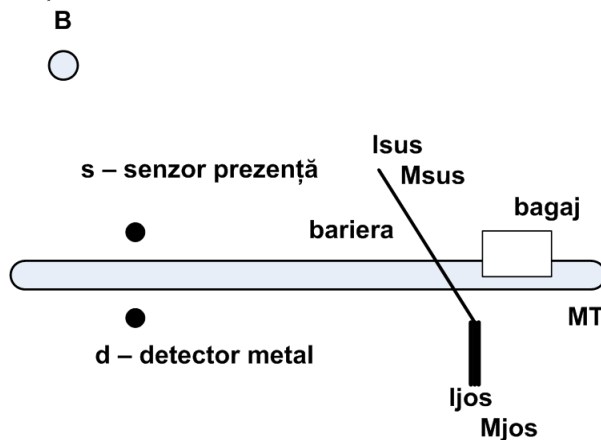


Fig. 9.1. Detectarea bagajelor care conțin metale

Aplicația are ca scop detectarea automată și evacuarea manuală a bagajelor care conțin metal. Inițial, bariera este ridicată. Pornirea benzii transportoare se face prin apăsarea unui buton  $B$ . Atunci când un bagaj apare în dreptul senzorului  $s$ , bariera trebuie să coboare. Bariera se va ridica atunci când bagajul dispăre din dreptul senzorului  $s$ . Dacă în timp ce se află în dreptul senzorului  $s$ , se activează senzorul  $d$ , banda va fi oprită și se așteaptă ca bagajul să fie preluat de pe bandă pentru a reporni banda și a ridica bariera. Dacă timp de 2 minute nu apare nici un bagaj în dreptul senzorului  $s$ , banda va fi oprită iar repornirea se va face de la buton.

*Elemente de execuție:*

- motorul benzii transportoare
- motorul barierei

*Elemente de măsură:*

- două limitatoare de cursă
- un senzor de prezență
- un detector de metal

## 2. Soluția de automatizare

Pentru controlul acestei aplicații s-a ales un automat programabil de tip PEP Smart pentru care s-a dezvoltat un proiect ISaGRAF ce cuprinde două programe conform figurii 9.2.

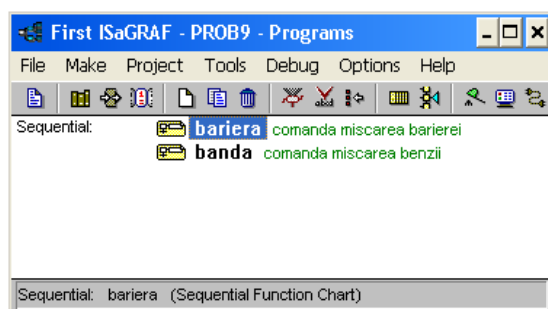


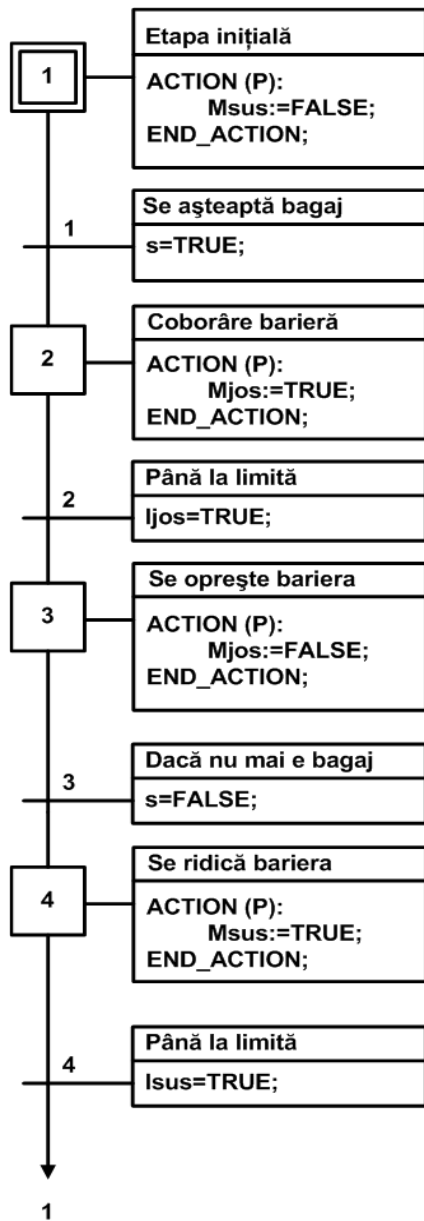
Fig.9.2. Structura proiectului ISaGRAF

*Dicționarul de variabile globale:*

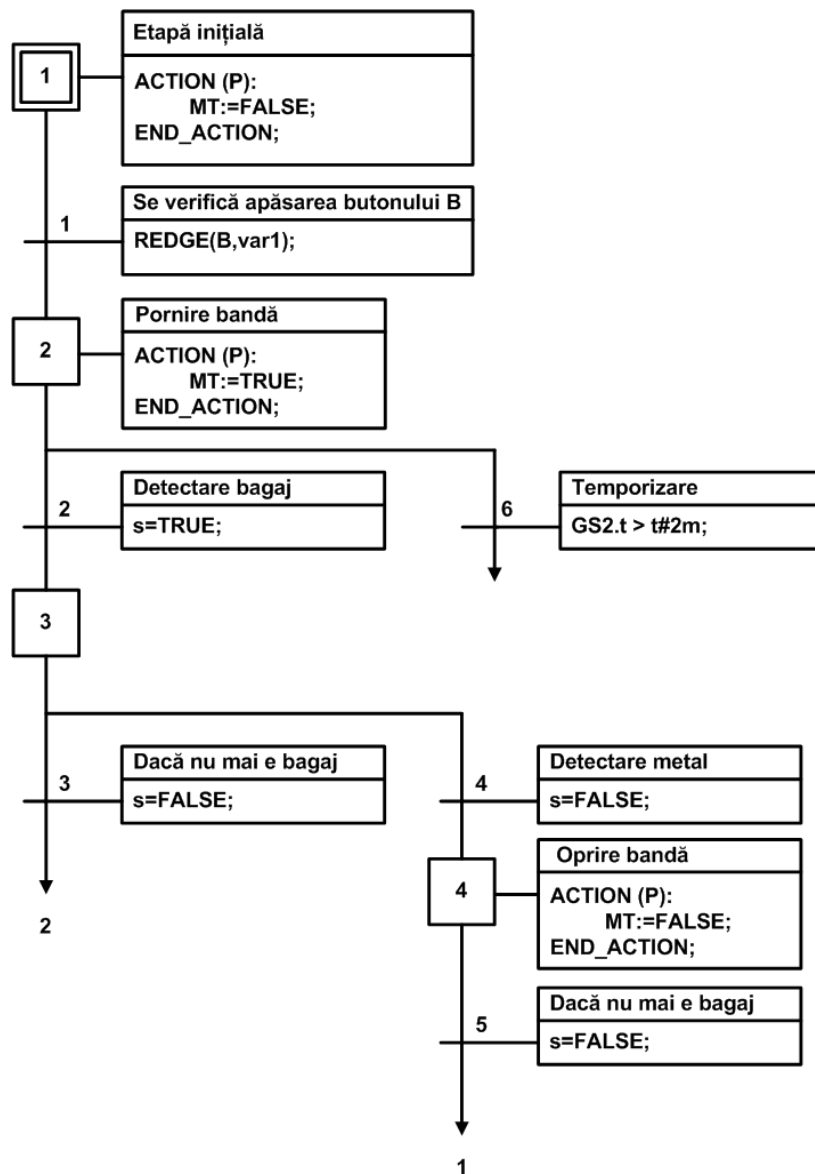
- *Variabile de intrare booleene:*
  - **s:** senzor de prezență bagaje
  - **d:** senzor detecție metal
  - **lsus:** limitator de cursă sus
  - **ljos:** limitator de cursă jos
  - **B:** buton de pornire
- *Variabile de ieșire booleene:*
  - **MT:** comandă banda transportoare
  - **Msus:** comandă bariera sus
  - **Mjos:** comandă bariera jos

- Variabilă internă de tip boolean:
  - **var1**: necesar funcției REDGE

Program bariera:



**Program bandă:**



*Observații:*

- Se observă ca testarea impulsului de la butonul B este făcută folosind funcția REDGE, deoarece trebuie depistată tranziția stării butonului și nu starea lui la un moment dat
- S-au folosit 2 programe ce rulează în paralel întrucât bariera și banda transportoare funcționează independent și nu pot fi gestionate în aceeași diagramă

*Propuneri:*

- Să se construiască diagrama Ladder care simulează acest proiect ISaGraf

## Problema 10: Sortarea a patru tipuri de piese

### 1. Descrierea sistemului și a procesului

Componentele sistemului sunt:

- un motor al benzii transportoare;
- un senzor prezență piesă (pentru detecție);
- un cititor cod bare cu ieșire pe doi biți;
- 4 senzori de prezență piesă în dreptul clapetelor;
- 4 clapete comandate digital (cu delay);
- un buton pornire bandă.

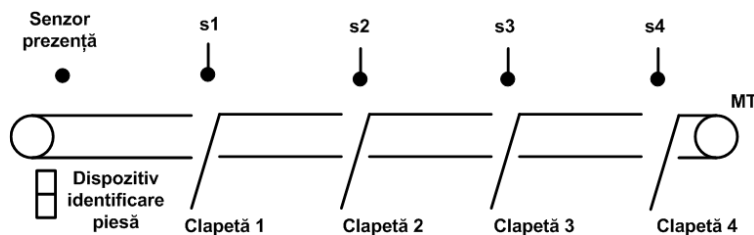


Fig.10.1. Sortarea a patru tipuri de piese

Scopul aplicației este de a sorta 4 tipuri de piese care vin pe o bandă transportoare. Inițial pe bandă nu se află nici o piesă iar banda este oprită. Aplicația pornește prin apăsarea butonului. Când o piesă ajunge în dreptul senzorului de prezență pentru detecție, tipul ei este determinat de către cititorul de bare și transmis prin 2 biți de date. Atunci când o piesă ajunge în dreptul clapetei corespunzătoare tipului său (ex. piesă de tip 1 în dreptul clapetei 1), banda transportoare trebuie oprită iar piesa evacuată prin deschiderea și închiderea clapetei. Comanda clapetei se face printr-o singură comandă digitală, care trebuie menținută o perioadă de 5 secunde pentru a asigura deschiderea respectiv închiderea clapetei.

*Elemente de execuție:*

- motorul benzii transportoare
- cele 4 clapete

*Elemente de măsură:*

- un dispozitiv de identificare piesă

- 5 senzori de prezență piesă
- 1 buton

## 2. Soluția de automatizare

Pentru controlul acestei aplicații se alege un automat programabil de tip PEP Smart pentru care se dezvoltă un proiect ISaGRAF, ce cuprinde 5 programe în secțiunea sequential, așa cum este prezentat și în figura 10.2.

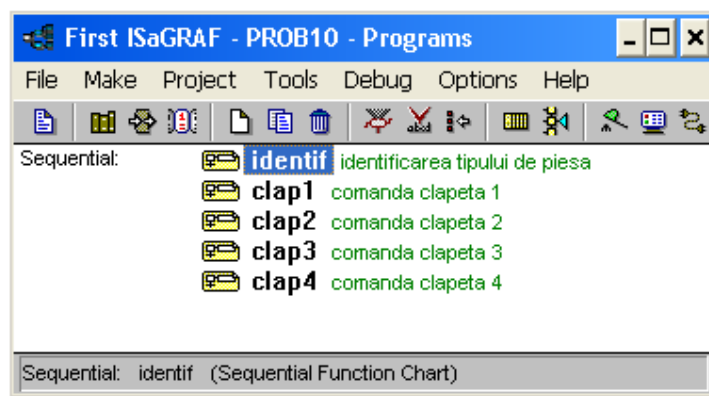


Fig.10.2. Structura proiectului ISaGRAF

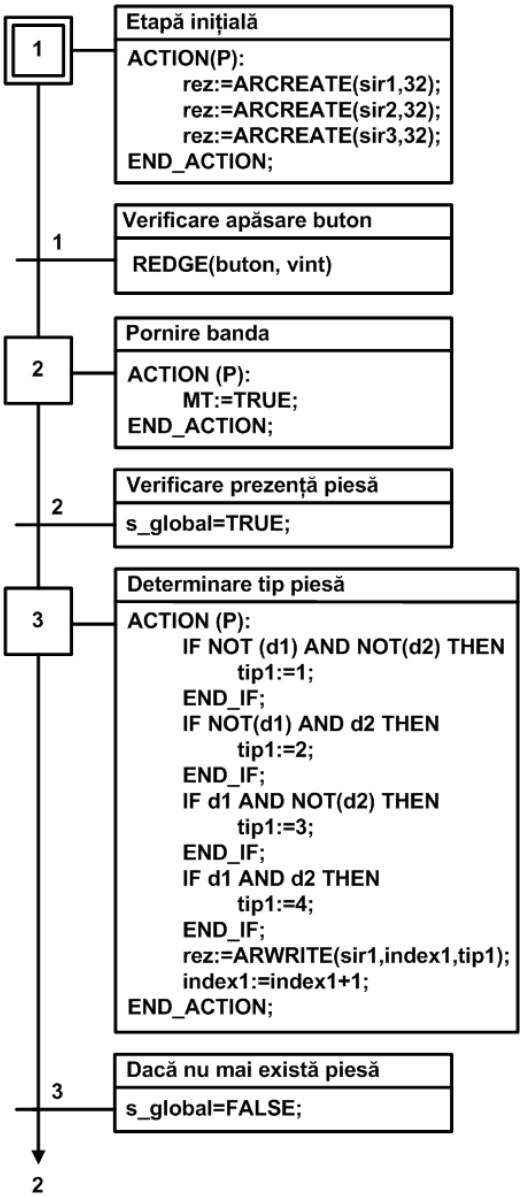
*Dicționarul de variabile globale:*

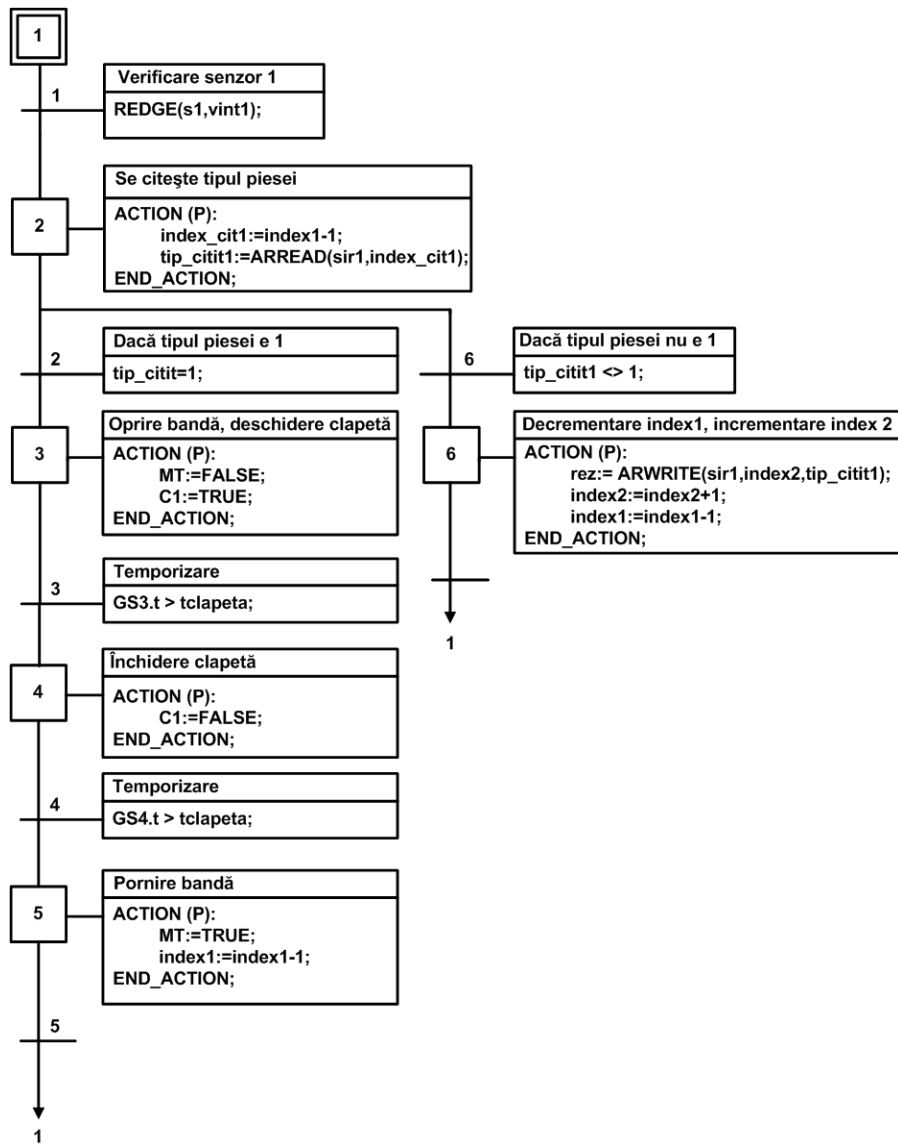
- *Variabile de intrare booleene:*
  - **s\_global**: atașată senzorului de prezență din poziția de identificare;
  - **d1**: prima linie de la dispozitivul de identificare;
  - **d2**: a doua linie de la dispozitivul de identificare;
  - **s1, s2, s3, s4**: atașate senzorilor de prezență
  - **buton**: atașată butonului
- *Variabile de ieșire booleene:*
  - **MT**: comandă banda transportoare;
  - **C1, C2, C3, C4**: comandă deschiderea/ închiderea clapetelor;
- *Variabile interne booleene:*
  - **vint, vint1, vint2, vint3, vint4**: folosite de funcțiile REDGE

- *Variabile timer:*
  - const: **tclapeta** = t#2s;
- *Variabile interne de tip integer:*
  - variabile: **rez, tip1, tip2, tip3, index1, index2, index3, index\_cit1, index\_cit2, index\_cit3, tip\_citit1, tip\_citit2, tip\_citit3**
  - constante: **sir1=1; sir2=2; sir3=3;**

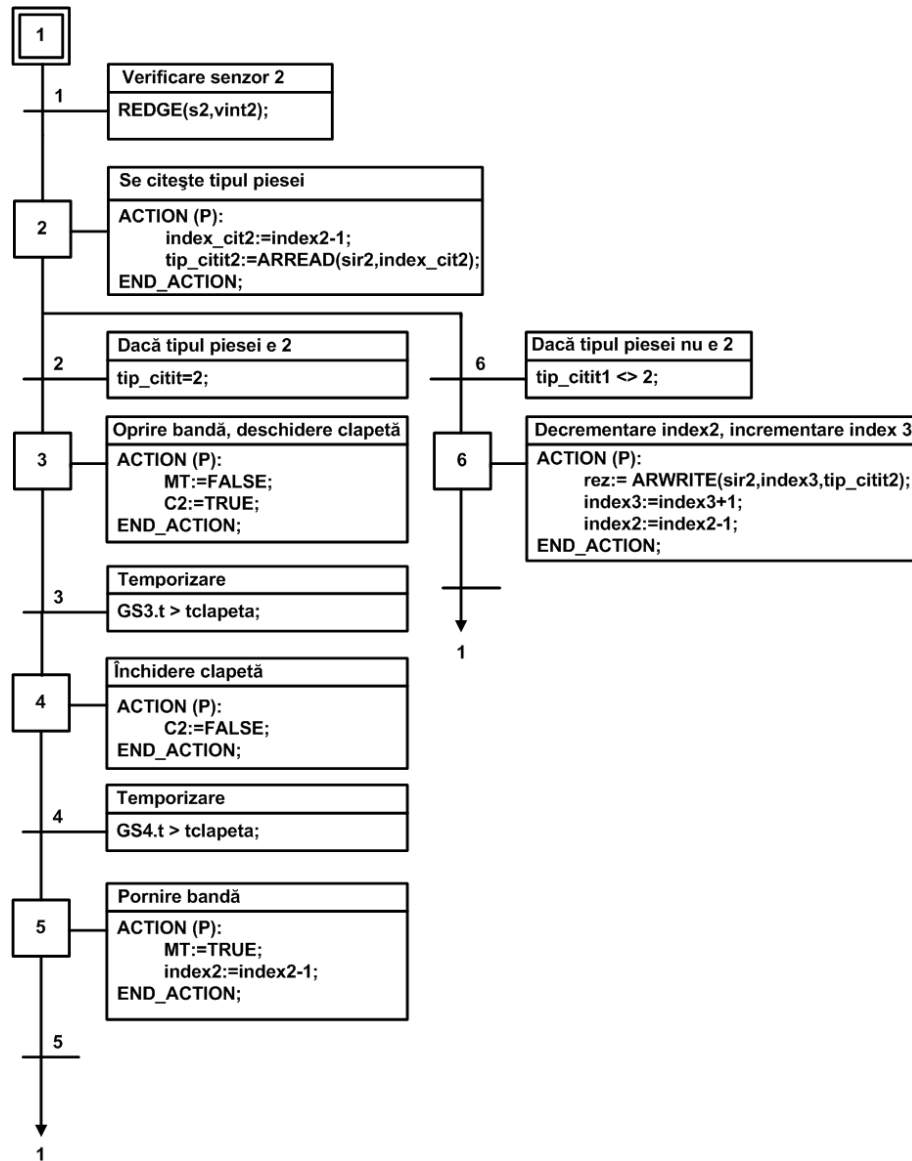


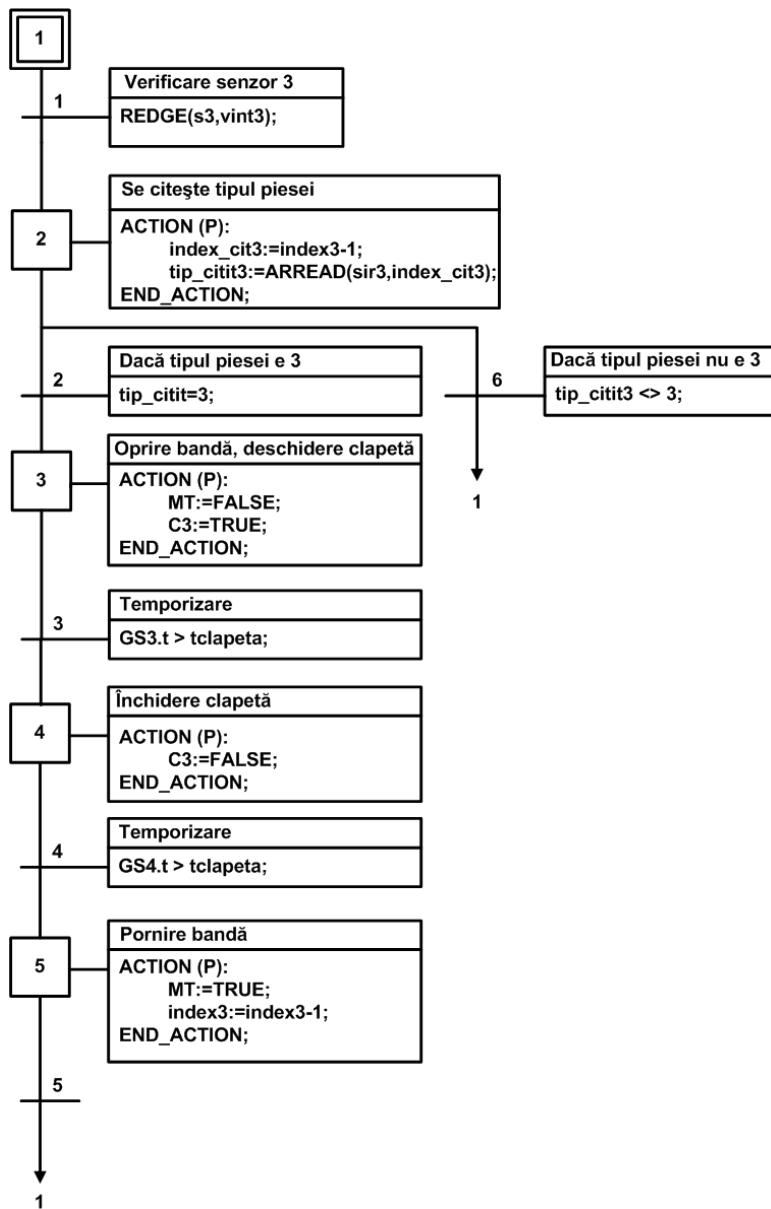
Program identifi:

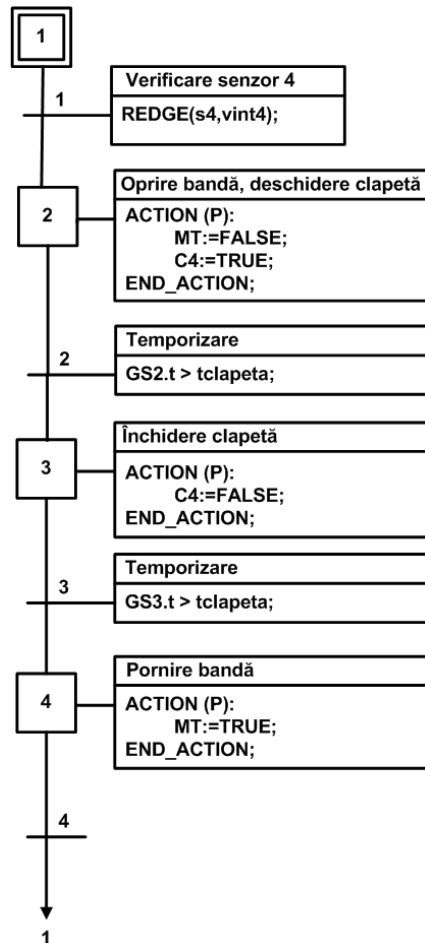


**Program Clapeta1:**

## Program clapeta2:



**Program clapeta3:**

**Program clapeta4:***Observații:*

- Soluția adoptată pentru a putea determina exact ce tip de piesă ajunge în dreptul senzorilor a fost de a crea 3 șiruri de numere întregi, pentru primele 3 clapete (pentru a patra nu mai este necesar deoarece aici nu pot ajunge decât piese de tipul 4). Fiecare șir este modificat când o piesă ajunge în dreptul senzorului de prezență anterior. Există câte o variabilă *index* asociată fiecărui șir care indică prime poziție liberă din șir. Șirurile sunt de tip FIFO, tipul piesei care se găsește în dreptul unui senzor este citit întotdeauna de pe

prima poziție a șirului asociat. În cazul în care tipul piesei nu corespunde locației, valoarea este scoasă din șirul respectiv (prin shiftarea șirului) și introdusă în prima poziție liberă din șirul asociat clapetei următoare.

*Propuneri:*

- Să se modifice programul pentru situația în care nu mai există senzori de prezență în dreptul clapetelor, cunoscându-se însă viteza benzii transportoare și distanțele între poziția de identificare și cele 4 clapete.

## Problema 11: Umplerea automată a unor containere

### 1.Descrierea procesului

Aplicația constă în umplerea cu lichid a 3 containere (A, B, C) și evacuarea lor pe o bandă transportoare. Umplerea containerelor trebuie făcută în următoarea manieră:

- containerul A: 5 secunde cu lichid de tip A;
- containerul B: 7 secunde cu lichid de tip A și 7 secunde cu lichid de tip B;
- containerul C: 3 secunde cu lichid de tip C, 5 secunde cu lichid de tip B și 8 secunde cu lichid de tip A.

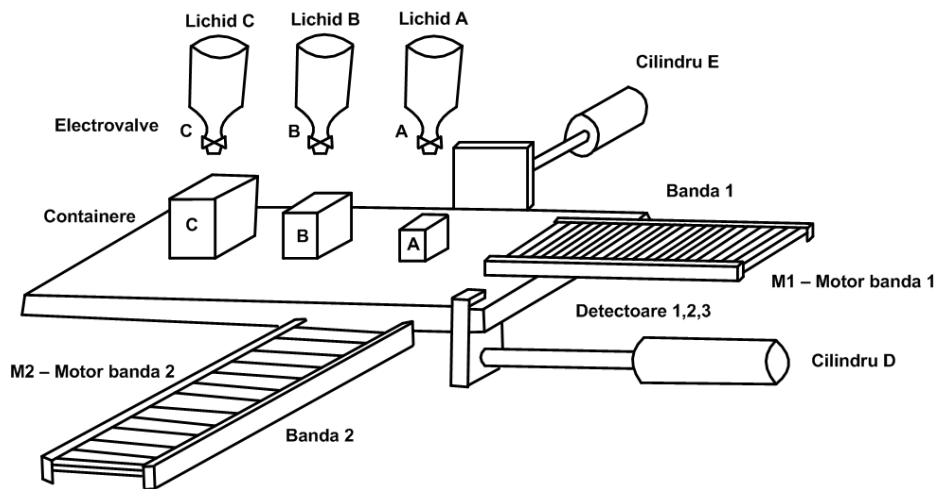


Fig.11.1. Umplerea automată a unor containere

În cadrul sistemului există o bandă transportoare 1 pe care vin, unul după altul, cele trei containere A, B, C. Primul dintre ele care ajunge la platformă este containerul C, apoi B și ultimul cel de tip A. Cilindru E este responsabil cu evacuarea recipientelor cu ajutorul celei de-a doua benzi transportoare.

Inițierea procesului se face prin pornirea benzii transportoare 1 pe care sunt aduse containerele. În momentul în care un container de tip C se găsește pe platformă, banda 1 va fi oprită iar cilindrul D va avansa o poziție.

Când containerul C activează detectorul 2, banda 1 va fi din nou activată iar cilindrul D va fi oprit; banda 1 se va opri din nou când containerul B ajunge la platformă și în consecință cilindrul D va avansa din nou până când containerul C activează detectorul 3 iar containerul B activează detectorul 2. În acest moment banda 1 este repornită până când containerul A atinge platforma, moment în care banda este oprită. În acel moment cele trei valve vor fi deschise simultan, fiecare fiind menținută deschisă un anumit timp, astfel încât containerul A se va umple cu lichid A timp de 5 secunde, containerul B cu lichid de tip B timp de 7 secunde iar containerul C timp de 3 secunde cu lichid de tip C. Când toate aceste temporizări au expirat, valvele vor fi închise, cilindrul E va avansa pentru a evacua containerul A până activează detectorul 4. În acest moment cilindrul E se retrage. După ce a ajuns în poziția de retragere, cilindrul D va fi retras până activează detectoarele 1 și 2. Apoi containerele B și C vor fi umplute cu lichid de tip A, respectiv B, după care urmează evacuarea containerului B. În final containerul C va fi umplut cu lichid de tip A și va fi evacuat. După evacuare cilindrul D va fi retras și un nou ciclu poate începe.

*Elemente de execuție:*

- 3 electrovalve;
- 2 cilindri cu dublu efect;
- 2 motoare ale benzilor transportoare.

*Elemente de măsură:*

- 4 detectoare de poziție.

## 2. Soluția de automatizare

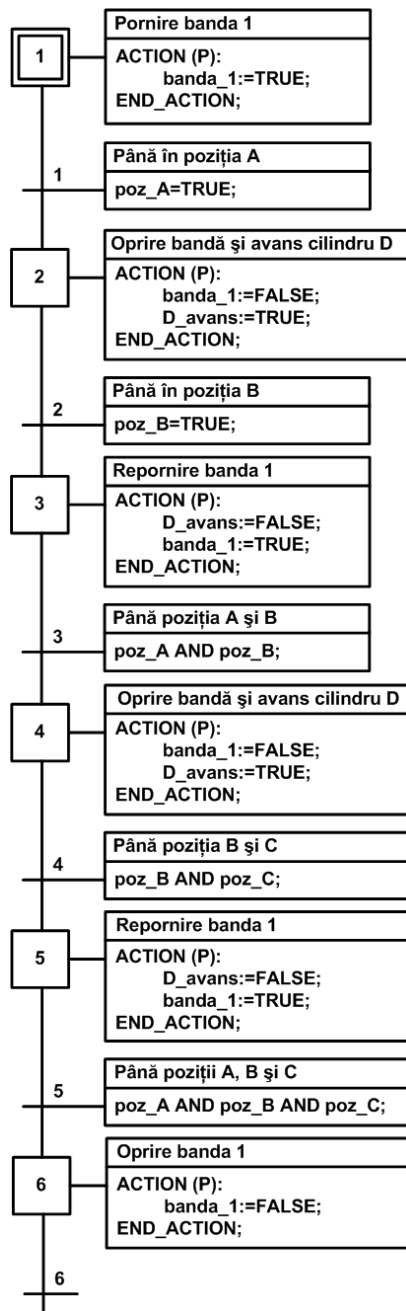
Pentru controlul acestei aplicații s-a ales un automat programabil de tip PEP Smart pentru care s-a dezvoltat un proiect ISaGRAF ce cuprinde un program principal.

*Dicționarul de variabile globale:*

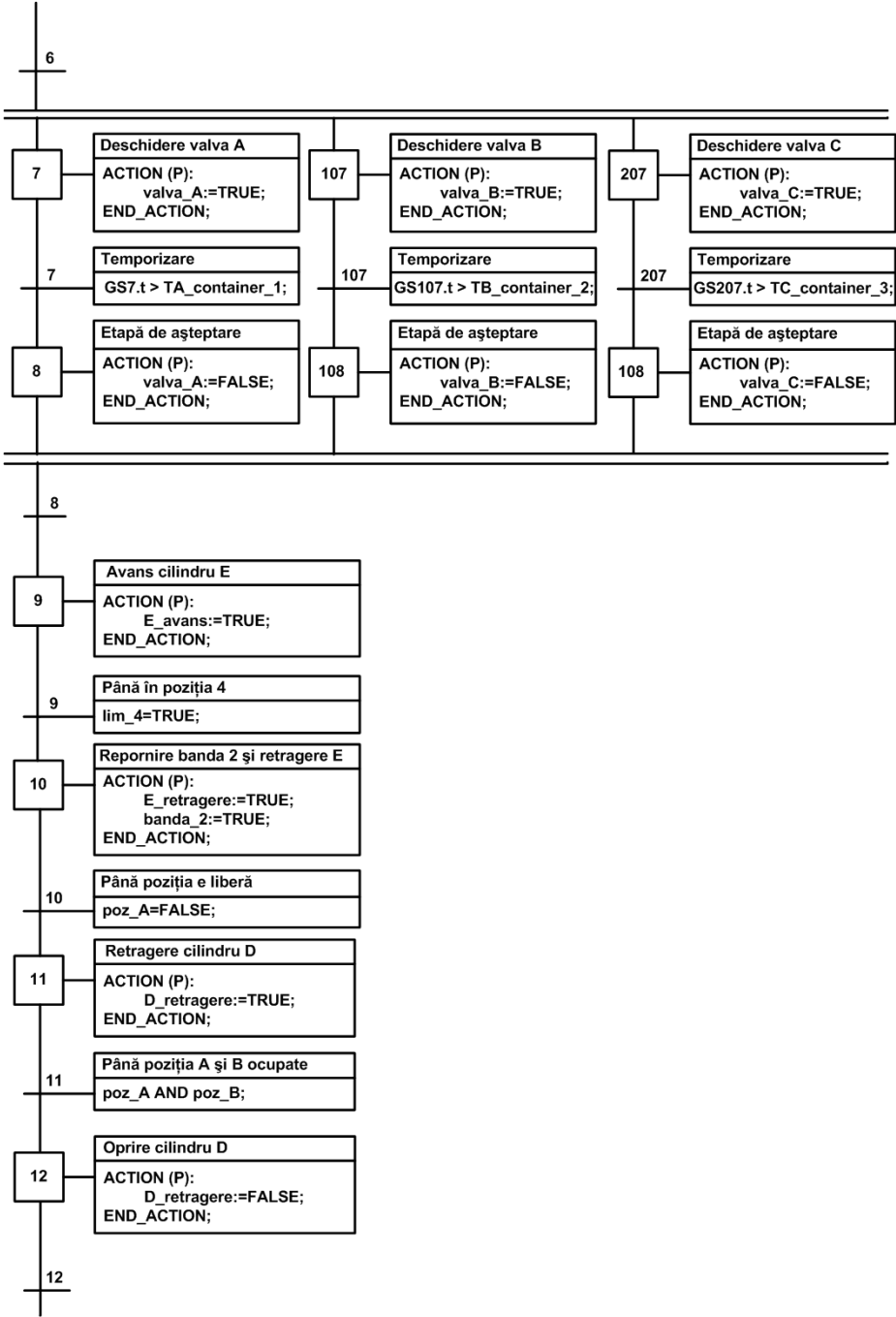
- *Variabile de intrare booleene:*
  - **poz\_A**: detectorul de poziție A;
  - **poz\_B**: detectorul de poziție B;
  - **poz\_C**: detectorul de poziție C;
  - **lim\_4**: limitatorul de avans al cilindrului E.
- *Variabile de ieșire booleene:*
  - **banda\_1**: activare / dezactivare banda 1;

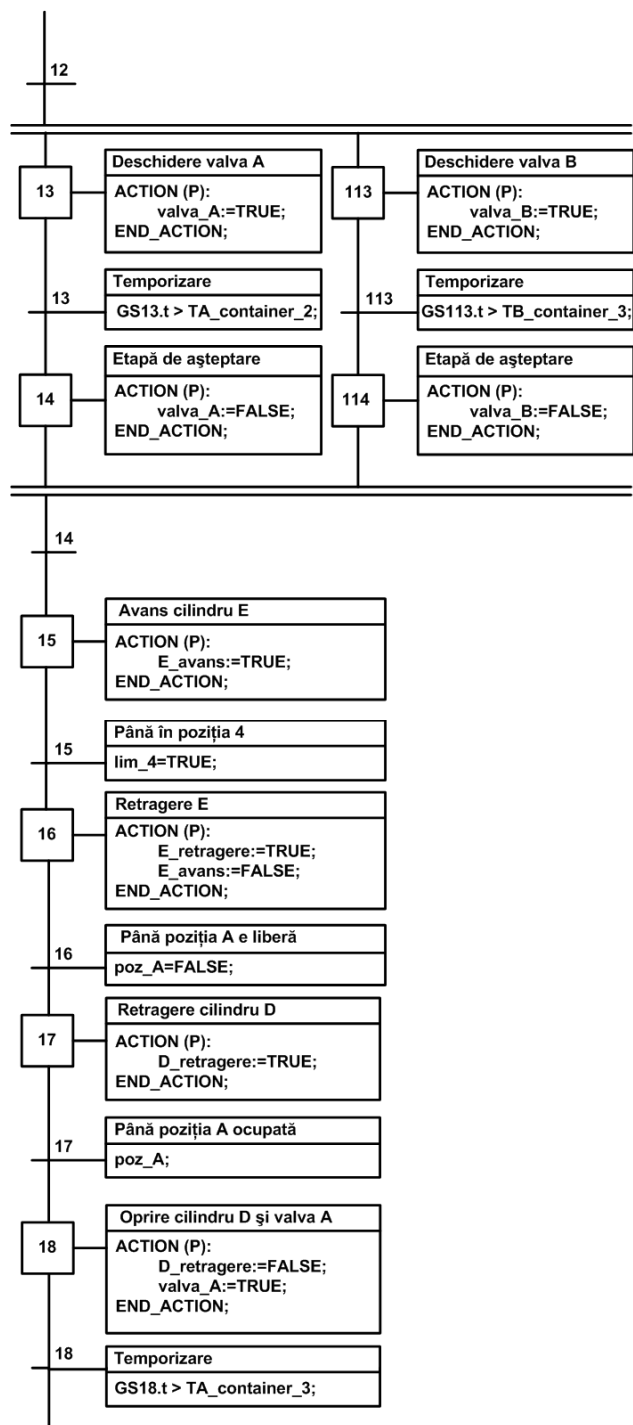


- **banda\_2**: activare / dezactivare banda 2;
- **D\_avans**: avans cilindru D;
- **D\_retragere**: retragere cilindru D;
- **E\_avans**: avans cilindru E;
- **E\_retragere**: retragere cilindru E;
- **valva\_A**: comanda de deschidere/închidere valva A;
- **valva\_B**: comanda de deschidere/închidere valva B
- **valva\_C**: comanda de deschidere/închidere valva C;
- *Constante de tip Timer:*
  - **TA\_container\_1**: perioada de timp pentru umplerea containerului 1 cu lichid de tip A
  - **TB\_container\_2**: perioada de timp pentru umplerea containerului 2 cu lichid de tip B
  - **TC\_container\_3**: perioada de timp pentru umplerea containerului 3 cu lichid de tip C
  - **TA\_container\_2**: perioada de timp pentru umplerea containerului 2 cu lichid de tip A
  - **TB\_container\_3**: perioada de timp pentru umplerea containerului 3 cu lichid de tip B
  - **TA\_container\_3**: perioada de timp pentru umplerea containerului 3 cu lichid de tip A

**Programul main:**

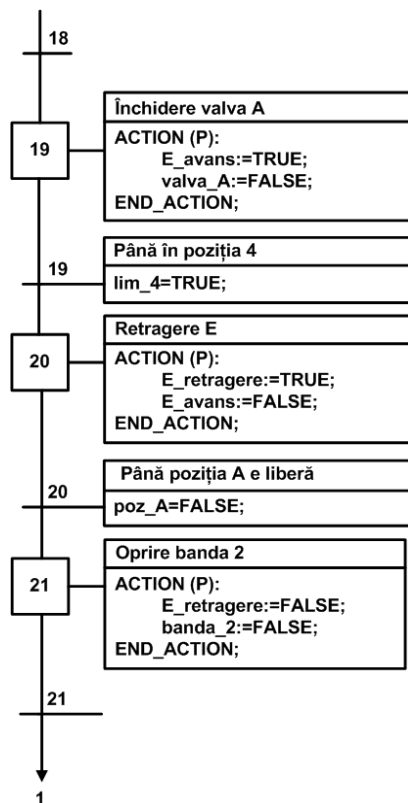
97 Problema 11- Umplerea automată a unor containere





99

## Problema 11- Umplerea automată a unor containere

*Observații:*

- În cadrul secvențelor de deschidere a valvelor, temporizările au fost implementate prin testarea permanentă a parametrului GSxxx.t asociat unei etape. Acest parametru indică timpul de când o etapă este activă. O altă variantă era folosirea unor variabile de tip Timer.
- Umplerea containerelor este făcută în paralel cu ajutorul paralelismului diagramelor SFC.

## Problema 12: Controlul unor uși automate aflate la intrarea într-o incintă

### 1. Descrierea aplicației

Elementele sistemului de controlat sunt:

- două uși, fiecare dintre ele mișcată cu ajutorul a câte două motoare cu două sensuri de rotație (mișcare stânga – dreapta pentru fiecare ușă);
- câte doi senzori de capăt cursa (deschis – închis) pentru fiecare ușă;
- un senzor de prezență persoană în dreptul ușii.

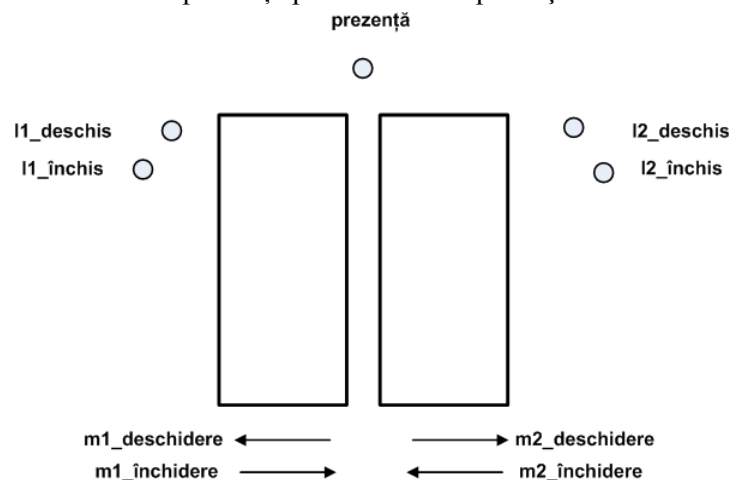


Fig.12.1. Controlul unor uși automate aflate la intrarea într-o incintă

Funcționarea dorită a sistemului este: inițial ușile sunt închise. Dacă o persoană apare în dreptul ușilor, ușile vor începe să se deschidă. Mișcarea de deschidere este menținută atâta timp cât, pentru fiecare ușă în parte, senzorul respectiv de deschidere este inactiv.

Dacă ambele uși sunt deschise și nu mai există nicio persoană în dreptul ușii timp de 2 secunde, ușile încep simultan să se închidă. Închiderea ușilor se oprește când ambele ajung la capăt de cursă sau apare o persoană în dreptul ușilor, în timp ce acestea se închideau. Ele își vor opri închiderea și vor începe să se deschidă.

## 101 Problema 12 – Controlul unor uși automate aflate la intrarea într-o incintă

*Elemente de execuție:*

- Cele 4 motoare ale ușilor.

*Elemente de măsură:*

- 4 limitatoare de cursă;
- un senzor de prezență persoană.

### 2. Soluția de automatizare

Pentru controlul acestei aplicații s-a ales un automat programabil de tip PEP Smart pentru care s-a dezvoltat un proiect ISaGRAF ce cuprinde un program principal în secțiunea Sequential vezi fig. 12.2.

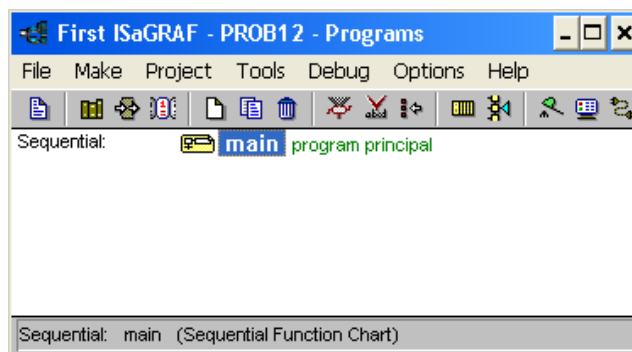
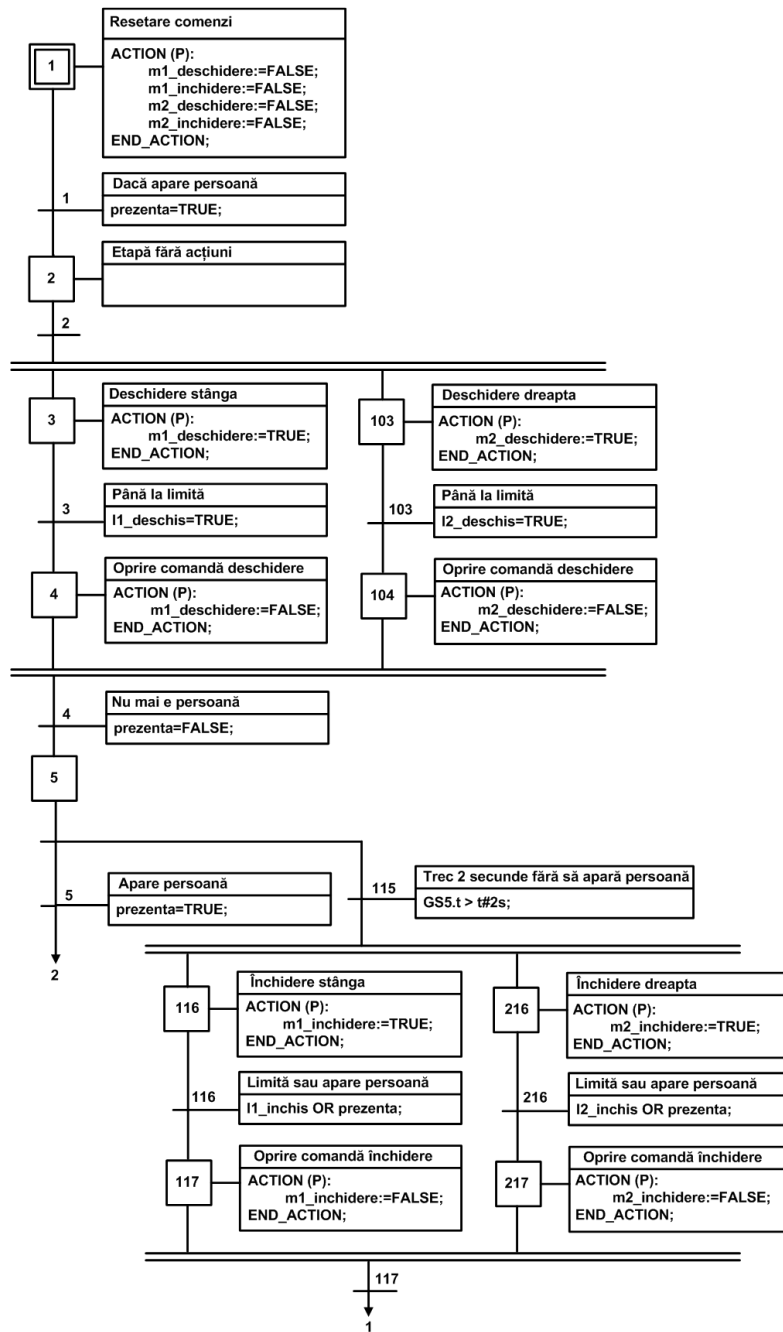


Fig.12.2.Structura proiectului ISaGRAF

*Dicționarul de variabile globale:*

- *Variabile de intrare booleene:*
  - **prezenta**: senzorul de prezență persoană;
  - **l1\_deschis**: limitatorul de ușă 1 deschisă;
  - **l1\_inchis**: limitatorul de ușă 1 închisă;
  - **l2\_inchis**: limitatorul de ușă 2 deschisă;
  - **l2\_deschis**: limitatorul de ușă 2 închisă.
- *Variabile de ieșire booleene:*
  - **m1\_deschidere**: comanda de deschidere ușă 1;
  - **m1\_inchidere**: comanda de închidere ușă 1;
  - **m2\_deschidere**: comanda de deschidere ușă 2;
  - **m2\_inchidere**: comanda de închidere ușă 2.

**Program main:**



**103**      *Problema 12 – Controlul unor uși automate aflate la intrarea într-o incintă*

*Observații:*

- Închiderea și deschiderea ușilor trebuie făcută neapărat într-un paralelism deoarece acțiunile încep în același moment, se desfășoară independent, dar procesul poate continua numai după ce ambele secvențe s-au încheiat.
- Etapa 2, aparent inutilă deoarece nu are acțiuni asociate, a fost introdusă pentru a putea intra în paralelismul de deschidere a ușilor atunci când condiția 5 devine adevărată.

## Problema 13: Sortarea a două tipuri de piese pe o bandă transportoare

### 1. Descrierea instalației și a procesului

Elementele sistemului sunt:

- o bandă transportoare acționată de un motor cu un singur sens de rotație;
- 3 senzori de prezență piesă;
- 1 senzor de detecție tip piesă (activ = piesa de tip 1, inactiv = piesa de tip 2);
- 2 pistoane care pot avansa și se pot retrage;
- 4 limitatoare de cursă, câte 2 pentru fiecare piston.

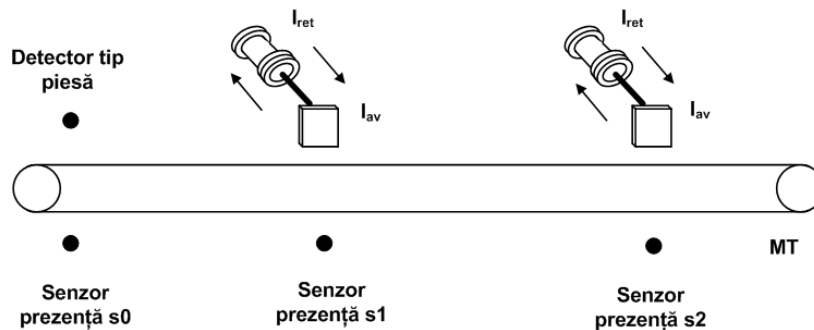


Fig. 13.1. Sortarea a două tipuri de piese

Pe o bandă transportoare vin piese de două tipuri. Inițial pe bandă nu se află nicio piesă iar pistoanele sunt retrase. Banda transportoare trebuie pornită o dată cu aplicația. Dacă o piesă ajunge în dreptul senzorului de prezență s0, banda trebuie oprită iar după o secundă de așteptare pentru stabilizarea valorii semnalului senzorului de detecție tip piesă, piesa poate fi identificată. Semnalul senzorului de identificare este valid numai când o piesă se găsește în dreptul lui s0. După identificarea piesei banda va reporni.

Se presupune că doar două piese pot exista în porțiunea de bandă de sortare, astfel încât dacă o a treia piesă apare în dreptul lui s0 în timp ce piesele anterioare sunt încă neevacuate, banda se va opri, va fi activată o

## 105 Problema 13 – Sortarea a două tipuri de piese pe o bandă transportoare

alarmă iar aplicația se va relua prin apăsarea și ridicarea unui buton special de reset.

O piesă va fi evacuată dacă ajunge în dreptul senzorului corespunzător tipului său. Evacuarea unei piese începe prin oprirea benzii, apoi pistonul respectiv avansează și se retrage. După încheierea retragerii pistonului, banda va fi repornită.

*Elemente de execuție:*

- un motor cu un singur sens de rotație ce acționează banda transportoare;
- două pistoane care pot avansa și se pot retrage;
- un dispozitiv de semnalizare în caz de eroare.

*Senzori și elemente de măsură:*

- 3 senzori de prezență piesă;
- 4 limitatoare de cursă pentru pistoane;
- un dispozitiv de identificare tip piesă;
- un buton de reset.

## 2. Soluția de automatizare

Pentru controlul acestei aplicații s-a ales un automat programabil de tip PEP Smart pentru care s-a dezvoltat un proiect ISaGRAF ce cuprinde trei programe în secțiunea Sequential conform figurii 13.2.

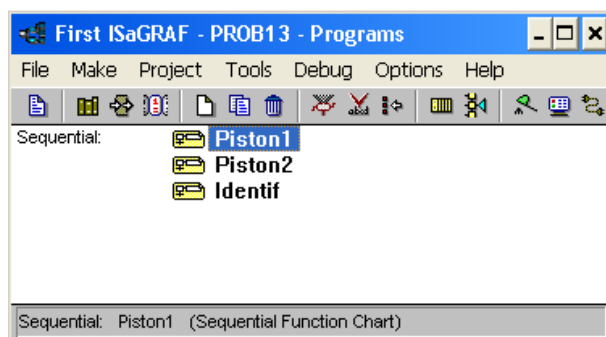


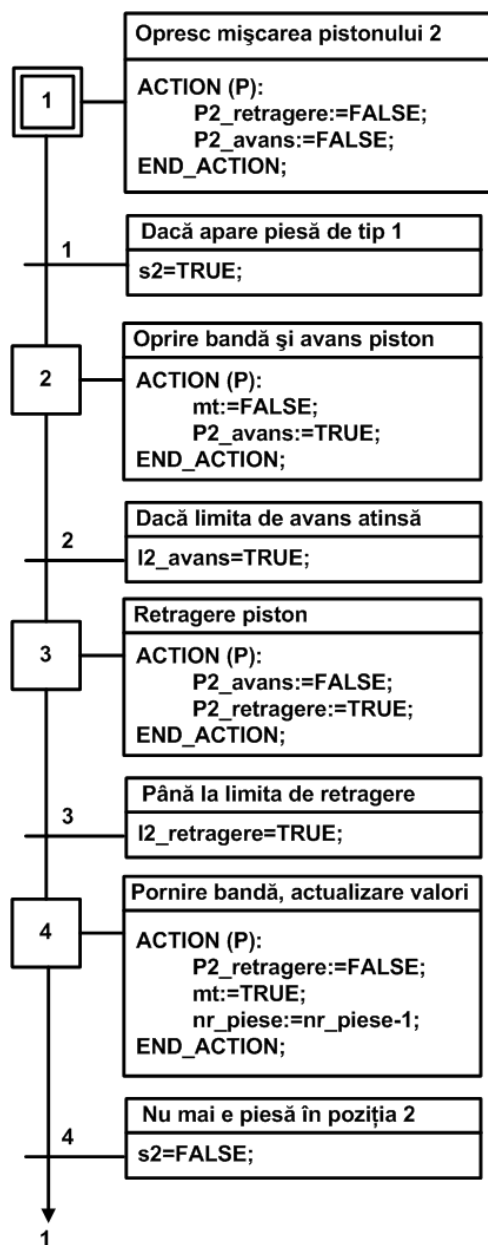
Fig. 13.2. Structura proiectului ISaGRAF

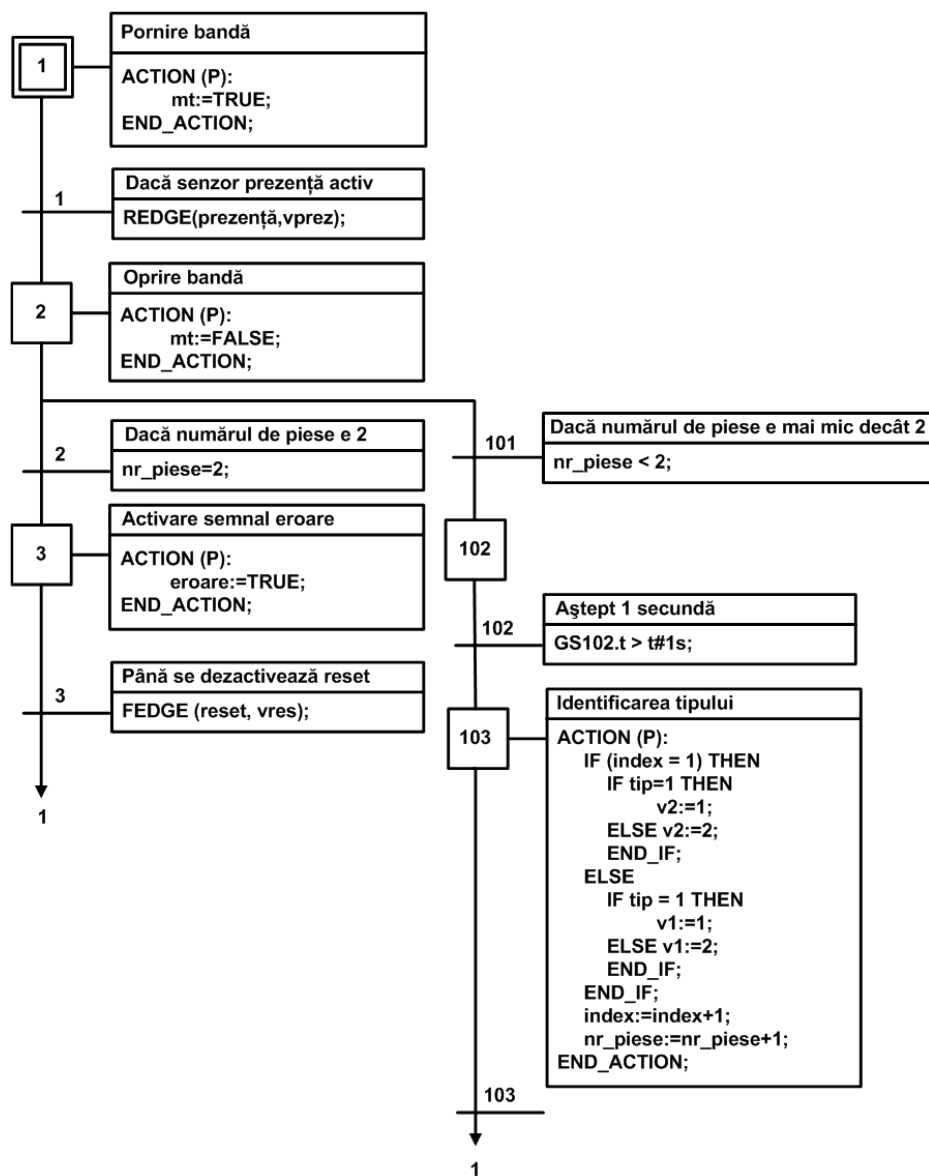
*Dicționarul de variabile globale:*

- *Variabile de intrare booleene:*
  - **tip**: semnal de la dispozitivul de identificare;

- **prezenta**: semnalul de la senzorul de prezență din postul de identificare piesă;
  - **s1**: semnalul senzorului din poziția 1;
  - **s2**: semnalul senzorului din poziția 2;
  - **I1\_avans**: semnalul limitatorului de avans al pistonului 1;
  - **I2\_avans**: semnalul limitatorului de avans al pistonului 2;
  - **I1\_retragere**: semnalul limitatorului de retragere al pistonului 1;
  - **I2\_retragere**: semnalul limitatorului de retragere al pistonului 2;
  - **reset**: semnalul de la butonul de reset.
- *Variabile de ieșire booleene:*
    - **mt**: comanda de pornire/ oprire a motorului benzii transportoare;
    - **P1\_avans**: comandă avans piston 1;
    - **P1\_retragere**: comandă retragere piston 1;
    - **P2\_avans**: comandă avans piston 2;
    - **P2\_retragere**: comandă retragere piston 2;
    - **eroare**: comanda semnalului de eroare.
- Variabile interne întregi:
    - **v1**: folosită pentru memorarea tipului primei piese ce urmează a fi evacuată de către pistonul 1;
    - **v2**: folosită pentru memorarea tipului celei de-a doua piese ce urmează a fi evacuată de către pistonul 1;
    - **index**: conține numărul de piese aflate pe bandă înainte de poziția 1;
    - **nr\_piese**: conține numărul de piese aflate pe bandă în drum către pozițiile de evacuare.
- Variabile interne booleene:
    - **vprez**: necesară funcției REDGE;
    - **vres**: necesară funcției FEDGE;



**Program piston2:**

**Program Identif:****Observații:**

- Pentru a determina corect tipul piesei care ajunge în dreptul pistonului 1, au fost folosite 3 variabile: index, v1 și v2;

- În programul *Identif* se observă folosirea funcției FEDGE, necesară determinării momentului când variabila reset trece din TRUE în FALSE (corespunde ridicării butonul după ce a fost apăsat).

***Propuneri:***

- Să se modifice programul pentru situația în care numărul de piese permis pe bandă este oarecare  $n$ ;



## Problema 14: Procesul de coacere al biscuiților

### 1. Descrierea instalației și a procesului

Elementele care compun instalația sunt:

- un cuptor de ardere conținând:
  - un traductor de temperatură ce transmite un semnal analogic în gama  $4 \div 20$  mA, corespunzător unei game de temperaturi între  $0 \div 500^{\circ}\text{C}$ ;
  - un injector de gaz, comandat digital (pornește și oprește flacăra);
- o banda transportoare
  - un senzor prezență biscuit, activ atât timp cât prin dreptul său trece un biscuit;
  - un senzor de apreciere calitate biscuit, activ doar dacă biscuitul este rebut;
  - un macaz care direcționează biscuiții, comandat simplu digital (inactiv – traseu biscuit bun, activ – traseu biscuit rebut);

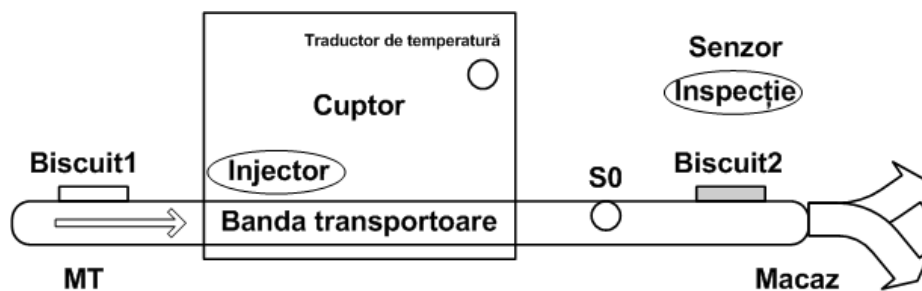


Fig.14.1. Cuptor de ardere

Inițial banda transportoare trebuie pornită. Pe ea vin biscuiți. În interiorul cuptorului temperatura trebuie menținută între două valori  $T_{min}$  ( $230^{\circ}\text{C}$ ) și  $T_{max}$  ( $290^{\circ}\text{C}$ ).

În momentul în care un biscuit trece de senzorul de prezență, trebuie testată calitatea lui. Dacă este rebut, macazul va trece pe traseul rebut pentru un timp de 5 secunde. Dacă vin mai mult de 5 biscuiți rebut consecutiv, prima dată se shiftază domeniul de temperatură cu 10% (se crește  $T_{min}$  și

Tmax cu 10%) dacă se mai întâmplă și după aceasta, banda transportoare va fi oprită și tot procesul se va relua doar la apăsarea unui buton de repornire. La fel și dacă temperatura scade sub Tmin\_abs (180°C).

## 2. Soluția de automatizare

Pentru controlul acestei aplicații s-a ales un automat programabil de tip PEP Smart pentru care s-a dezvoltat un proiect ISaGRAF ce cuprinde un program părinte și două programe fiu (biscuit și temper) în secțiunea Sequential, plus un program ciclic în secțiunea Begin ce realizează conversia din unități CAN în unități de măsură ingineresti.

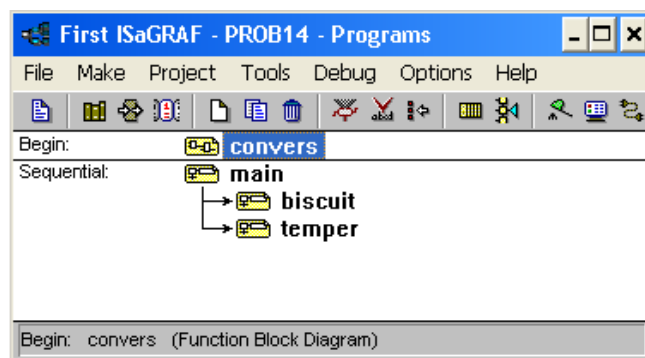


Fig.14.2. Structura proiectului ISaGRAF

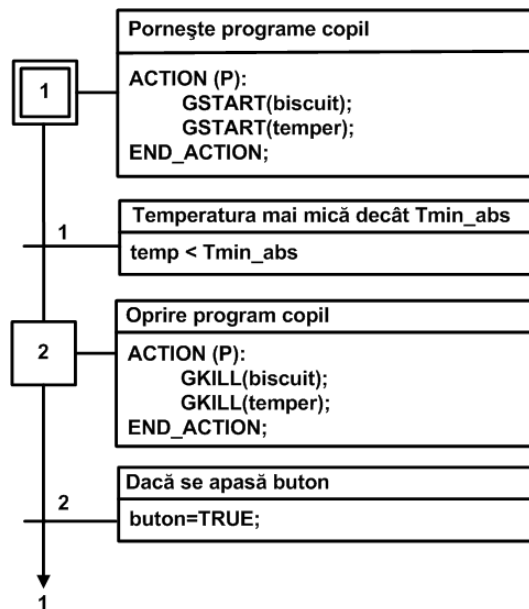
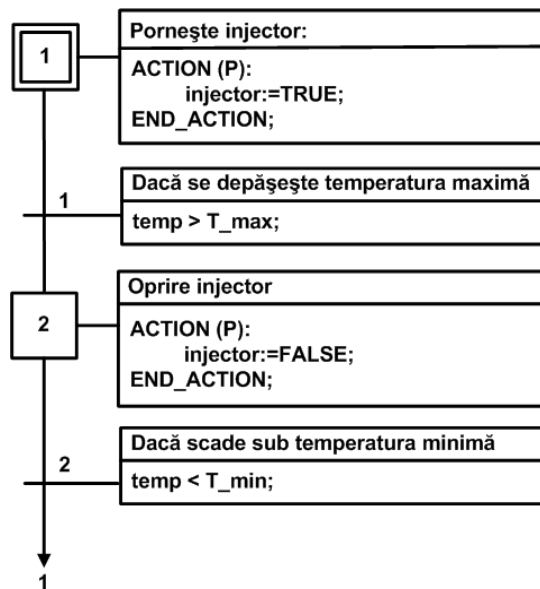
*Dicționarul de variabile globale:*

- *Variabile de intrare booleene:*
  - **buton**: atasata butonului de repornire ciclu
  - **s0**: atasata senzorului de prezență
  - **inspectie**: atasata senzorului de inspecție biscuiți
- *Variabile de ieșire booleene:*
  - **MT**: comanda motorului benzii transportoare
  - **Injector**: comanda injectorului
  - **Macaz**: comanda macazului
- *Variabile de intrare analogice*
  - **Tempcit**: atașată traductorului de temperatură

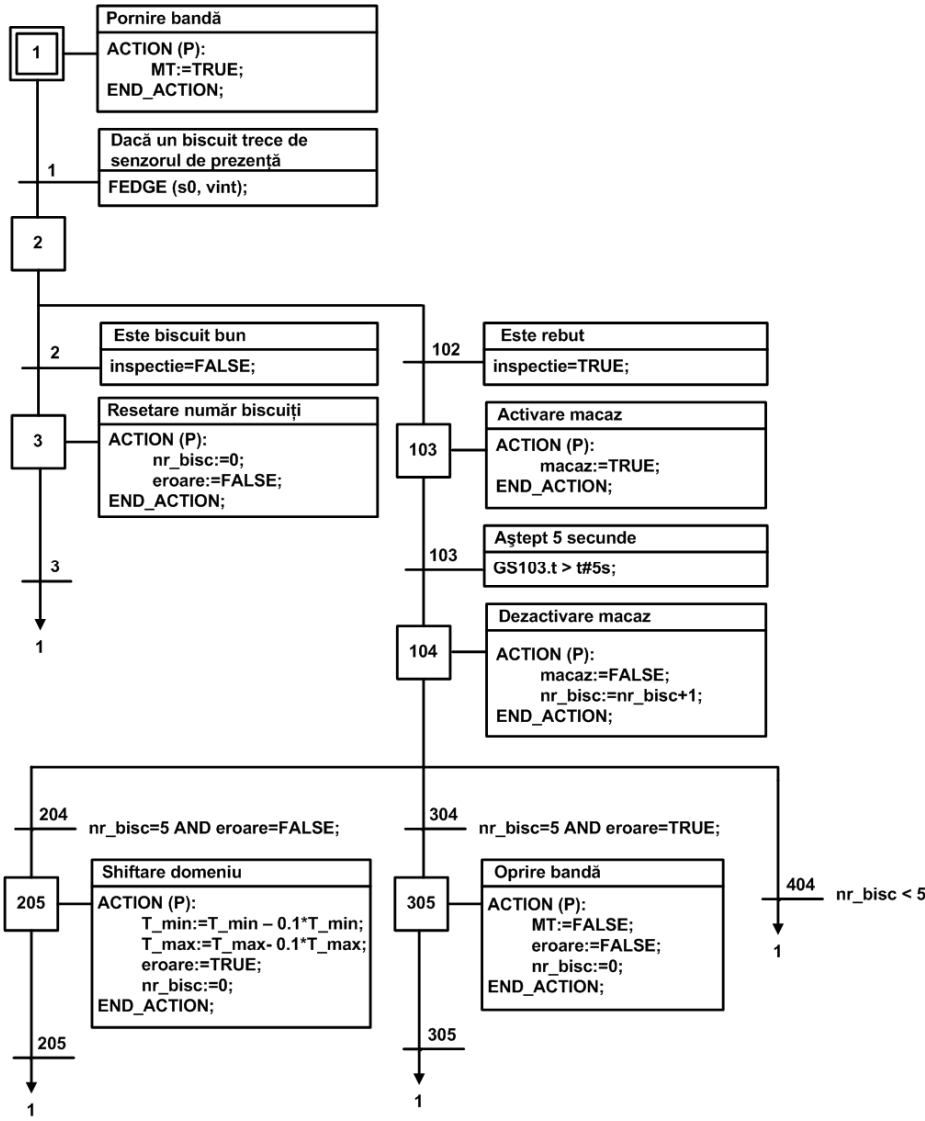
113

*Problema 14 - Procesul de coacere al biscuiților*

- *Variabile interne booleene*
  - **Eroare**: True dacă de 2 ori consecutiv vin 5 bisuciți rebut
  - **Vint**: necesară funcției FEDGE
- *Variabile interne analogice*
  - **Temp**: valoarea curentă a temperaturii, in grade
  - **Tmin, Tmax**: valorile limită ale gamei de temperatură
  - const **Tmin\_abs** = 180

**Program main:****Program temper:**

Program biscuit:



Program convers:

temp:= tempcit \*100/ 4096

*Observații:*

- S-a ales soluția unui proiect cu un program părinte și doi fii deoarece programele trebuie oprite în cazul apariției unui anumit eveniment (aici temperatura < temperatura minimă absolută), indiferent ce etape sunt active la momentul respectiv. O altă soluție, mult mai neelegantă și care ar complica diagramele foarte mult, ar fi testarea evenimentului în toate etapele unui diagrame SFC.
- Testarea faptului că o serie de 5 biscuiți rebut apare de 2 ori la rând s- făcut prin gestionarea unei singure variabile booleene, *eroare*.
- S-a folosit funcția FEDGE (Falling Edge Detection) deoarece trebuie testată tranziția stării senzorului s0 din activă în inactivă.