# PROIECTAREA SISTEMELOR DE CONDUCERE CU AUTOMATE PROGRAMABILE

**Prof. dr. ing. SORIN LARIONESCU – UTCB, larionescu@home.ro**

Pentru a prezenta câteva caracteristici a unei metode de proiectare pe care o folosesc în cazul sistemelor de conducere cu automate programabile trebuie, la început, să fac o delimitare a automatelor programabile de celelalte calculatoare care sunt folosite în cazul unui sistem de domotică. Aceasta este cu atât mai necesar cu cât exista automate programabile implementate pe PC, aşa numitele Soft-PLC.

Automatele programabile (AP) pot fi considerate microcalculatoare specializate care funcţionează în timp real, adică asigură o limită maximă pentru durata procesului de achiziţie, prelucrare şi redare a informaţiilor. Arhitectura tipică a unui AP este prezentată în Fig. 1. Se observă rolul important al intrărilor/ieşirilor logice şi analogice si al posibilităţilor de extensie.

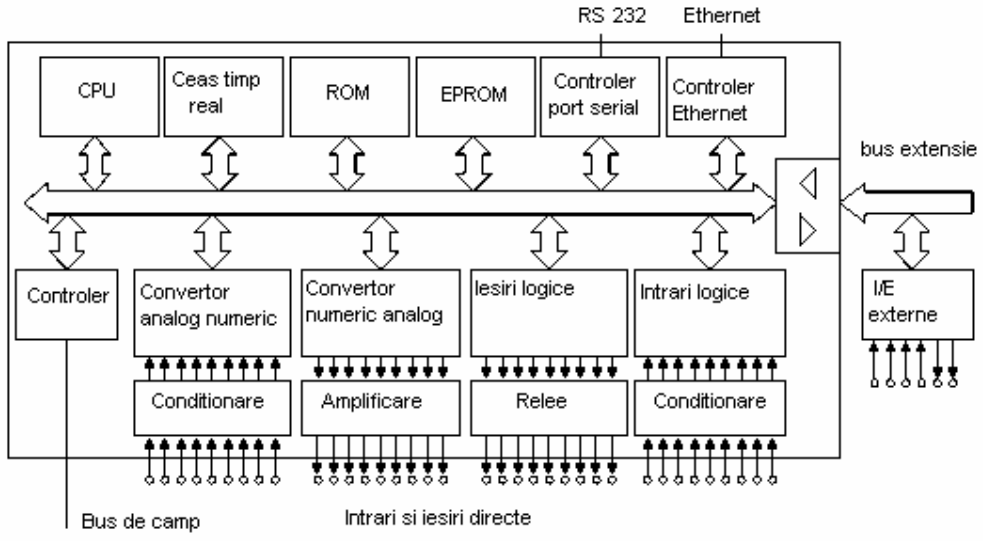


*Fig. 1*

Intr-un sistem de domotică, Fig. 2, pot exista la nivelul de management şi nivelul de automatizare calculatoare care nu îndeplinesc condiţia de funcţionare în timp real, de exemplu diversele servere (web, baze de date, ftp, etc), gateway sau staţii clienţi. Sistemul lor de programare este diferit de cel pentru AP.

Deşi în Fig. 2 nu apar, aproximativ 20% dintre calculatoarele folosite în conducerea automată în timp real nu sunt de tip AP, principala deosebire constând în caracteristicile software care permit o programare mai elastică. Într-adevăr, schema de funcţionare pentru AP constă în scanarea intrărilor, executarea algoritmului de conducere, actualizarea ieşirilor şi realizarea operaţiilor de întreţinere. Proiectarea în această situaţie este concentrată asupra algoritmului de conducere deoarece ciclurile de intrare, ieşire şi întreţinere sunt ascunse. Celelalte tipuri de calculatoare folosite în conducerea automată, numite PAC (Programmable Automation Controller) oferă un acces mai profund la resursele hardware ale sistemului.

Şi din Fig. 2 se observă rolul important al intrărilor/ieşirilor şi comunicării prin magistrala (bus) de automatizare şi magistralele de câmp. Automatele programabile pot juca diferite roluri într-un sistem de domotică: interconectare directă I/E, Interconectare I/E prin intermediul magistralei de câmp sau gateway între magistrale.

Semnalele I/E logice şi analogice sunt prelucrate în mod diferit, Fig. 3 şi din această cauză şi metodele de proiectare sunt diferite. Eşantionarea cu o perioadă constantă este esențială atât pentru semnalele analogice cât şi pentru semnalele logice. In continuare mă voi referii numai la proiectarea sistemelor cu evenimente discrete care folosesc AP.

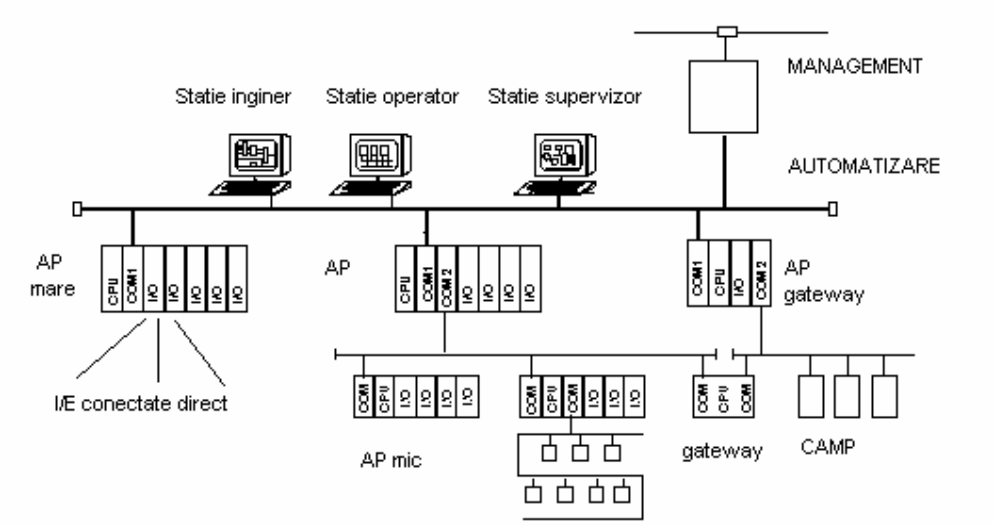


*Fig. 2*

Proiectarea unui sistem de conducere cu AP are cel puțin următoarele etape:

1. Identificarea procesului.
2. Stabilirea algoritmului de conducere şi a performanțelor necesare.
3. Programarea AP
4. Configurarea AP

Orice discuție privind proiectarea sistemelor de conducere cu automate programabile nu poate fi făcută în afara standardelor IEC 61131 şi IEC 61499. Standardul IEC 61131 are următoarele secțiuni:

- IEC 61131-1 Generalități
- IEC 61131-2 Testare
- IEC 61131-3 Programare şi tipuri de date
- IEC 61131-4 Ghidul utilizatorului
- IEC 61131-5 Comunicații
- IEC 61131-7 Conducerea Fuzyy

Evoluția standardelor referitoare la AP este prezentată în tabelul următor:

| **Anul** | **Standarde nationale** | **Standardul international** |
|---|---|---|
| 1977 | DIN 40 719-6 (scheme bloc)<br>GRAFCET (Franța) | IEC 848 |
| 1983 | DIN 19239 PLC programming<br>Allen Bradley: limbaje de programare pentru AP | |
| 1993 | DIN EN 661131 Part 3 | IEC 61131-3 |
| 1994 | DIN EN 661131 Parts 1 and 2 | |
| 1995 | | IEC 61131-4 |
| 2005 | | IEC 61499 Conducere distribuita |

O importanță deosebită are standardul IEC 61131-3 care prevede următoarele modele pentru programarea AP:

1. LD (Ladder Diagram) – schemă desfăşurată cu contacte şi relee
2. FBD (Function Block Diagram) – schema bloc
3. IL (Instruction List) – program tip assembler
4. ST (Structured Text) – program tip Pascal
5. SFC (Sequential Function Charts) –grafcet

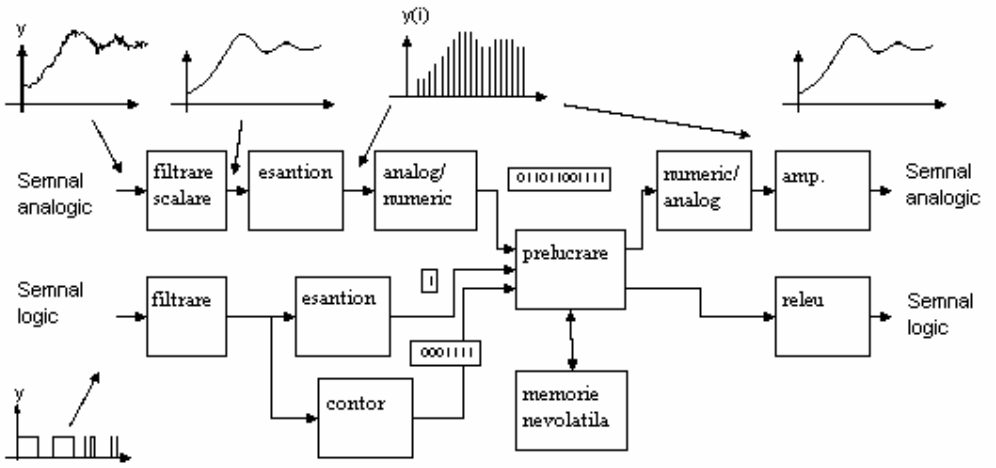


*Fig. 3*

Pentru programarea sistemelor de conducere distribuite cu AP s-a elaborat standardul IEC 61499. În *Fig. 4* se prezintă o schemă tehnologică simplă pentru un sistem de conducere distribuit format dintr-un PC, un AP, un regulator PID şi un robinet de reglare, toate interacţionând prin intermediul unei magistrale (bus). Un program sub formă de FBD conform IEC 61499 pentru acest sistem de conducere distribuit apare în Fig. 5. Între blocurile funcţionale există o legătură la nivelul fluxului de evenimente şi la nivelul fluxului de date. Fiecare bloc poate să aibă diferite intrări şi ieşiri cum ar fi referinţa SP şi ieşirea AUT pentru regulatorul PID. Blocul RESTART furnizează eveniment pentru execuţia periodică a celorlalte blocuri funcţionale.

În cadrul foarte general furnizat de aceste standarde vreau să prezint o modalitate de proiectare a sistemelor cu AP pe care am elaborat-o. Bineînţeles că metoda are multe limitări dar este mai simplă şi, lucru important, destul de algoritmizată, permiţând şi inginerilor mai puţin experimentaţi proiectarea unor sisteme de conducere cu performanţe bune.
În primul rând, modelul pentru sistemul de conducere este o reţea Petri conformă şi nu un grafcet cum prevede IEC 61131-3. Grafcetul este mai general şi mai elastic pentru construcţia de modele decât reţeaua Petri /11/. În schimb modelul Petri este mai uşor de analizat şi tradus sub forma unui model matematic care să stea la baza elaborării programului pentru AP. Stabilirea relaţiilor logice o fac cu ajutorul unor automate elementare cu prioritate la pornire, spre deosebire de soluţia recomandată în /8/ care foloseşte automate elementare cu prioritate la oprire. Soluţia mea este mai aproape de spiritul grafcetului şi evită erori care pot apare la elaborarea unor modele care conţin paralelisme de tip conveyor. Metoda de proiectare va fi prezentată în continuare pe baza exemplului automatului care porneşte un motor electric şi după o rotaţie îl opreşte în poziţia iniţială, Fig. 6.

Primul pas în proiectarea unui sistem automat constă în alegerea soluţiei de conducere automată. În cazul de faţă se doreşte o soluţie simplă, care să nu controleze continuu poziţia

axului motorului. Motorul trebuie să pornească la acţionarea butonului de pornire *p* şi să se oprească la acţionarea contactului *a* care indică terminarea unei rotaţii. Algoritmul de conducere prezentat ia în considerare numai evenimente discrete logice şi prin urmare sistemul automat va fi de tip *cu evenimente discrete*.

Automatistul împreună cu tehnologul elaborează schema tehnologică cu aparatura de automatizare din Fig. 4. Modul în care se realizează aceasta nu poate fi algoritmizat. Soluţia aleasă depinde de foarte mulţi factori: banii disponibili pentru instalaţia de automatizare, aparatele de automatizare care pot fi procurate, mediul de lucru al sistemului, fiabilitatea impusă sistemului, calificarea necesară personalului de întreţinere, costul şi timpul acordat proiectului de automatizare, etc. In cazuri simple, ca acesta, se folosesc exemple asemănătoare. *Schema tehnologică* cu aparatura de automatizare prezintă instalaţia care urmează să fie condusă automat, motorul electric împreună cu traductorul de poziţie şi staţia de comandă cu numărul 1. *Traductorul de poziţie* este format din reductorul de turaţie, cama, palpatorul şi contactul *a.* La terminarea unei rotaţii palpatorul acţionează contactul *a* care se închide. Motorul se mai roteşte puţin şi datorită formei camei palpatorul deschide contactul *a*. În această poziţie motorul trebuie să se oprească. Traductorul trebuie ales după caracteristicile tehnice necesare din cataloagele de specialitate sau trebuie proiectat şi construit. *Staţia de comandă* a poziţiei motorului are la intrare contactele *a* şi *p,* şi la ieşire contactul *k2*. Simbolul care o reprezintă indică faptul că staţia de comandă se găseşte în alt loc decât motorul electric. Distanţa dintre staţie şi motor poate ajunge la câteva sute de metri. Din punct de vedere fizic staţia de comandă este un dulap, cutie sau tablou care conţine diferite componente şi aparate de automatizare. In mod curent staţia de comandă poate conţine:

- Alimentarea cu energie electrică a instalaţiei tehnologice.
- Alimentarea instalaţiei de automatizare.
- Automatul, automatul programabil, regulatoarele sau microcalculatorul de proces.
- Sistemul de semnalizare.
- Sistemul de protecţie.
- Sistemul de comutare în diferite regimuri de funcţionare.
- Sistemul de comunicare la distanţă.

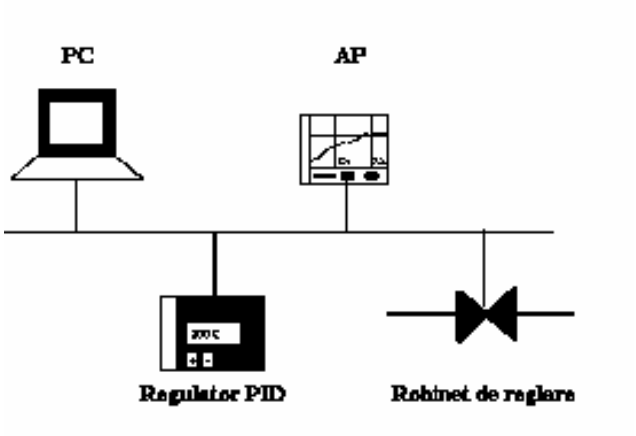


*Fig. 4*

Stațiile de comandă trebuiesc proiectate şi construite din punct de vedere hardware (mecanic, electric şi electronic) şi software. În această lucrare se va pune accentul, în special, pe proiectarea sistemului de conducere.

În continuare se va prezenta în special proiectarea automatului, elementului de execuție, a sistemului de semnalizare şi comutare în diferite regimuri de lucru.

*Caietul de sarcini*. Algoritmul simplu de conducere al motorului este prezentat sub formă de grafcet în Fig. 7 a). Există trei etape şi trei tranziții. Etapa A este activă inițial şi prin convenție, dacă nu este specificat altfel, se consideră că toate acțiunile, exprimate prin expresii logice, iau în această etapă valoarea *0*, adică valoarea logică *fals.* Deci şi motorul electric este oprit în această etapă.

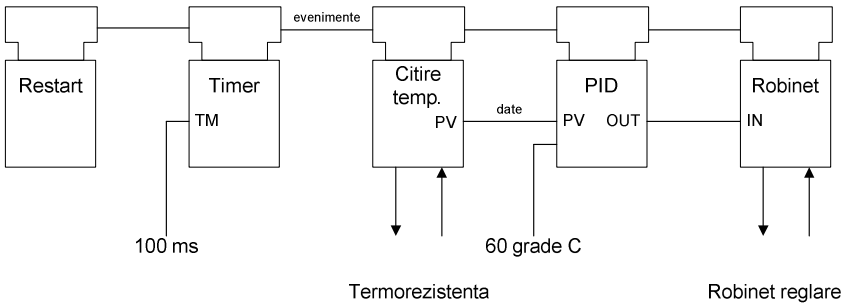


*Fig. 5*

*Analiza structurală.* Să considerăm că există o rețea Petri echivalentă grafcetului. Structura sa este identică, iar din punct de vedere grafic pătratele (etapele) se înlocuiesc cu cercuri (poziții). În etapa inițială A se pun o singură marcă. Deoarece fiecare din cele trei tranziții are un singur arc de intrare şi un singur arc de ieşire Rețeaua Petri este de tip maşină de stare. Graful asociat rețelei Petri este conex deoarece din orice nod[1] se poate ajunge în oricare alt nod, mai mult este tare conex deoarece această performanță se realizează prin intermediul unor arce orientate.

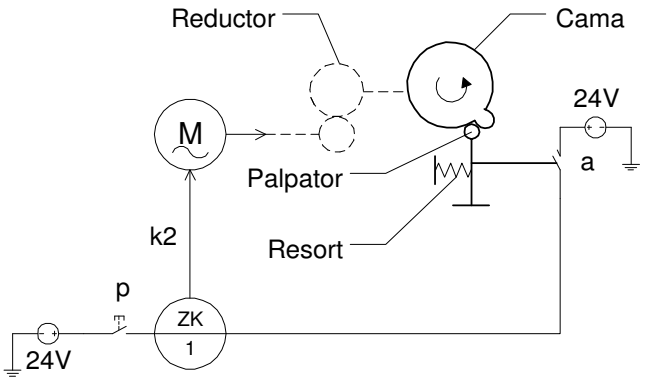


*Fig. 6*

Teorema lui Commoner /15/ spune că dacă graful este tare conex atunci rețeaua este *viabilă*. Dacă marcajul inițial are o singură marcă, cum este cazul aici, atunci rețeaua este

[1] Nodurile grafului sunt pozițiile. Laturile grafului sunt arcele orientate care leagă nodurile atunci când se ignoră tranzițiile.

viabilă şi *sigură*. Analizând structura rețelei se observă imediat că nu există conflicte structurale, deci rețeaua Petri este conformă şi echivalentă cu grafcetul. Toate concluziile pe care te obținem lucrând cu rețeaua Petri sunt valabile şi pentru grafcet.

*Analiza comportamentală*. Tranziția (1), validată inițial, se declanşează când se apasă butonul de pornire şi *p=1*. Devine activă etapa B şi motorul porneşte. La terminarea unei rotații contactul traductorului este acționat de palpator şi *a=1*. Tranziția (2) se declanşează şi devine activă etapa C. Această etapă nu este interpretată de nici o acțiune nouă. Motorul se roteşte în continuare. Rostul acestei etape este să memoreze că a apărut evenimentul specific terminării unei rotații. Cama rotindu-se încet, la un moment dat palpatorul nu mai este împins şi contactul *a* nu mai este acționat. Expresia logică simplă care determină valoarea variabilei *b* va fi atunci egală cu unu (adevărat) şi tranziția (3) se declanşează. Etapa A redevine activă. Există deci un ciclu repetitiv care coincide cu algoritmul dorit pentru automatul sistemului.

Analiza comportamentală trebuie să ia în considerare şi alte situații de funcționare în afară de cea de bază. Ce se întâmplă dacă motorul se roteşte încet şi dorim să se oprească. Vedem că acest lucru nu este posibil. Conform caietului de sarcini motorul se roteşte întotdeauna până când se termină o rotație. Eventualele modificări ale caietului de sarcini se vor discuta o primă varianta a proiectului.

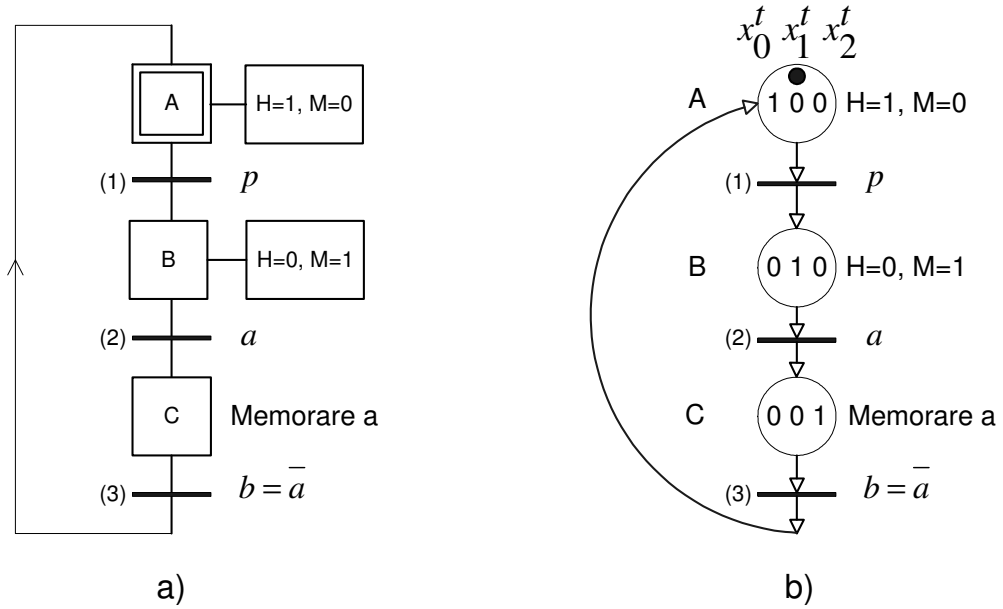


*Fig. 7*

*Metoda de programare Grafcet.* Caietul de sarcini Grafcet din Fig. 7a) poate fi implementat direct la unele automate programabile folosind o interfață grafică. Metoda este foarte utilă în cazul automatelor complexe.

*Metoda listei de instrucțiuni.* Se foloseşte rețeaua Petri conformă care descrie funcționarea automatului. Pentru exemplul considerat aceasta este prezentată în Fig. 7 b). Metoda de proiectare are următoarele etape:

***Etapa 1.*** Codificarea locațiilor cu ajutorul codului distribuit (1 din n). Rezultatul apare în Fig. 7b). Se observă că fiecărei locații îi corespunde un singur bit egal cu unu. Metoda de proiectare se bazează pe observația că acest bit poate fi implementat cu ajutorul unui automat elementar cu prioritate la pornire. Condiția de setare coincide cu condiția de activare a locației iar condiția de resetare coincide cu condiția de dezactivare a locației.

***Etapa 2.*** Calculul condiţiilor de setare şi resetare a biţilor corespunzători fiecărei etape. Setarea coincide cu activarea locaţiei care are loc dacă toate locaţiile precedente sunt active *şi* evenimentele care interpretează tranziţiile de la aceste locaţii au apărut, adică au valoarea logică unu. Resetarea are loc dacă locaţia respectivă este activă *şi* evenimentele care declanşează tranziţiile posterioare tranziţiei au apărut. Exemplul studiat este mai simplu pentru că fiecare locaţie are câte o singură tranziţie precedentă şi o singură tranziţie posterioară

$$n=0 \quad s_0 = x_2 \bar{a} \quad r_0 = x_0 p \tag{1}$$

$$n=1 \quad s_1 = x_0 p \quad r_1 = x_1 a \tag{2}$$

$$n=2 \quad s_2 = x_1 a \quad r_2 = x_2 \bar{a} \tag{3}$$

***Etapa 3.*** Determinarea relaţiilor logice pentru locaţiile active iniţial cu ajutorul următoarei formule care provine din expresia automatului elementar cu prioritate la pornire:

$$x_n^{t+\Delta} = s_n^t + \overline{r_n^t} x_n^t + i \tag{4}$$

Se calculează simplificările posibile.

În exemplul nostru este activă iniţial locaţia A caracterizată de bitul cu n=0. Înlocuind condiţiile *(1)* în (4) se obţine:

$$x_0^{t+\Delta} = x_2^t \bar{a} + \overline{(x_0^t p)} x_0^t + i = x_2^t \bar{a} + (\overline{x_0^t} + \overline{p^t}) x_0^t + i = x_2^t \bar{a} + \overline{p^t} x_0^t + i \tag{5}$$

***Etapa 4.*** Determinarea relaţiilor logice pentru locaţiile inactive la momentul de timp iniţial. Se foloseşte tot o formulă derivată din relaţia logică a automatului elementar cu prioritate la pornire:

$$x_n^{t+\Delta} = (s_n^t + \overline{r_n^t} x_n^t)\bar{i} \tag{6}$$

Pentru automatul din Fig. 7 b) locaţiile B şi C nu sunt active iniţial şi aplicând formula precedentă pentru condiţiile *(3)* şi *(3)* se obţine:

$$x_1^{t+\Delta} = (x_0^t p^t + \bar{a} x_1^t)\bar{i} \tag{7}$$

$$x_2^{t+\Delta} = (x_1^t + x_2^t) a^t \bar{i} \tag{8}$$

***Etapa 5.*** Determinarea relaţiilor logice dintre stările x şi ieşirile y ale automatului la momentul de timp t. Aceste relaţii rezultă din reţeaua Petri care specifică pentru fiecare locaţie, caracterizată prin anumite valori ale variabilelor de stare x, care valorile variabilelor de ieşire y. De exemplu, pentru reţeaua Petri din Fig. 7 b) variabila de ieşire ym care comandă

motorul M are valoarea 1 numai pentru locaţiile B şi C. Diagrama Karnaugh corespunzătoare şi relaţia logică stabilită[2] sunt prezentate în Fig. 8

$y_m^t = x_1^t + x_2^t$

| $x_2^t$ \ $x_0^t x_1^t$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | - | 0 |
| 1 | 1 | - | - | - |

*Fig. 8*

$y_h^t = x_0^t$

| $x_2^t$ \ $x_0^t x_1^t$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | - | 1 |
| 1 | 0 | - | - | - |

*Fig. 9*

Din aceiaşi reţea Petri rezultă că lampa H semnalizează numai în locaţia A şi deci numai atunci variabila de ieşire corespunzătoare $y_h$ are valoarea 1. Relaţia logică corespunzătoare determinată cu ajutorul diagramei Karnaugh este prezentată în Fig. 9.

***Etapa 6.*** Elaborarea tabelului de configurare. Se stabileşte o corespondenţă între variabilele relaţiilor logice şi denumirile elementelor componente ale automatului programabil: intrări, ieşiri, memorii (relee), timere (relee de timp), etc. Pentru exemplul considerat acestea apar în Tab 1 şi Tab 2.

[2] Există tentaţia ca în cazul unor relaţii logice simple, cum este cazul exemplului prezentat, să se deducă direct relaţia logică fără ajutorul diagramei Karnaugh. Este o metodă greşită pentru că permite ignorarea anumitor situaţii care sunt puse însă în evidenţă de către diagrama Karnaugh. În exemplul prezentat acesta este cazul cu situaţia în care toate variabilele de stare sunt egale cu zero, situaţie care nu apare în reţeaua Petri. Diagrama Karnaugh ne-a silit să precizăm valorile de ieşire în acest caz. Am hotărât, de exemplu, că în acest caz lampa de semnalizare H nu este aprinsă.

*Tab 1 Configurare APL pentru intrări şi ieşiri*

| ***i*** | ***p*** | ***a*** | $\mathbf{y_m}$ | $y_h$ |
|---|---|---|---|---|
| I0.0 | I0.1 | I0.2 | Q0.1 | Q0.0 |
| M0.0 | M0.1 | M0.2 | M0.14 | M0.15 |

*Tab 2 Configurare APL pentru stări şi memorii de lucru*

| $x_0^t$ | $x_1^t$ | $x_2^t$ | $x_0^{t+\Delta}$ | $x_1^{t+\Delta}$ | $x_2^{t+\Delta}$ | ***tampon1*** | ***tampon2*** |
|---|---|---|---|---|---|---|---|
| M0.3 | M0.4 | M0.5 | M0.6 | M0.7 | M0.8 | M0.9 | M0.10 |
| Q0.3 | Q0.4 | Q0.5 | | | | | |

***Etapa 7.*** Elaborarea programului de funcționare a automatului programabil logic sub forma unei liste de instrucțiuni.

Pentru exemplul studiat lista de instrucțiuni corespunzătoare automatului programabil logic Klockner Moeller PS3 este prezentată în Tab 3. Se observă că programul are următoarele secțiuni:

- ✓ Achiziția intrărilor (Input scan)
- ✓ Calculul relațiilor logice (Logic scan)
- ✓ Actualizarea variabilelor de stare
- ✓ Calculul variabilelor de ieşire
- ✓ Furnizarea ieşirilor (Output scan)
- ✓ Operațiuni de semnalizare sau testare

*Tab 3 Programul sub formă de listă de instrucțiuni*

| ***Adresa*** | ***Instrucțiunea*** | ***Comentariu*** | ***Adresa*** | ***Instrucțiunea*** | ***Comentariu*** |
|---|---|---|---|---|---|
| 000 | LI0.0 | Achiziția intrărilor | 029 | LM0.2 | |
| 001 | =M0.0 | | 030 | AM0.5 | |
| 002 | LI0.1 | | 031 | =M0.10 | |
| 003 | =M0.1 | | 032 | LM0.9 | |
| 004 | LI0.2 | | 033 | OM0.10 | |
| 005 | M0.2 | | 034 | ANM0.0 | |
| 006 | LM0.5 | Calcul $x_0^{t+\Delta}$ | 035 | =M0.8 | |
| 007 | ANM0.2 | | 036 | LM0.6 | Actualizare stări |
| 008 | =M0.9 | | 037 | =M0.3 | $x_n^{t+\Delta}$ --> $x_n^t$ |
| 009 | LNM0.1 | | 038 | LM0.7 | |
| 010 | AM0.3 | | 039 | =M0.4 | |
| 011 | =M0.10 | | 040 | LM0.8 | |
| 012 | LM0.9 | | 041 | =M0.5 | |
| 013 | OM0.10 | | 042 | LM0.4 | Calcul ieşiri $y_n^t$ |
| 014 | OM0.0 | | 043 | OM0.5 | |
| 015 | =M0.6 | | 044 | =M0.14 | |
| 016 | LM0.3 | Calcul $x_1^{t+\Delta}$ | 045 | LM0.3 | |
| 017 | AM0.11 | | 046 | =M0.15 | |
| 018 | =M0.9 | | 047 | LM0.14 | Furnizare ieşiri $y_n^t$ |