Cursul 1

1.1. Procese industriale. Sisteme cu stări finite

Prin proces industrial se înţelege un ansamblu de transformări mecanice, electrice sau de altă natură care au loc în instalaţii industriale. Procesele sunt descrise prin relaţiile cauzale care există între mărimile de intrare şi de ieşire ale procesului (fig.1.1).

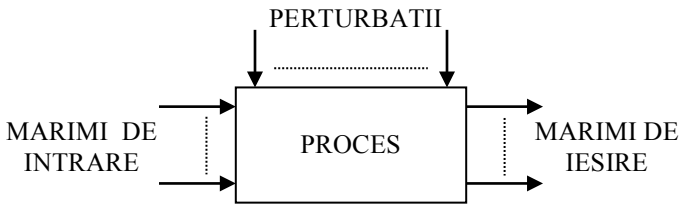

Figura 1.1

Mărimile de ieşire, numite şi mărimi reglate, sunt acelea care caracterizează modul de desfăşurare al procesului, variaţia (valoarea) lor constituind obiectul conducerii procesului. Mărimile de intrare sunt acelea care determină în mod cauzal variaţia mărimilor de ieşire. Ele se împart în două categorii:

- mărimi comandate, numite şi comenzi, care se modifică manual sau automat astfel încât să se asigure desfăşurarea procesului conform obiectivelor impuse;
- mărimi necomandate, numite şi perturbaţii, de care depinde desfăşurarea procesului, dar ale căror valori nu pot fi controlate în scopul conducerii procesului, ci se modifică independent.

Clasificarea setului de mărimi de intrare se face de către proiectant.

Un exemplu este cel al unui robot destinat asamblării automate (fig.1.2).

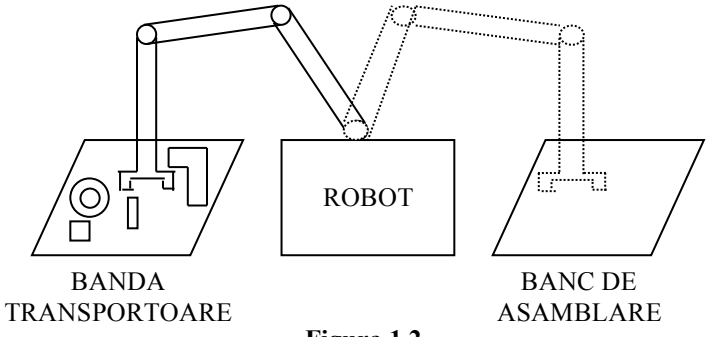

Figura 1.2

Obiectivul acestui proces este de a prelua o anumită piesă de pe banda transportoare şi a o monta într-o poziţie bine determinată pe bancul de asamblare. Este un proces complex cu mai multe mărimi de ieşire: poziţia braţului robotului, poziţia şi prezenţa piesei în sistemul de prindere, poziţia finală a piesei. Mărimile de intrare pentru proces sunt: tipurile de piese existente pe banda transportoare şi poziţia lor, starea unor motoare de acţionare, a unor microîntrerupătoare şi a unor limitatoare de cursă, a unor traductoare de poziţie şi altele. Procesul este condus prin comenzi de pornire şi oprire a motoarelor de acţionare, comenzi de prindere şi desprindere a pieselor, etc.
Aparatura destinată conducerii procesului constituie aşa numitul "sistem de conducere". Sistemul de conducere urmăreşte desfăşurarea procesului şi generează comenzile către proces în scopul realizării obiectivelor impuse. Clasificarea mărimilor de intrare ale procesului în comenzi şi perturbaţii este o operaţie necesară în vederea elaborării temei de proiectare a sistemului de conducere. Unele mărimi de ieşire ale procesului, unele perturbaţii precum şi un program de desfăşurare a procesului constituie mărimi de intrare ale sistemului de conducere (fig.1.3).

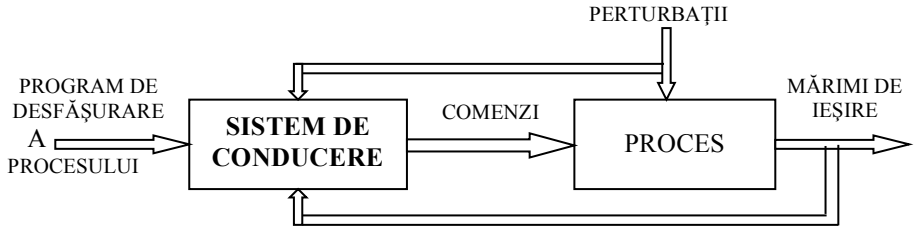

Figura 1.3

În funcţie de modul de funcţionare, sistemele de conducere se clasifică în sisteme continue şi sisteme discrete (sisteme cu stări finite). În principiu orice sistem care operează la momente discrete de timp şi a cărui intrări, ieşiri şi structură internă işi pot atribui numai un număr finit de configuţii distincte, poartă denumirea de sistem discret sau sistem cu stări finite.
Circuitele de comutaţie constituie componentele de bază în proiectarea sistemelor de conducere discrete moderne şi studiul lor constituie obiectivul prezentului curs. Mai trebuie subliniat că sistemele de conducere discrete pot conduce atât procese continue cât şi procese discrete.

1.2 Discretizarea, eşantionarea, cuantizarea

Atunci când sistemul de conducere este un sistem cu stări finite şi informaţia pe care trebuie să o utilizeze este continută în mărimi continue, aceste mărimi sunt supuse unui proces de discretizare. Discretizarea se realizează cu CAN, acestea fiind dispozitive care transformă mărimea continuă aplicată la intrare într-un număr furnizat la ieşire, număr ce este funcţie de valoarea mărimii de intrare.

Conversia analog-numerica, prin care unei mărimi care variază continuu în timp şi care poate lua o infinitate de valori i se asociază o secvenţă de numere, constă dintr-un proces de eşantionare şi unul de cuantizare (fig.1.4 a, b, c).

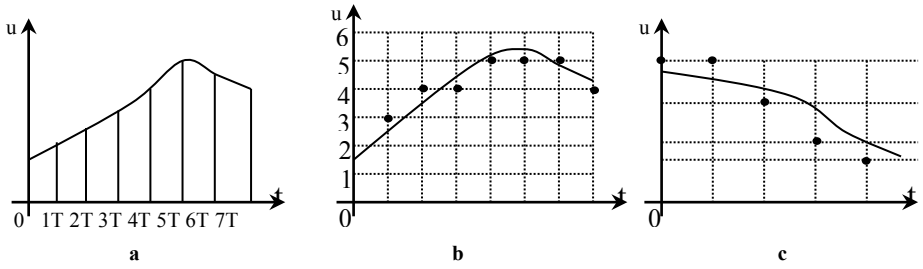

Figura 1.4

Eşantionarea constă în preluarea valorilor mărimii continue numai în anumite momente, de obicei la intervale constante de timp.

Intervalul de eşantionare se alege în funcţie de banda de frecvenţă a semnalului continuu astfel încât acesta să se poată reconstitui din eşantioanele extrase. Conform teoremei eşantionării, dependenţa dintre perioada de eşantionare T şi frecvenţa maximă din spectrul semnalului continuu $f_{\max}$, este:

$$T < \frac{1}{2 f_{\max}} \qquad (1.1)$$

Cuantizarea constă în asocierea unui număr pentru toate valorile semnalului continuu cuprinse într-un interval denumit cuantă. Se defineşte o cuantizare liniară şi una neliniară.

1.3 Sisteme de numeraţie

Numerele pot fi reprezentate în diferite moduri în funcţie de sistemul de numeraţie utilizat. Dintre sistemele de numeraţie folosite o mai largă

răspândire au sistemul zecimal, binar, octal şi hexazecimal. Toate aceste sisteme de numeraţie sunt sisteme poziţionale caracterizate prin faptul că ponderea fiecărei cifre din reprezentarea unui număr depinde de poziţia acesteia în reprezentare.

În general, într-un sistem de numeraţie cu o bază pozitivă întreagă "b", un numar oarecare N se exprimă în felul următor:

$$N_b \rightarrow a_n a_{n-1} \cdots a_1 a_0 \qquad (1.2)$$

iar valoarea lui poate fi determinată cu formula:

$$N = a_n.b^n + a_{n-1}.b^{n-1} + \cdots + a_1 b + a_0 \qquad (1.3)$$

(a) Sistemul de numeraţie zecimal (b=10) este sistemul cel mai frecvent folosit în elementele care realizează interacţiunea dintre om şi sistem (introducerea datelor, afişarea rezultatelor). La reprezentarea numerelor se utilizează zece cifre.

(b) Sistemul de numeraţie binar (b=2) este sistemul utilizat pentru reprezentarea internă a numerelor în sistemele de calcul. La reprezentarea numerelor în sistem binar se folosesc două cifre: 0 şi 1.

Exemplu: $N_2 \rightarrow 1101 \quad N = 1.2^3 + 1.2^2 + 1 = 13$

(c) Sistemul octal de numeraţie (b=8) este utilizat la introducerea şi extragerea datelor numerice în sisteme mici de calcul pentru simplificarea dispozitivelor de conversie. La reprezentare se folosesc 8 cifre.

Ex: $N_8 \rightarrow 361 \qquad N = 3.8^2 + 6.8 + 1 = 241$

Conversia din sistemul octal în sistemul binar se face înlocuind fiecare cifră octală cu reprezentarea ei binară.

Ex: $361_8 \leftrightarrow 011110001_2$

Conversia inversă, din sistem binar în sistemul octal, se face înlocuind fiecare grup de 3 cifre alăturate, începând cu cifra cea mai puţin semnificativă, cu cifra octală corespunzătoare.

Ex: $11\underbrace{001}\underbrace{011}_2 \rightarrow 313_8$

(d) Sistemul hexazecimal de numeraţie este asemănător cu sistemul octal şi are aceeasi destinaţie, cu precizarea că este preferat deoarece conduce la o reprezentare foarte compactă a numerelor ceea ce constituie un avantaj real atunci când volumul datelor de introdus este mare.

Conversia din cod hexazecimal în cod binar şi invers se face după aceleasi principii ca şi pentru codul octal.

Sistemul utilizează cele zece cifre din sistemul zecimal urmate de primele şase litere mari din alfabet.

1.4 Coduri binare

În sistemele numerice pentru reprezentarea numerelor şi a simbolurilor se utilizează succesiuni de cifre binare de 0 şi 1. Corespondenţa dintre elementele mulţimii simbolurilor şi elementele mulţimii succesiunilor de 0 şi 1 defineşte un cod.

1.4.1 Reprezentarea datelor numerice

1.4.1.1 Codul binar. Reprezentarea numerelor binare cu semn

Există trei moduri de reprezentare a numerelor binare cu semn:

- reprezentarea prin mărime şi semn;
- reprezentarea prin complementul faţă de 1;
- reprezentarea prin complementul faţă de 2.

În reprezentarea prin mărime şi semn prima cifră reprezintă semnul şi este, prin convenţie, 0 pentru numerele pozitive şi 1 pentru numerele negative. Celelalte cifre reprezintă mărimea numărului în cod binar natural. De exemplu, într-un sistem de calcul în care se alocă 8 cifre pentru reprezentarea numerelor, numerele +53 si –53 se exprimă astfel:

$$+53 \rightarrow 00110101$$

$$-53 \rightarrow 10110101$$

În reprezentarea prin complementul faţă de 1 numerele pozitive se reprezintă la fel ca şi în reprezentarea prin mărime şi semn, iar cele negative se obţin prin schimbarea fiecărei cifre din reprezentarea prin mărime şi semn cu complementul ei, mai puţin cifra de semn. Pentru numerele considerate avem:

$$+53 \rightarrow 00110101$$

$$-53 \rightarrow 11001010$$

În reprezentarea prin complementul faţă de 2, numerele pozitive se reprezintă la fel ca şi în reprezentarea prin mărime şi semn, iar cele negative se obţin în două etape:

1- se formează complementul faţă de 1;

2- se adună o unitate la poziţia cea mai puţin semnificativă.

Pentru exemplul considerat avem:

$$+53 \rightarrow 00110101$$

$$-53 \rightarrow 11001011$$

1.4.1.2 Codul zecimal-binar 8421

Codul zecimal-binar se foloseşte atunci când ponderea operaţiilor de prelucrare a datelor este mică în comparaţie cu ponderea operaţiilor de introducere/extragere (care se face în sistem zecimal) şi conversia în cod binar în vederea prelucrării este neeconomică. De asemenea codurile zecimal-binare se folosesc în cazurile în care se cere efectuarea de calcule exacte asupra numerelor zecimale fracţionare.

Într-o codificare zecimal-binară fiecărei cifre zecimale îi corespunde o combinaţie de $n \geq 4$ cifre binare.

Dintre codurile zecimal-binare cel mai cunoscut este codul 8421 în care fiecare cifră zecimală este codificată separat cu 4 cifre binare, grupurile binare scriindu-se în ordinea în care au fost scrise cifrele zecimale.

Ex: $53 \rightarrow 01010011_{BCD}$

Observaţie: Întrucât din cele 16 combinaţii posibile ce se pot forma cu 4 cifre binare în codul 8421 (cod BCD) se folosesc numai 10, operaţiile aritmetice cu numere codificate în cod BCD prezintă unele particularităţi.

Ex.1: Adunând 3 cu 4 în cod BCD rezultă:

$$\begin{array}{r} 0011+ \\ 0100 \\ \hline 0111 \end{array} \quad \Rightarrow 7 \text{ rezultat}$$

Ex.2: Adunând 4 cu 7 în cod BCD se obţine:

$$\begin{array}{r} 0100+ \\ 0111 \\ \hline 1011+ \\ 0110 \\ \hline 10001 \end{array}$$

↑ rezultat corect

-> acest număr nu există în cod BCD. Pentru a obţine codul rezultatului corect care este 00010001BCD trebuie facută o corecţie adunând codul cifrei 6.(corespunzător combinaţiilor neutilizate)

Ex.3: Adunând numerele 8 cu 9 în cod BCD ar trebui obţinut codul 00010111.

$$\begin{array}{r} 1000+ \\ 1001 \\ \hline 10001+ \\ 0110 \\ \hline 10111 \end{array}$$

corecţia

← corect

1.4.1.3 Coduri continue

Codurile continue sunt frecvent intâlnite, alături de codul binar natural, în dispozitivele numerice. Pentru a putea fi definite este necesar ca

în prealabil să fie definită noţiunea de adiacenţă referitoare la cifre sau combinaţii de cifre.

Într-un sistem de cifre două cifre se zic adiacente dacă diferă printr-o unitate. Sunt adiacente de asemenea cifrele extreme. Ca urmare, două combinaţii de cifre sunt adiacente dacă ele nu diferă decât prin cele două cifre ale unui singur rang, cifre care la rândul lor sunt adiacente. De exemplu numărul zecimal 460 are următoarele şase numere adiacente: 360, 560, 450, 470, 469, 461.

În cazul particular al sistemului binar (nu există decât două cifre 0 şi 1 evident adiacente) două combinaţii adiacente sunt două combinaţii care au două cifre diferite numai într-un singur rang.

Un cod cuprinzând toate combinaţiile posibile şi în care două combinaţii consecutive sunt adiacente se numeşte cod continuu. Dacă, în plus, ultima combinaţie este adiacentă cu prima, el se numeşte continuu ciclic. Dintre codurile binare continue ciclice cel mai frecvent folosite sunt codurile binare reflectate. Ele se numesc reflectate deoarece, dacă se inşiră cuvintele codului în ordinea naturală a numerelor pe care le codifică, pentru primele "m-1" cifre binare din cele "m" cifre binare ale cuvântului de cod, numărate de la dreapta la stânga, se poate găsi o axă de simetrie faţă de care se reflectă cele "m-1" cifre binare ale cuvântului de cod. Apoi, în cadrul fiecărei jumătăţi, pentru primele "m-2" cifre se mai găseşte câte o axa de simetrie s.a.m.d.

În fig.1.5 este dat codul binar reflectat cu patru cifre binare alături de codul binar natural. Codul binar natural reflectat, care este un cod ciclic, poartă denumirea de cod Gray.

COD BINAR NATURAL				COD BINAR REFLECTAT			
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1

1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

Figura 1.5

1.4.2 Reprezentarea datelor alfa-numerice

Codificarea datelor alfa-numerice este necesară pentru a putea imprima şi memora mesaje sau comenzi. În mod obişnuit se codifică 90 de caractere distincte care cuprind:

- 52 de simboluri pentru literele mari şi mici ale alfabetului;
- 10 simboluri pentru cifrele zecimale;
- 28 de simboluri pentru caractere speciale.

Codificarea a 90 de caractere necesită minimum 7 biţi. Datorită faptului că sistemele de calcul au magistrala de date organizată pe cel puţin 8 biţi, codificarea caracterelor alfanumerice se face cu 8 biţi, bitul al 8-lea putand fi folosit pentru verificarea paritatii.

Exemplu de cod alfa-numeric frecvent utilizat este codul ASCII.

1.5 Circuite de comutaţie

Prelucrarea şi păstrarea datelor în sistemele de conducere cu stări finite se realizează cu ajutorul unor circuite cu numai două stări stabile, distincte, numite circuite de comutaţie. De exemplu, bobina unui releu poate fi parcursă de un curent sau nu, contactele unui releu pot fi închise sau deschise, un tranzistor poate fi blocat sau saturat, etc. În vederea utilizării lor la sinteza sistemelor de conducere cu stări finite, celor două stări distincte ale circuitelor de comutaţie li se vor asocia cele doua cifre ale codului binar: 0 şi 1.

Considerând drept criteriu de clasificare a circuitelor de comutaţie modul lor de funcţionare, avem:

a) circuite cu comutaţie dinamică;
b) circuite cu comutaţie statică.

Circuitele cu comutaţie dinamică au fost primele circuite utilizate în sistemele de conducere şi au fost materializate de către releele electromagnetice.

Circuitele de comutaţie statică cele mai frecvent utilizate în practică sunt circuitele de comutaţie cu porţi logice. În general o poartă poate avea mai multe intrări U_i unde $i = 1,2,\cdots,n$. Ieşirea porţii Y este funcţie de intrările acesteia, $Y = f(u_1, u_2, \cdots, u_n)$ unde f este funcţie de comutaţie sau funcţie logică.

CAP. II FUNCŢII LOGICE

2.1 Elemente de algebră booleană

2.1.1 Axiomele şi teoremele algebrei booleene

Analiza şi sinteza circuitelor de comutaţie se face cu ajutorul algebrei booleene.

Se consideră o mulţime *B*, cu cel puţin două elemente distincte, în care se definesc două operaţii binare, operaţia SAU (pentru care se foloseşte operatorul "+") şi operatia SI (pentru care se foloseşte operatorul "**.**"), precum şi o relaţie de echivalenţă între elementele mulţimii *B*, pentru care se foloseşte simbolul "=".

În mulţimea *B* există două constante caracteristice, constanta 0 şi constanta 1.

Mulţimea *B* considerată mai sus este o algebră booleană dacă sunt satisfăcute următoarele axiome:

A.1. Operaţiile SAU, respectiv SI, sunt asociative. Pentru orice $a,b,c \in B$

$$\begin{aligned} &(a+b)+c = a+(b+c) \\ &(a.b).c = a.(b.c) \end{aligned} \qquad (2.1)$$

A.2. Operaţiile SAU, respectiv SI, sunt comutative. Pentru $\forall a,b \in B$

$$\begin{aligned} &a+b = b+a \\ &a.b = b.a \end{aligned} \qquad (2.2)$$

A.3. Există în mulţimea B două elemente 0 şi 1 cu efect nul faţă de cele două operaţii. Astfel pentru $\forall a \in B$

$$\begin{aligned} &a+0 = 0+a = a \\ &a.1 = 1.a = a \end{aligned} \qquad (2.3)$$

A.4. Operaţiile SAU, respectiv SI, sunt distributive una faţă de alta. Pentru $\forall a,b,c \in B$

$$\begin{aligned} &a+(b.c) = (a+b)(a+c) \\ &a.(b+c) = a.b + a.c \end{aligned} \qquad (2.4)$$

A.5. Fiecare element a din mulţimea B are un complement în B, notat $\bar{a}$, astfel încât:

$$\begin{aligned} a + \bar{a} &= 1 \\ a.\bar{a} &= 0 \end{aligned} \qquad (2.5)$$

Pe baza axiomelor de mai sus se pot demonstra o serie de teoreme dintre care cele mai importante vor fi prezentate în continuare.

Teorema 1. Idempotenţa elementelor mulţimii B pentru operaţiile SAU respectiv SI.

$$\begin{aligned} a + a &= a \\ a.a &= a \end{aligned} \qquad (2.6)$$

Teorema 2. Teorema elementelor absorbante pentru operaţiile SAU respectiv SI.

$$\begin{aligned} a + 1 &= 1 \\ a.0 &= 0 \end{aligned} \qquad (2.7)$$

Teorema 3. Legile absorbţiei.

$$a + a.b = a \qquad a.(\bar{a} + b) = a.b$$

$$a.(a + b) = a \qquad a.b + a.\bar{b} = a$$

$$a + \bar{a}.b = a + b \qquad (a + b).(a + \bar{b}) = a$$

Teorema 4. Unicitatea complementului: orice element $a \in B$ are un singur complement in B.

Teorema 5. Legea dublei complementări.

$$\bar{\bar{a}} = a$$

Teorema 6. Legile lui De Morgan.

$$\begin{aligned} \overline{a.b.c.\cdots.z} &= \bar{a} + \bar{b} + \bar{c} + \cdots + \bar{z} \\ \overline{a + b + c + \cdots + z} &= \bar{a}.\bar{b}.\bar{c}.\cdots.\bar{z} \end{aligned} \qquad (2.8)$$

2.1.2 Algebra circuitelor de comutaţie

În studiul circuitelor de comutare se utilizează o algebră booleană în care mulţimea B considerată are numai două elemente, 0 şi 1, corespunzătoare celor două stări stabile ale circuitelor de comutare. Operaţia SAU este definită în tabelul 2.1, operaţia SI în tabelul 2.2, iar complementarea în tabelul 2.3.

SAU	0	1
0	0	1
1	1	1

Tabelul 2.1

SI	0	1
0	0	0
1	0	1

Tabelul 2.2

NU	
0	1
1	0

Tabelul 2.3

Această algebră booleană numită şi algebra comutaţiei este identică cu algebra booleană folosită în logică în care însă cele două elemente ale mulţimii *B* sunt constantele logice “adevărat” şi “fals”. Din acest motiv algebra comutaţiei este denumită frecvent algebra logicii, operaţia SAU este numită şi sumă logică, operaţia SI este numită şi produs logic, iar complementarea este denumită negaţie. De asemenea circuitele de comutaţie se mai numesc şi circuite logice, iar funcţiile de transfer ale acestora se numesc funcţii logice.