LUCRAREA 1

INTRODUCERE ÎN MEDIUL DE PROGRAMARE SIMULINK

1.1. Obiectivele lucrării

În această primă lucrare de laborator, căreia îi sunt rezervate patru ore, se urmăreşte atingerea de către studenţi a următoarelor obiective:

- însuşirea modului de lucru cu mediul de programare SIMULINK;
- familiarizarea cu obiectele din biblioteca SIMULINK;
- construirea diagramelor destinate simulării unor elemente simple de simulare, în cadrul mediului de programare SIMULINK;

1.2. Prezentarea conţinutului lucrării

MATLAB este un pachet de programe de înaltă performanţă, dedicat calculului numeric şi reprezentărilor grafice în domeniul ştiinţei şi tehnici Una dintre aplicaţiile specifice versiunii 6.5 al mediului MATLAB este SIMULINK. Acest pachet de programe este utilizat pentru simularea matematică a sistemelor dinamice cu ajutorul unor elemente dinamice fundamentale.

1.2.1. Lansarea în execuţie a mediului SIMULINK

Mediul SIMULINK poate fi activat prin intermediul mediului MATLAB, în două moduri:

1. Se face click pe icona Simulink  , din bara de instrumente a mediului Matlab;
2. Din mediul Matlab, în linia de comanda se editează comanda ***simulink*** şi se execută, figura 1.1.

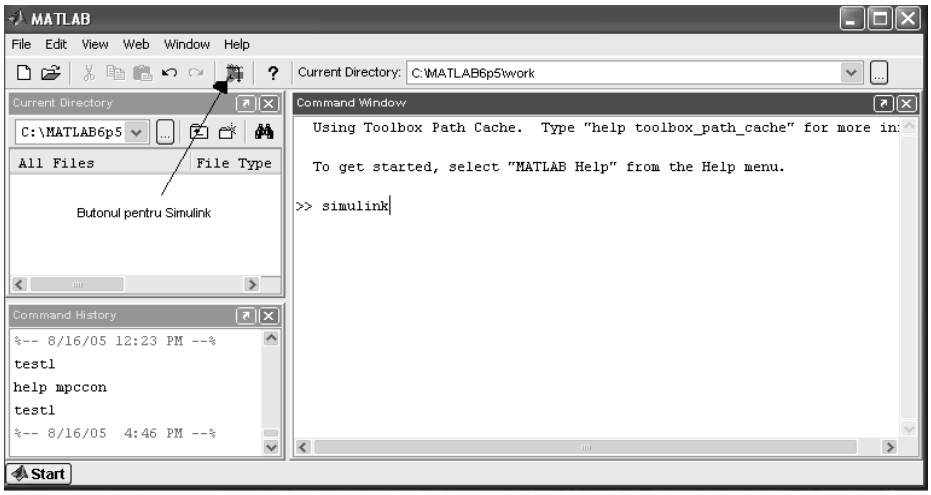


Fig. 1.1. Lansarea în execuţie a mediului SIMULINK, utilizând comanda *simulink*.

În urma acţiunii uneia din comenzile specificate anterior, este lansat în execuţie mediul SIMULINK. Pe ecran se va deschide o fereastră ce conţine componentele aflate în biblioteca SIMULINK, figura 1.2.

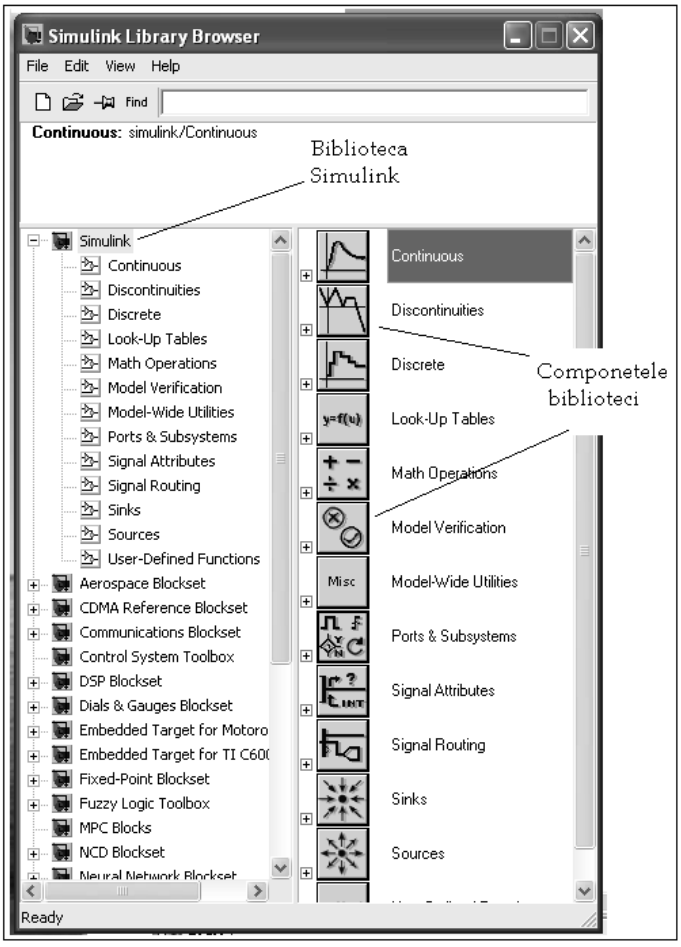


Fig. 1.2. Biblioteca mediului Simulink.

Pentru construirea unei diagrame se vor selecta comenzile ***New, Model*** din mediul de comenzi ***File*** al mediului Simulink, figura 1.3. În urma execuţiei acestei acţiuni, pe ecran se va deschide o fereastră destinată construirii şi simulării diagramelor, ilustrată în figura 1.4.

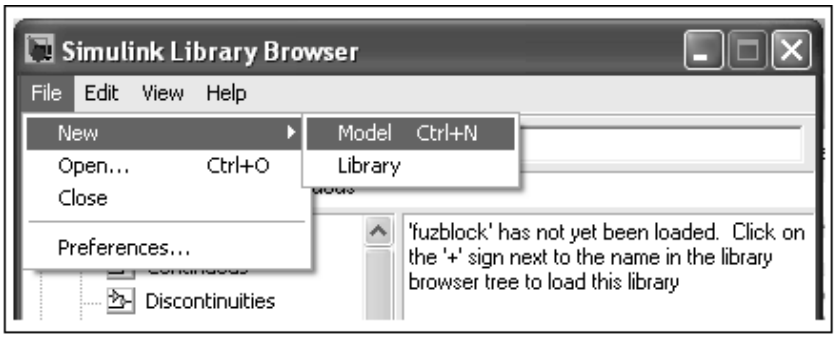


Fig. 1.3. Lansarea în execuţie a ferestrei de construire a unei diagrame.

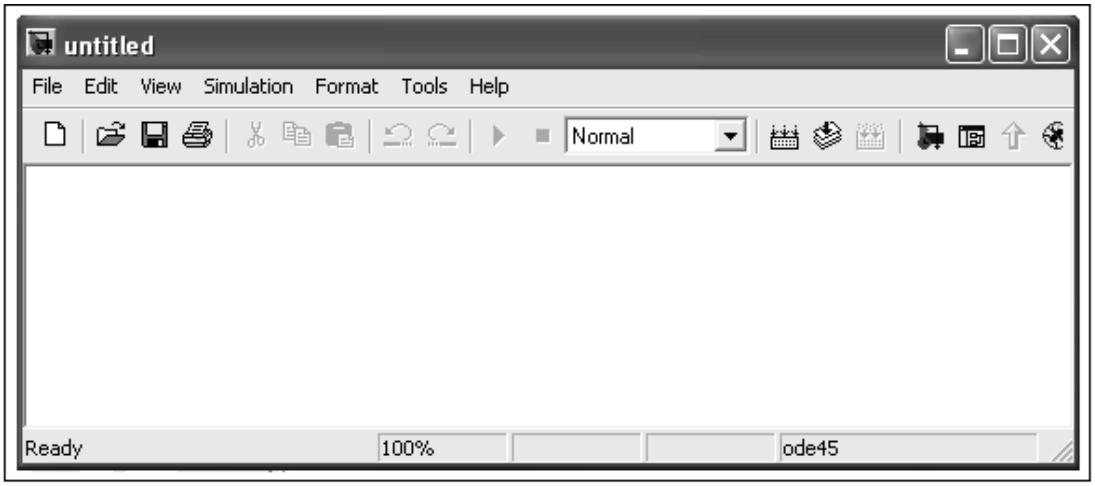


Fig. 1.4. Imaginea ferestre diagramei destinată construiri şi simulării diagramelor.

1.2.2. Prezentarea componentelor din biblioteca Simulink

Biblioteca mediului Simulink (Library Simulink) conţine un set componente destinate realizării unor operaţii elementare, având semnificaţie matematică sau de natura generării şi prelucrării semnalelor, figura 1.2. Semnificaţia celor mai utilizate componente din bibliotecă sunt prezentate în tabelul 1.1.

Tabelul 1.1.

Semnificaţia celor mai utilizate componente din biblioteca SIMULINK

Nr. crt.	Denumire	Semnificaţie
1	Source	Generarea semnalelor sursă
2	Sinks	Reprezentarea grafică a dinamicii sistemelor
3	Discrete	Simularea sistemelor discrete in timp
4	Continuous	Simularea sistemelor liniare şi sistemelor neliniare
5	Math Operation	Realizarea operaţiilor matematice
6	Signal Routing	Realizarea conexiunilor

La rândul ei fiecare componentă conţine un set de instrumente. Dacă se va executa dublu-click pe oricare dintre componentele din bibliotecă, în partea dreaptă a ferestrei din fig 1.2 vor apare instrumentele componentei respective, figura 1.5. În cele ce urmează se vor detalia câteva din cele mai utilizate componente ale bibliotecii Simulink.

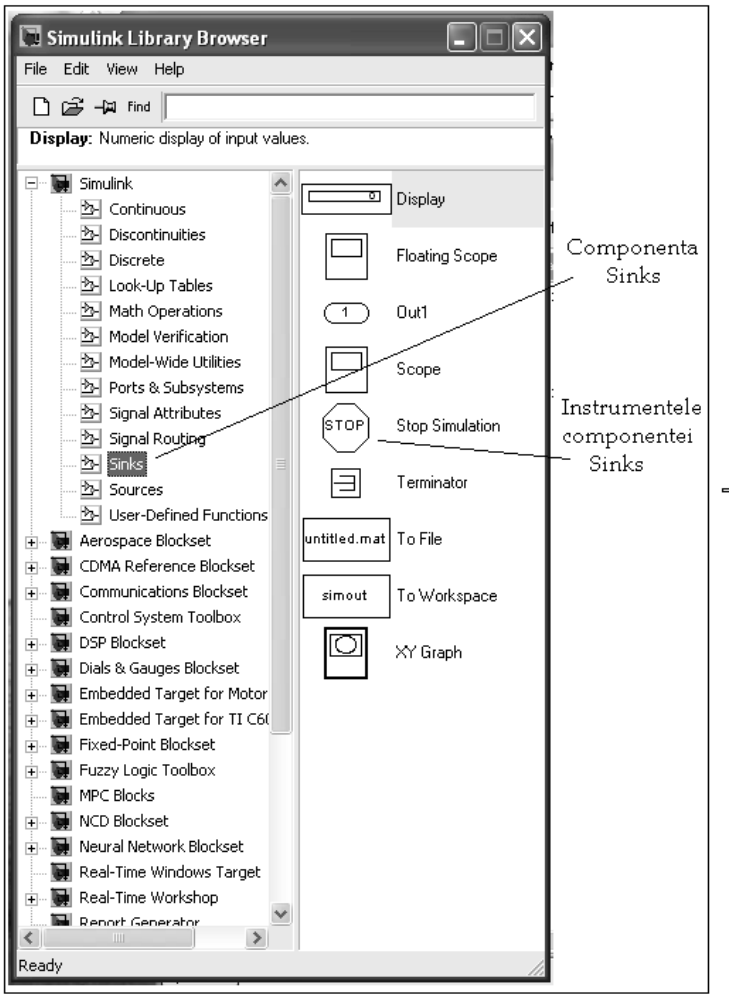


Fig.1.5. Instrumentele componentei *Sinks*.

A1. Componenta semnalelor sursă (Signal Source Library)

Componenta semnalelor sursă conţine instrumente generatoare de semnale de intrare aplicate sistemului studiat. Aceste instrumente sunt obţinute prin activarea componentei **Sources**. Principalele instrumente asociate componentei semnalelor sursă sunt prezentate în tabelul 1.2. Semnalele sursă cele mai utilizate în cadrul aplicaţiilor sunt: semnalul de ceas, constanta, semnalul sinusoidal şi semnalul treaptă.

Tabelul 1.2.

Componenta **Source**

Nr. crt.	Denumire instrument	Semnificaţie
1	Constant	Semnal de intrare constant
2	Clock	Semnal de ceas
3	Sine Wave	Semnal sinusoidal
4	Step	Semnal treaptă
5	Ramp	Semnal rampă

A2. Componenta semnalelor de ieşire ***(****Signal* ***Sinks*** *Library****)***

Instrumentele componentei *Sinks* sunt obţinute prin activarea pictogramei **Sinks**. Semnificaţia principalelor instrumente ale componentei *Sinks* sunt prezentate în tabelul 1.3.

Tabelul 1.3

Componenta semnalelor de ieşire

Nr.ctr.	Denumire	Semnificaţie
1	Scope	Vizualizare mărimi de ieşire pe osciloscop
2	To Workspace	Mediul de lucru
3	Stop Simulation	Sfârşitul simulării

A3. Componenta sistemelor continue ***(Continuous****-Library)*

Instrumentele componentei sistemelor liniare şi neliniare sunt obţinute prin activarea pictogramei **Continuous**. Principalele instrumente asociate componentei sistemelor liniare sunt prezentate în tabelul 1.4. Componenta sistemelor liniare şi neliniare conţine instrumente dedicate funcţiilor matematice algebrice: sumator, amplificator etc.

Tabelul 1.4.

Componenta sistemelor liniare

Nr.crt.	Denumire	Semnificaţie
1	Integrator	Integrator
2	Derivative	Derivator
3	Transfer Fcn	Funcţia de transfer

A4. Componenta funcţiilor matematice

Prin activarea pictogramei ***Math Operation*** sunt obţinute instrumentele componentei funcţiilor matematice. Semnificaţia componentelor din blocul funcţiilor matematice sunt prezentate în tabelul 1.5. Meniul funcţiilor matematice conţine obiectele pentru funcţii matematice algebrice, funcţia histerezis, funcţia de întârziere şi saturare.

Tabelul 1.5.

Componenta sistemelor neliniare

Nr.crt.	Denumire	Semnificaţie
1	Sin	Funcţia trigonometrică sinus
2	Min	Funcţia minim
3	Gain	Amplificator
4	Sum	Sumator
5	Product	Produs

1.3. Partea experimentală

Pentru a realiza simularea dinamică a unui sistem, utilizând mediul SIMULINK, sunt necesare parcurgerea următoarelor etape:

1. Determinarea modelului matematic.
2. Identificarea blocurilor corespunzătoare elementelor dinamice, care modelează sistemul.
3. Realizarea diagramei sistemului, formată din blocuri standard (aflate în biblioteca Simulink) sau a blocurilor proprii (create de utilizator).
4. Configurarea fiecărui bloc, în funcţie de modelul matematic şi parametrii asociaţi sistemului.
5. Lansarea în execuţie, etapă realizată prin comanda *Start* din meniul *Simulation.*
6. Selectarea opţiunilor necesare vizualizării rezultatelor simulării.

Exemplul 1 Fie modelul de forma:

$$y = 3 \cdot u \tag{1.1}$$

asociat unui element proporţional. Să se construiască diagrama care va implementa relaţia (1.1) în mediul Simulink, pentru u=5;

Rezolvare

A1. Identificarea blocurilor. Relaţia 1.1. poate fi implementată prin intermediul următoarelor blocuri:

- pentru generarea semnalului de intrare *u* se va utiliza blocul *Constant* din componenta *Source Library*.
- Blocul *Gain* din componenta *Math Operations*, pentru amplificare intrări *u* cu factorul de amplificare 3,
- Blocul *Display* pentru vizualizarea rezultatului din componenta *Sinks*.

A2. Construirea diagramei. Pentru realizarea diagramei se procedează în modul următor:

1. Toate blocurile necesare (*Constant, Gain, Display*) vor fi copiate din biblioteca SIMULINK în fereastra de construire a diagramei. Pentru realizarea acestei etape se execută succesiv operaţiile:
   - Se activează fereastra *Source-Library*.
   - Se execută click pe blocul *Constant* cu butonul din dreapta mouse-ului. Cât timp este apăsat butonul, se realizează o copie a blocului şi se plasează în diagrama bloc.
   - Similar sunt copiate blocurile: *Gain* din fereastra *Math Operations* şi respectiv *DisplayBlock* din fereastra *Sink-Library*.

Operaţii uzuale pentru copierea, mutarea şi ştergerea blocurilor:

- copiere - se va utiliza butonul drept al mouse-ului;

- mutarea blocurilor - se va utiliza butonul stâng al mouse-ului;
- ştergerea blocurilor - se va utiliza butonul Delete.

2. Blocurile se vor conecta conform diagramei din figura 1.6. Acestea pot fi conectate prin apăsarea butonului drept al mouse-ului şi tragerea unei săgeţi de la ieşirea unui bloc la intrarea altui bloc. In diagrama astfel obţinută, blocurile pot fi deplasate şi aranjate cu ajutorul butonului stâng al mouse-ului.

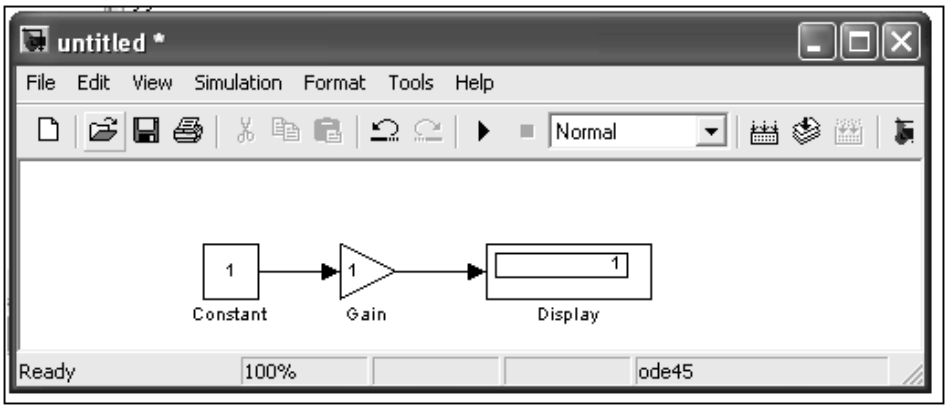


Fig.1.6. Diagrama asociată relaţiei 1.1.

A3. Configurarea blocurilor. Prin configurare se înţelege setarea anumitor parametri numerici asociaţi blocurilor. In cadrul diagramei din figura 1.6, sunt necesare configurarea blocurilor *Constant* şi *Gain.* Etapa de configurare decurge astfel:

(1) Blocul Constant

- Se execută click pe blocul *Constant*;
- In urma activării blocului Constant se va deschide căsuţa de dialog specifică blocului, figura 1.7. In câmpul specific introducerii parametrului se setează:

Constant value: 5

- Validarea valorilor introduse se face prin apăsarea butonului *Apply*, figura 1.7.
- Pentru închiderea căsuţei de dialog se utilizează butonul Close.

(2) Blocul *Gain*

- In urma activării blocului *Gain* se va deschide căsuţa de dialog specifică blocului, figura 1.8
- In câmpul specific parametrilor se setează conform figurii 1.8., parametrul:

Gain: 3

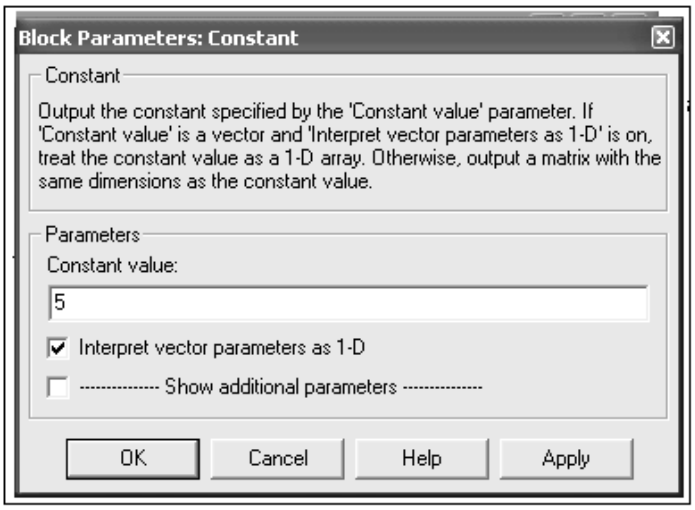


Fig. 1.7. Configurarea blocului *Constant*.

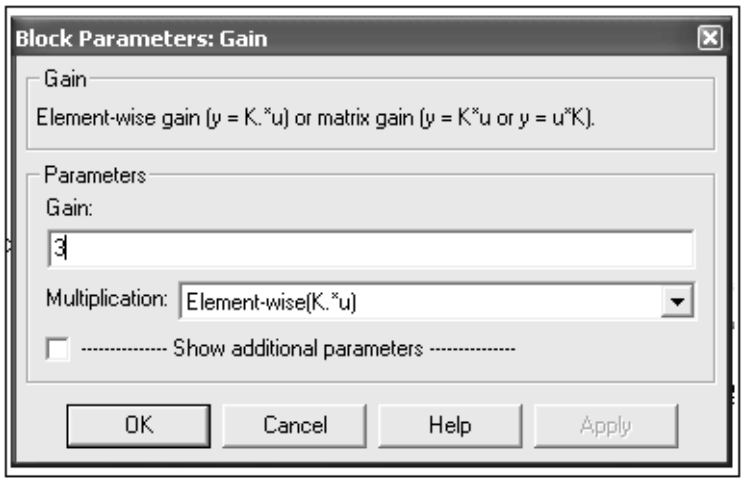


Fig.1.8. Configurarea blocului *Gain.*

Diagrama se va salva sub numele de *prob11*, alegând comanda *File/SaveAs*, cu extensia "mdl".

A4. Lansarea în execuţie. Aceasta este realizată prin comanda *Start* din meniul *Simulation* (figura 1.9). In cadrul blocului *Display* se va obţine rezultatul evaluarii expresiei 1.1.

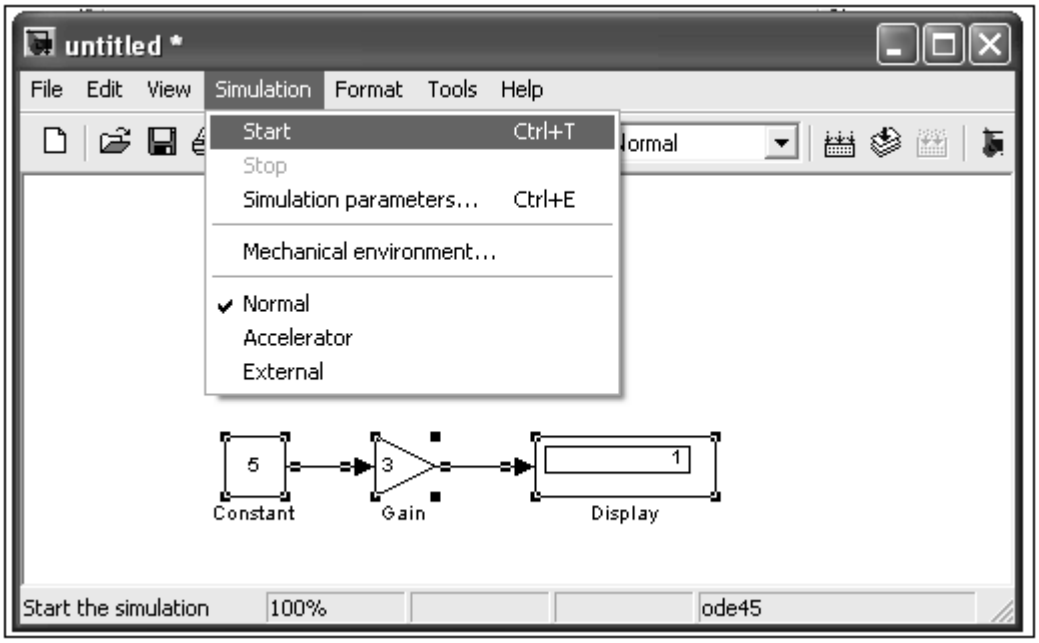


Fig.1.10 .

Exemplul 2. Fie sistemul de ordinul 1, figura 1.10, descris prin modelul matematic:

$$T \cdot \dot{y} + y = K \cdot u \tag{1.2}$$

unde T=0.9 reprezintă constanta de timp a ecuaţiei, exprimată în minute,iar K=3 este factorul de amplificare asociat variabilei de intrare u, $u = 1(t)$.

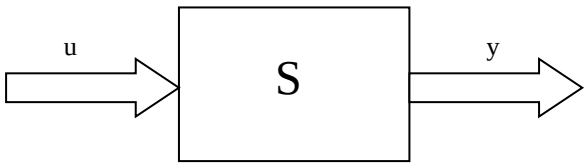


Fig. 1.10. Structura sistemului descris de modelul matematic (1.2).

Condiţia iniţială a ecuaţiei diferenţiale este $y(0) = 1$. Sistemul va fi simulat pe intervalul de timp $[0, 10]$ s.

Rezolvare. Modelul matematic (1.2) se aduce la forma standard (1.3) :

$$\frac{dy}{dt} = \frac{1}{T}\left[-y(t) + Ku(t)\right] \tag{1.3}$$

aceasta se integrează ajungând la expresia (1.4):

$$y(t) = y(0) + \int \frac{1}{T}\left[-y(t) + Ku(t)\right]dt \tag{1.4}$$

Ultima relaţie este implementată prin intermediul diagramei bloc figura 1.12

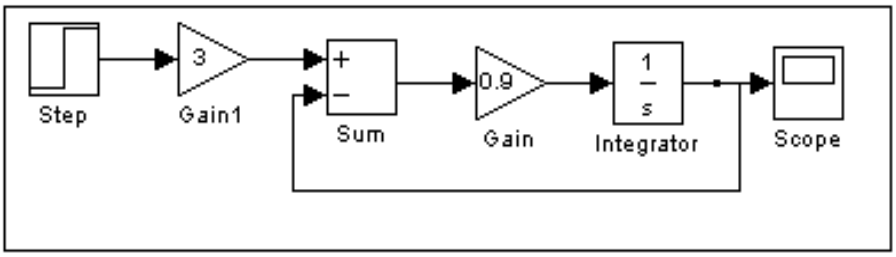


Fig.1.11. Diagrama sistemului de ordinul 1.

Creare diagramă. Pentru realizarea diagramei se vor avea în vedere următoarele etape:

A. Toate blocurile necesare vor fi copiate din biblioteca SIMULINK în schema bloc;
B. Conectarea blocurilor conform diagramei bloc din figura 1.11;
C. Configurarea fiecărui bloc, introducând valorile numerice pentru parametrii blocurilor prin intermediul căsuţei de dialog a fiecărui bloc.

Explicarea mai detaliată a paşilor de realizare a diagramei:

A. Copierea blocurilor în cadrul noii diagrame

- Se deschide Linear-Library. Click pe blocul Integrator cu butonul din dreapta mouse-ului. Cât timp este apăsat butonul se trage o copie a blocului peste schema bloc şi se plasează în diagrama bloc.
- Se copiază două Gain-Block şi un Sum-Block din Linear-Library.
- Se copiază Step-Block din Source-Library.
- Se copiază Scope-Block din Sink-Library.

B. Conectarea blocurilor

Conectarea blocurilor conform diagramei bloc din figura 1.11. Acestea pot fi conectate prin apăsarea butonului drept al mouse-ului şi tragerea unei săgeţi de la ieşirea unui bloc la intrarea altui bloc.

C. Configurarea blocurilor

În căsuţa de dialog a fiecărui bloc vom introduce valorile numerice ale parametrilor în câmpurile specifice acestora:

- Blocul Step necesită setarea parametrilor: grupul valorilor iniţiale la 0; grupul valorilor finale la 1; step time la 0.
- În cadrul blocului Integrator sunt setate: 0 - limita inferioară; 10 - limita superioară; 1 - condiţia iniţială.
- Gain - necesită valoarea 0.9.
- Gain1 - se setează amplificarea la valoarea 3.
- În cadrul blocului Sum se setează întâi ”+” pentru calea directă urmată de “-“ pentru reacţie.

Simulare dinamică. Aceasta este realizată prin comanda *Start* din meniul *Simulation.* Pentru o mai bună reprezentare grafică a rezultatelor, în meniul *Simulation/Parameters/Solver* vor fi setaţi următorii parametrii, figura 1.12 :

- "Start Time" şi "Stop Time" pentru setarea intervalului de integrare;
- "Euler", "Adam" sau "Runge-Kutta" din meniul *Solver Options* pentru alegerea metodei de integrare.

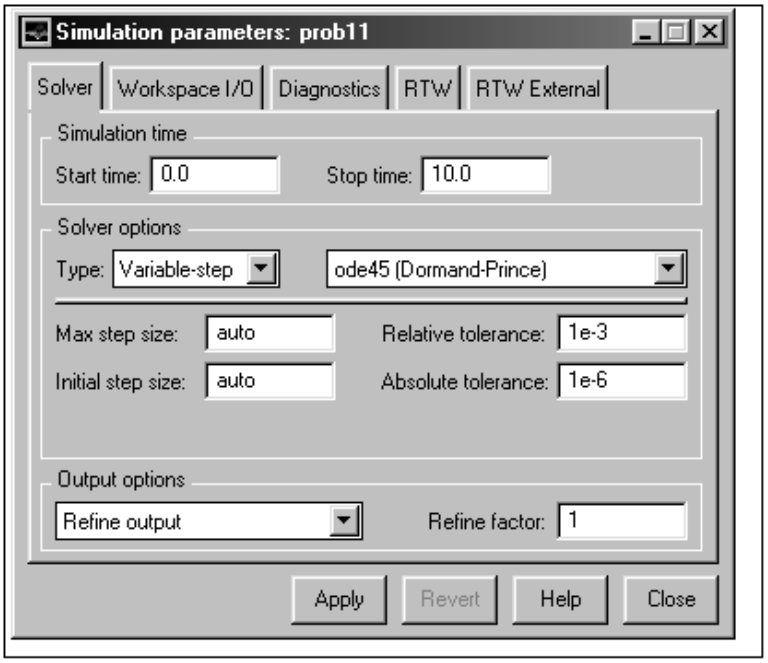


Fig.1.12. Meniul Simulation parameters.

Vizualizarea rezultatelor simulării este accesibilă prin intermediul blocului Scope. Blocul Scope prezintă câteva trăsături, figura 1.13:

- cel mai din stânga buton din fereastra Scope este butonul de *Zoom* care îţi permite să modifici dimensiunile graficului în ambele direcţii, x şi y (alegi aria de modificare cu mouse-ul);
- următoarele două butoane sunt de asemenea butoane *Zoom*. Ele îţi permit să modifici graficul în direcţia x, respectiv y.
- butonul cu binoclul este butonul *Auto-scale* care arată întreg răspunsul în timp (anulând orice modificare).
- următorul buton este butonul *Save-axes* care salvează sau îngheaţă axele .
- cel mai din dreapta buton este butonul *Properties* care permite în meniul *Axes* configurarea blocului Scope. In câmpurile Ymax, Ymin se introduc valori pentru determinarea limitelor de reprezentare pe axa y, figura 1.5.

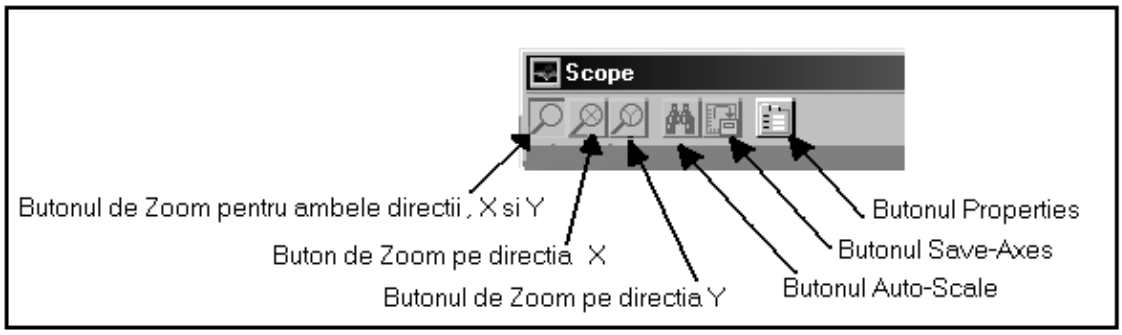


Fig.1.13. Bara de obiecte a blocului Scope.

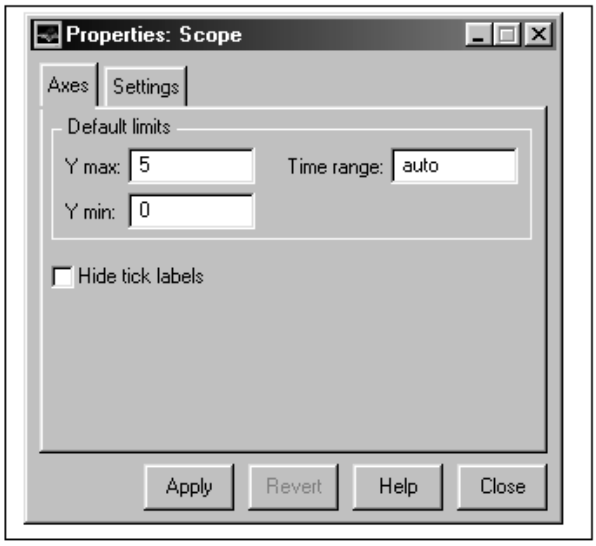


Fig.1.14. Căsuţa de dialog a butonului Properties.

În figura 1.15 este redată dinamica ecuaţiei de ordinul 1 (1.2) obţinută în condiţiile impuse. Răspunsul este aperiodic de ordinul 1.

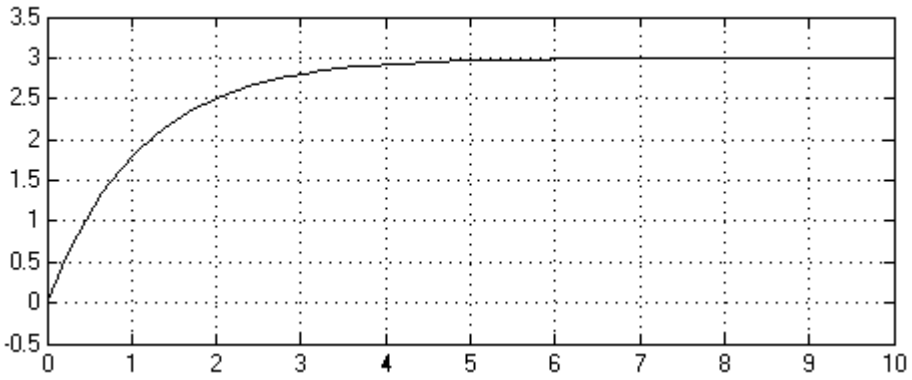


Fig.1.15. Rezultatul obţinut prin simularea sistemului de ordinul 1.

Exemplu 3. Să se construiască diagrama care va implementa relaţia (1.2) de la exemplul 2, utilizând *blocul Subsystem*.

În cazul în care o diagramă devine foarte complicată, Simulink oferă posibilitatea de a grupa mai multe blocuri într-un singur bloc, utilizând blocul Subsystem din componenta Port &Subsystem. Ca exemplu, vom crea un subsistem ce va conţine toate blocurile dintre blocul Step şi blocul Scope din figura 1.11. Diagrama rezultată se va fi prezenta în figura 1.16. Dacă se va executa click pe blocul Subsystem se va deschide o noua diagramă, figura 1.17.

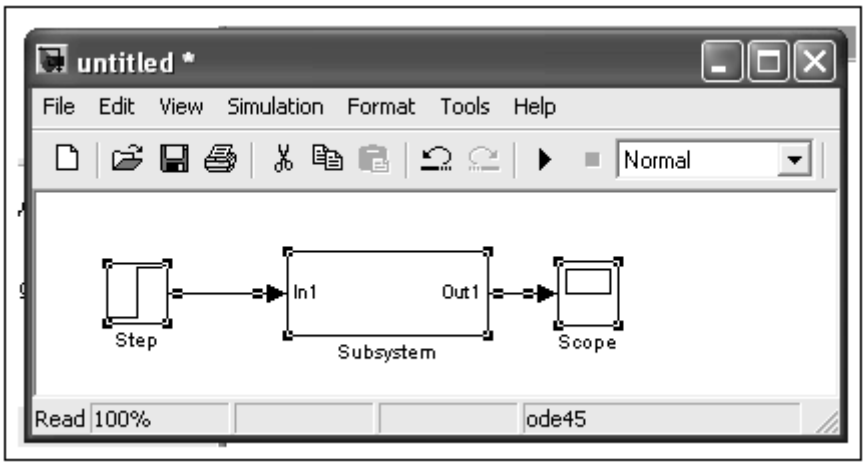


Fig. 1.16.

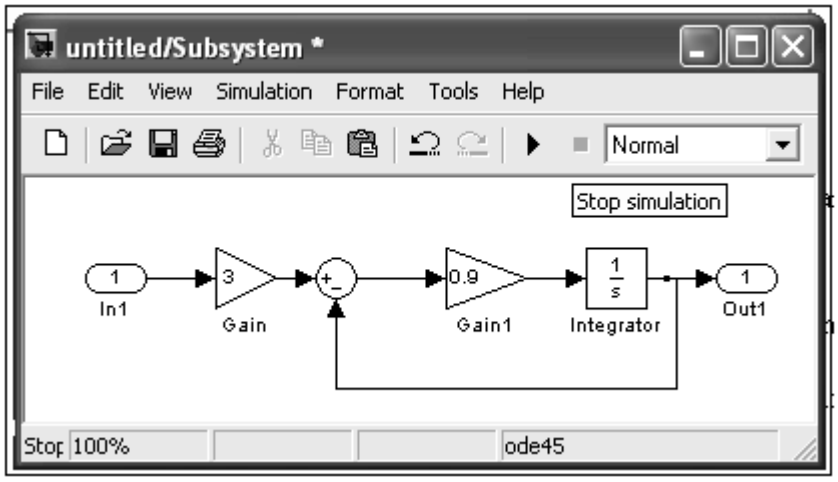


Fig.1.17.

1.4. Întrebări şi exerciţii

1. Care sunt avantajele mediului de simulare Simulink?
2. Să se construiască diagrama de simulare pentru următoarele ecuaţii:
   a) y=2u, unde u=4;
   b) $y = 3 \cdot u + e^u$, u=5;
   c) $3 \cdot \dot{y} + y = 2 \cdot u$, y(0)=0, u(0)=1,5;
3. Să se construiască diagrama ce realizează conversia gradelor Celsius în grade Fahrenheit.

$$T_F = \frac{9}{5} T_C + 32$$

LUCRAREA 2

ANALIZA EXPERIMENTALĂ A DINAMICII SISTEMELOR

2.1. Obiectivele lucrării

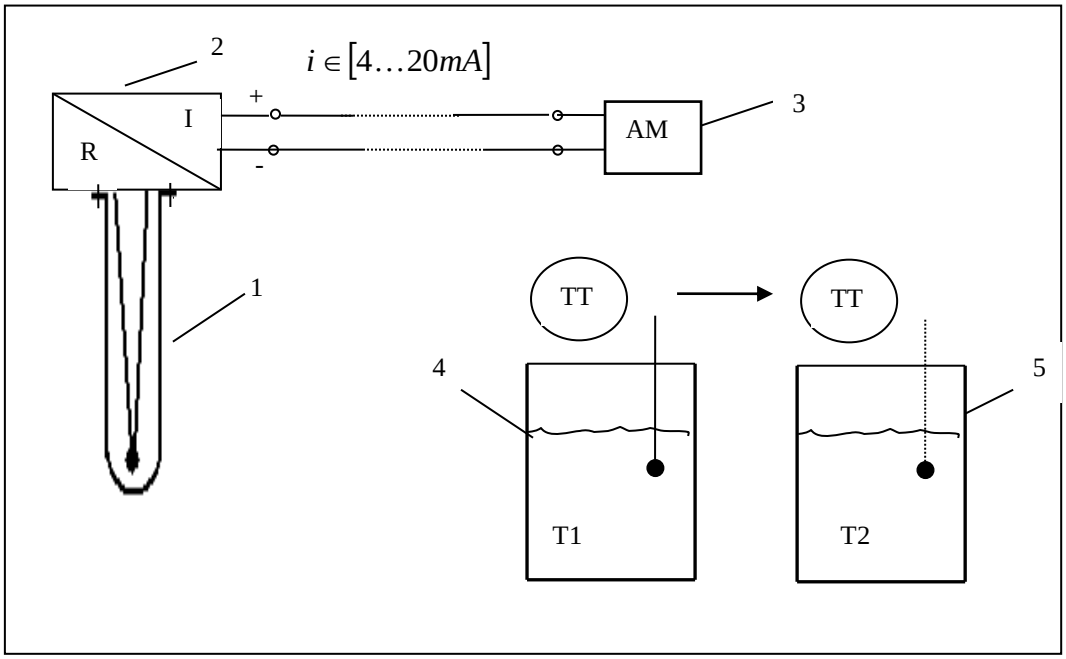
În prezenta lucrare, căreia îi sunt afectate patru ore, se urmăreşte atingerea următoarelor obiective:

- înţelegerea noţiunilor: sistem tehnic, dinamica unui sistem tehnic, simulare;
- cunoaşterea elementelor de baza privind dinamica sistemelor.

2.2. Prezentarea conţinutului lucrării

În cele ce urmează va fi prezentat modul de determinare experimentală a modelului matematic dinamic asociat unei termorezistente.

Pentru o termorezistenţă PT100[1], prevăzută cu un adaptor încorporat *rezistentă-intensitate*, care generează la ieşire 4-20mA, se va determina experimental caracteristica dinamică. In figura 2.1 sunt prezentate elementele de baza necesare efectuării experimentului.



Fg.2.1.Determinarea experimentală a caracteristicii dinamice a unei termorezistenţe;

1- termorezistenţa; 2- adaptorul rezistenţa-intensitate; 3- aparat de măsurat;

4,5- vase cu temperaturi T_1 şi T_2 diferite.

[1] Termorezistenţă cu R=100 Ω la T=100^{0}C.

Prin trecerea rapidă din vasul 4 în vasul 5, sau invers, lichidele din vase având temperaturile $T1 \neq T2$, se aplică la intrarea termorezistentei un semnal treaptă. După aplicarea treptei de temperatură la intrare, se notează valorile intensităţii **i** în timp.

2.3. Partea experimentală

1. Se va determina experimental răspunsul în timp, la variaţia treaptă, pentru traductorul de temperatură de tip termorezistentă, conform celor precizate în paragraful 2.2. Se recomandă următoarele:

- se va face un experiment de probă pentru formarea unei imagini de ansamblu şi în mod deosebit, pentru stabilizarea scării de timp şi a intervalelor de timp la care se vor face citirile;
- se vor determina două răspunsuri în timp: pentru treapta pozitivă (de la rece la cald) şi pentru treapta negativă(de la cald la rece);

2. Pe baza răspunsurilor în timp de la punctul 1 se va determina MMD al traductorului. In acest scop se propun trei forme de modele matematice dinamice (MMD) de aproximare:

- MMD de ordinul I în variaţii

$$a\frac{d\Delta i}{dt} + \Delta i = b\Delta T, \tag{2.1}$$

- MMD de tip functie de trasfer

$$H(s) = \frac{b}{as+1}, \tag{2.2.}$$

- MMD cu valori absolute

$$a\frac{di}{dt} + i = bT; \tag{2.3}$$

Constantele **a** şi **b** ale relaţiilor anterioare se vor determina astfel:

- Constanta **b** se va calcula pe baza relaţiei

$$b = \frac{i}{T}, \tag{2.4}$$

- Constanta **a** se va calcula pe baza relaţiei

$$a = \frac{a_1 + a_2 + a_3}{3}, \tag{2.5}$$

Pentru a determina constanata $\mathbf{a_1}$ se va alege un punct pe grafic, prin care se duce o tangenta în acel punct la grafic. Tangenta va intersecta dreapta ce determina starea staţionara a sistemului. Distanţa dintre punctul de pe grafic şi punctul determinat de intersecţia tangentei cu dreapta ce determină starea staţionara a sistemului, se va nota cu a_1, figura 2.2 . In acelaşi mod se vor determina şi celelalte două constante a_2, a_3.

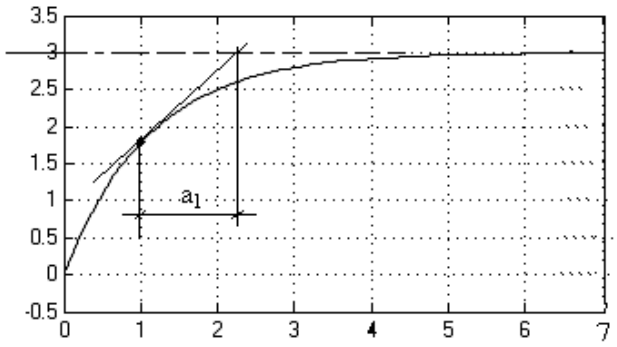


Fig. 2.2.

4. Pentru sistemul

$$a\frac{di}{dt} + i = bT,$$

unde **a** şi **b** au valorile determinate la punctul 2, să se determine **i(t)** şi graficul său atunci când temperatura T variază sub forma de treaptă de la 20 ^{0}C la 90 ^{o}C.

4. Pentru sistemul de la punctul 3 se va construi diagrama de simulare a sistemului în mediul de programare Simulink, conform celor învăţate în cadrul primei lucrării de laborator.

2.3. Întrebări şi exerciţii

1. Să se facă aprecieri privind calitatea MMD utilizat la aproximarea dinamicii traductorului de temperatură investigat în paragraful 2.3.
2. Puteţi explica conţinutul noţiunii dinamica unui sistem tehnic?
3. Sa se construiască raspunsul în timp al următoarei ecuaţii diferenţiale

$$4\frac{dT}{dt} + T = 0{,}35 \cdot Q_c$$

unde $Q_c(t)=1000 m^3N/h$; $T(0)=335^oC$.

4. Pentru sistemul din figura 2.3. sunt presupuse următoarele modele matematice dinamice:
   a) y=2u;
   b) $3\frac{dy}{dt} = u$; y(0)=0;
   c) $3\frac{dy}{dt} + y = 2u$ y(0)=0;

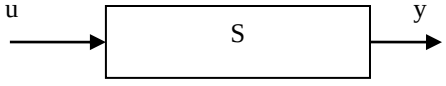


Fig. 2.3 Sistem monovariabil.

Să se construiască, pentru fiecare caz în parte, caracteristica dinamică a sistemului atunci când intrarea u(t) variază ca în figura 2.4.

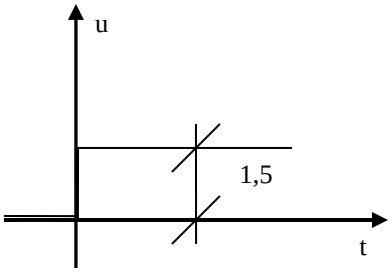


Fig. 2.4. Variaţia în timp a intrării u(t).

LUCRAREA 3

SISTEME DE MĂSURAT

3.1. Obiectivele lucrării

În cadrul acestei lucrări, căreia îi sunt afectate două ore (o şedinţă de laborator), se urmăreşte atingerea de către studenţi a următoarelor obiective:

- înţelegerea şi însuşirea corectă a noţiunii *măsurare*;
- însuşirea structurii unui *sistem de măsurat* (SM);
- cunoaşterea fizică a unor sisteme de măsurat din laborator;
- înţelegerea caracteristicii statice a unui element fizic (traductor sau aparat de afişare).

3.2. Prezentarea conţinutului lucrării

- Pentru noţiunea *măsurare* se va purta o discuţie în laborator care să aibă la bază cunoştinţele studenţilor asupra acestei noţiuni (predate la curs) şi exemple fizice concrete din activitatea practică.
- Un sistem de măsurat se compune principial din două părţi (fig. 3.1):
  - traductorul, care sesizează mărimea din proces (temperatură, debit, nivel, presiune etc.) şi generează la ieşire un semnal corespunzător;
  - aparatul de afişare (indicare, înregistrare) sau de măsurare, care prin prelucrarea semnalului primit de la traductor determină şi afişează valoarea mărimii din proces.
- Cu ajutorul cadrului didactic, studenţii vor cunoaşte fizic, principial, următoarele SM: SM-P, SM-N, SM-D şi SM-T. (P – *presiune*, N – *nivel*, D - *debit*, T – *temperatură*).

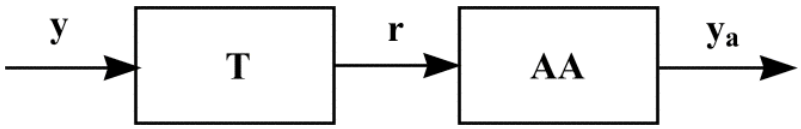


Fig. 3.1. Cele două părţi importante ale unui SM:
T – traductorul; AA – aparatul de afişare;
y – mărimea din proces (debit, presiune, nivel etc.); y_a – mărimea afişată.

- Divizaţi în două semigrupe, studenţii vor determina caracteristica statică a unui traductor de nivel (LT) sau a unui traductor de presiune (PT) şi a unui aparat de afişare.
- Se va prezenta şi discuta principial schema unui potenţiometru electronic (fig. 3.5 din subcap. 3.4);

- Se va prezenta principial structura unui SM realizat cu ajutorul unui calculator de proces.

3.3. Partea experimentală

Această parte cuprinde:

– cunoaşterea fizică a SM amintite în subcapitolul anterior;

– determinări experimentale asociate SM-P şi SM-N.

- Cunoaşterea fizică, principială, a celor patru SM se va realiza prin prezentarea acestora de către cadrul didactic, în laborator. Se va accentua componenţa acestora prin punerea în evidenţă a traductorului şi a aparatului de afişare. Se va observa dacă există asemănări între aparatele de afişare ale celor patru sistem de măsurat.
- Pentru studiul experimental al SM-P va fi utilizat montajul din fig. 3.2.

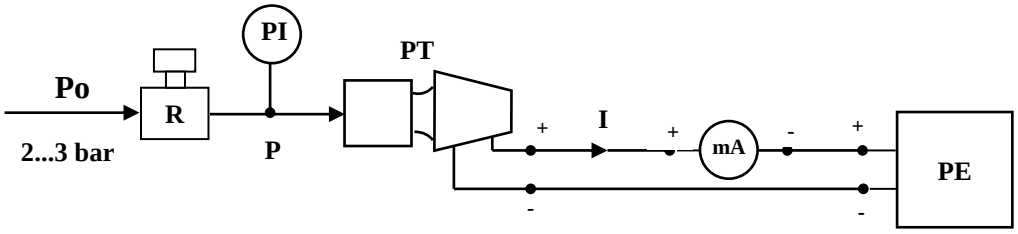


Fig. 3.2 Montaj pentru studiul experimental al SM-P:

R – reductor de presiune; PI – manometru (presiune indicată); mA – miliampermetru; PE – potenţiometru electronic.

Modul de lucru este prezentat în continuare.

1. Se notează mai întâi domeniile mărimilor de intrare şi ieşire pentru PT şi PE.

2. Cu ajutorul reductorului R se dau 6...10 valori presiunii P. pentru fiecare valoare a presiunii P se citesc valoarea intensităţii I şi valoarea indicată de potenţiometrul electronic PE. Valorile presiunii P, intensistăţii I şi celei indicate de potenţiometrul PE se trec într-un tabel de forma celui cu nr. 3.1.

Tabelul 3.1

Valori măsurate pentru SM – P

Presiunea P [bar]	P_1	P_2	. . .	P_n
Intensitatea I [mA]	I_1	I_2	. . .	I_n
Indicaţia PE	I_{PE1}	I_{PE2}	. . .	I_{PEn}

3. Cu datele din tabelul 3.1 se vor construi pe hârtie milimetrică, graficele $I = f(P)$ şi $I_{PE} = g(I)$. Aceste grafice reprezentând *caracteristicile statice* ale celor două elemente.

4. Se va determina ecuaţia caracteristicii statice (ca dreapta ce trece prin două puncte de coordonate curente) .

- Pentru studiul experimental al SM – N va fi utilizat montajul din fig. 3.3. Modul de lucru este prezentat în continuare.

1. Se notează mai întâi domeniile mărimilor de intrare şi ieşire pentru LT şi PE.

2. Cu ajutorul pompei SP se transportă apa din partea inferioară a vasului V în partea superioară a acestuia.

3. După cum se observă din figură, nivelul apei este măsurat în partea superioară a vasului. Modificarea acestuia se realizează prin deschiderea treptată a supapei dintre cele două compartimente ale vasului V.

Pentru fiecare valoare a nivelului apei în compartimentul superior al vasului se citesc valorile pentru nivelul H (cu ajutorul sticlei de nivel SN), pentru intensitatea I şi pentru indicaţia I_{PE} a potenţiometrului electronic PE. Valorile mărimilor H, I şi I_{PE} pentru cele 6... 10 determinări se trec într-un tabel asemănător cu tabelul 3.1.

4. Cu datele tabelate se construiesc, pe hârtie milimetrică, graficele I = f(H) şi I_{PE} = g(I). Aceste grafice reprezintă *caracteristicile statice* ale celor două elemente. Pe baza analizei graficelor se vor determina concret expresiile funcţiilor f şi g.

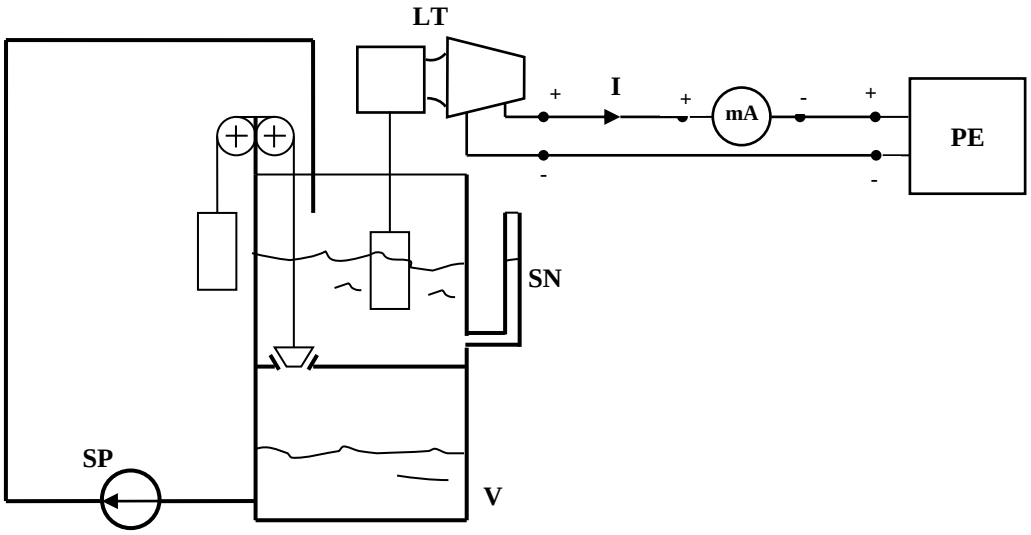


Fig. 3.3. Montaj pentru studiul experimental al SM-N:

LT- traductor de nivel; mA - miliampermetru; PE- potenţiometru electronic; SN- sticlă de nivel; SP- pompă centrifugală; V- vas cu două compartimente.

- In cadrul lucrării de laborator va fi prezentat, de asemenea un SM realizat cu calculatorul electronic (fig. 3.4).

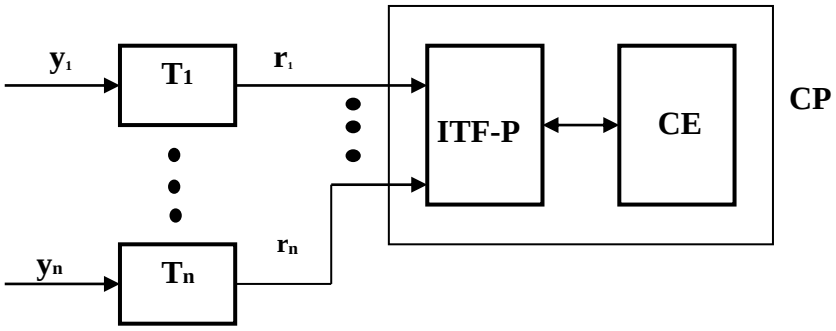


Fig. 3.4 Structura principiala a unui SM cu CE:

T_1,…, T_n-traductoare; ITF-P- interfaţă de proces; CE-calculator electronic; CP-calculator de proces.

3.4. Intrebări şi exerciţii

1. Stiţi ce este un manometru ? Solicitaţi cadrului didactic sau tehnicianului să vă arate un manometru secţionat. Manometrul conţine un traductor? Dar un aparat de afişare ?

2. Ce natură fizică are semnalul r pentru SM-P analizat ?

3. Care este domeniul de variaţie al acestui semnal ?

4. Prelucraţi datele experimentale obţinute la cele două SM cu ajutorul calculatorului electronic.

5. Să se construiască, pentru SM-P şi SM-N, graficele $I_{PE} = f_1(P)$ şi $I_{PE} = f_2(H)$. Ce semnificaţie au cele două grafice ?

6. In fig. 3.5 se prezintă schema principială a potenţiometrului electronic. Încercaţi să evidenţiaţi modul în care funcţionează potenţiometrul electronic(se va apela la cunoştinţele predate la curs.). Se va avea în vedere faptul că acesta trebuie să determine valoarea tensiunii U_i.

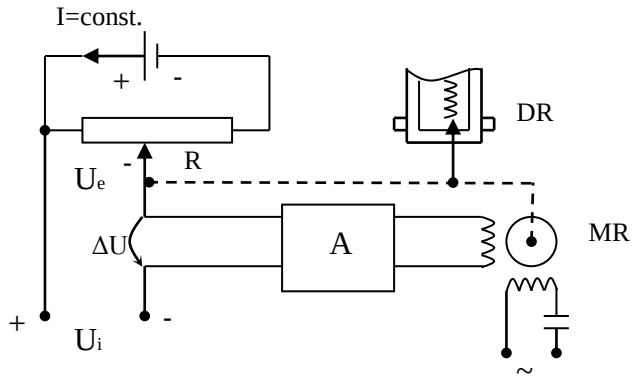


Fig.3.5 Schema principiala a potentiometrului electronic:

A-amplificator; MR-motor reversibil; DR-dispozitiv de inregistrare; E-sursa de alimentare; R-potentiometru bobinat; U_i-tensiune aplicata la intrare: U_e -tensiune culeasă între potenţiometru şi cursor.

LUCRAREA 4

CUNOAŞTEREA EXPERIMENTALĂ A REGULATOARELOR ANALOGICE ŞI NUMERICE

4.1. Obiectivele lucrării

Prin efectuarea acestei lucrări de laborator, pentru care sunt alocate două ore, se urmăreşte atingerea următoarelor obiective:

- familiarizarea cu funcţionarea unui regulator analogic;
- familiarizarea cu funcţionarea unui regulator numeric ;
- determinarea experimentală a caracteristicii statice (CS) a unui regulator proporţional, cu aplicaţie la un regulator numeric SHIMADEN.
- implementarea regulatoarelor PI si PID cu ajutorul mediului de programare Simulink.

4.2. Prezentarea conţinutului lucrării

În cadrul acestei lucrări se vor prezenta regulatoarele analogice şi numerice destinate SRA după abatere.

4.2.1. Regulatoare analogice

Regulatorul reprezintă unul dintre elementele de bază ale dispozitivului de automatizare şi este destinat elaborarii comenzii printr-o prelucrare adecvată a abaterii, abatere rezultată ca urmare a comparaţiei între referinţă şi reacţie. Această definiţie este valabilă pentru *regulatoarele destinate SRA după abatere*, spre deosebire de cele destinate SRA după perturbaţie la care algoritmul pentru determinarea comenzii este specific aplicaţiei.

Regulatorul este caracterizat prin mărimile din figura 4.1.

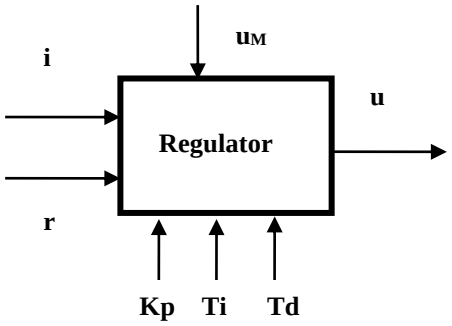


Fig. 4.1. Mărimi caracteristice ale regulatorului după abatere:

i - referinţă; r - reacţie; Kp,Ti,Td - parametrii de acordare ai regulatorului; u_M - comandă manuală;

Algoritmi pentru determinarea comenzii. La regulatoarele PID (proportional-integrator-derivator) comanda se elaborează prin prelucrarea abaterii conform relaţiei:

$$u = u_0 + K_p\left(e + \frac{1}{T_i}\int_0^t e\,dt + T_d\,\frac{de}{dt}\right),$$

unde: u este valoarea curentă a comenzii;
u_0 - valoarea comenzii în absenţa abaterii;
K_p - factorul de proprţionalitate;
T_i - constanta de integrare;
T_d - constanta de derivare.

Din analiza relaţiei rezultă proporţionalitatea comenzii cu *abaterea, integrala şi derivata acesteia*.

Dând valori convenabile parametrilor T_i şi T_d se pot obţine algoritmi specifici regulatoarelor P (proporţional), PI (proporţional-integrator) şi PD (proporţional-derivator), aşa cum este aratat în notele de curs.

Din punctul de vedere al realizării fizice regulatoarele pot fi analogice şi numerice. Cu toate că astăzi se utilizează în exclusivitate regulatoarele numerice, se consideră necesară prezentarea succintă a elementelor regulatorului analogic ELC 113, fabricat şi utilizat în România. La prezentarea standului „Studiul regulatoarelor analogice" se va evidenţia rolul regulatorului analogic ELC 113 şi se va observa că panoul frontal al acestuia permite afişarea referinţei abaterii şi comenzii.

4.2.2. Regulatoare numerice

În cadrul acestei lucrări de laborator se va studia experimental un regulator numeric SHIMADEN.

Principalele mărimi caracteristice ale regulatorului numeric după abatere sunt aceleaşi cu cele prezentate în figura 4.1.

Acest tip de regulator se încadrează în clasa regulatoarelor automate discrete, (prelucrarea semnalelor se face în formă numerică) şi prezintă următoarele caracteristici generale:

- este destinat unei singure bucle de reglare;
- are la bază un microprocesor specializat ;
- poate comunica prin legătură serială cu nivelul ierarhic superior;
- oferă facilităţi sporite în ceea ce priveşte algoritmii de reglare;
- oferă contacte de ieşire care pot fi integrate în sisteme de avertizare, protecţie sau comandă;
- parametrii de acordare BP, Ti, Td pot fi modificaţi local (de la tastatură) sau de la distanţă (prin linia serială);
- are posibilitatea acordării automate (autotuning);
- posedă convertoare analog/numerice şi numeric/analogice necesare pentru trecerea de la semnal continuu (analogic) la cel discret (numeric) şi invers;
- la comutările A/M şi M/A echilibrările se fac automat.

4.3. Partea experimentală

4.3.1. Prezentarea standului experimental

Determinările experimentale se execută la standul "Studiul regulatoarelor numerice". Standul cuprinde următoarele elemente:

- regulatoare numerice SHIMADEN şi FOXBORO;
- elemente de comandă manuală ELX 227 destinate generării semnalelor de reacţie pentru regulatoare;
- înregistrator ELR 35, care vizualizează comanda generată de regulatorul SHIMADEN;

4.3.2. Desfăşurarea lucrării

Înaintea determinărilor experimentale se vor identifica elementele componente de pe faţa panoului, precum şi elementele panoului frontal al regulatorului numeric SHIMADEN, elemente ce sunt prezentate în a*nexa 4.2* a acestei lucrări. Operarea regulatorului SHIMADEN şi diagramele de operare sunt prezentate în *anexele 4.1*, respectiv 4.3.

Determinarea caracteristicii statice a regulatorului P

Familia de caracteristici statice *u=f(r,BP)* se va trasa pentru situaţia în care prescrierea *i* este fixată, iar **BP** este parametru.

În vederea obţinerii acestei familii de caracteristici se vor parcurge următoarele etape:

a) scrierea expresiei analitice a CS a regulatorului **P** sub forma *u=f(r,i,BP)*;
b) pe baza relaţiei scrise la punctul a) se va determina analitic valoarea comenzii pentru situaţia *i=r* şi se vor formula observaţii privind dependenţa acesteia de parametrul **BP**;
c) se alimentează panoul cu energie, numai în prezenţa cadrului didactic;
d) se va efectua configurarea regulatorului SHIMADEN, în vederea obţinerii algoritmului proporţional;
e) cu regulatorul în modul **M** (manual) se stabileşte , prin acţionarea tastaturii, valoarea comenzii u_0 (exemplu 50%) care se citeşte la înregistrator şi/sau pe ecranul regulatorului;
f) de la elementul de generare a reacţiei ELX 227 se stabileşte o anumită valoare pentru mărimea de reacţie, *r* (exemplu 50%), care se citeşte pe indicatorul aparatului respectiv sau pe ecranul regulatorului;
g) se stabileşte o valoare identică pentru mărimea prescrisă, *i* (*r=i=50%*), care se citeşte în zona de afişare a panoului frontal, în aşa fel încât abaterea să fie nulă;
h) se va fixa o valoare pentru **BP** (exemplu BP=100%);
i) se trece regulatorul pe modul **A** (automat) şi se modifică *i* cu raţia de 10%, până când se acoperă tot domeniul [0…100%], citindu-se la înregistrator valorile corespunzătoare ale comenzii *u*.

Observaţie. La comutarea M/A indicaţia înregistratorului trebuie să rămână pe valoarea u_0 stabilită la punctul e), în condiţiile abaterii nule.

Experimentul se repetă pentru încă două valori ale parametrului BP cuprinse în intervalul 50…200. Rezultatele experimentale se trec într-un tabel de felul următor:

Tabelul 3.1

Rezultate experimentale privind CS a regulatorului proporţional

u_0			**%**			**r**			**%**		
			mA						**mA**		
K_p=				**K_p=**				**K_p=**			
i		**u**		**i**		**u**		**i**		**U**	
%	**mA**	**%**	**mA**	**%**	**mA**	**%**	**mA**	**%**	**mA**	**%**	**mA**
K_p^* efectiv=				**K_p^* efectiv=**				**K_p^* efectiv=**			

* Parametrul $K_p$efectiv se calculează.

Se reia experimentul pentru a se observa că <u>la intrări egale şi momente de timp diferite regulatorul proporţional generează comenzi egale.</u>

4.3.3. Prelucrarea datelor experimentale

Pe baza rezultatelor experimentale obţinute se trasează pe hârtie milimetrică familia de caracteristici pentru cele trei valori ale parametrului BP. Se va determina valoarea efectivă a parametrului BP.

4.4. Implementarea regulatoarelor PI si PID in mediul de programare SIMULINK

În biblioteca mediului de programare Simulink există componenta *Simulink Extras*, în cadrul căreia se găseşte instrumentul *Additional Linear*. Acesta conţine, pe lângă alte blocuri, şi blocurile corespunzătoare regulatorului PID şi regulatorului PID cu aproximarea componentei derivatoare. Regulatorul PID este unul ideal, care este de preferat a nu se folosi singur, ci conectat în serie cu un filtru trece-jos (cu funcţia de transfer de ordinul I). În cele mai multe cazuri poate fi utilizat regulatorul PID cu aproximarea componentei derivative. Parametrizarea regulatorului PID se face în fereastra de dialog corespunzătoare acestuia, P fiind factorul de proporţionalitate K_p, I reprezentând K_p/T_i (T_i este timpul de integrare), iar D fiind asociat factorului K_p*T_d (T_d este timpul de derivare). N este utilizat pentru $1/T_f$, unde T_f este constantă de timp a filtrului).

Pentru o mai bună configurare este de preferat implementarea propriului regulator, utilizând blocurile din *Linear Library*.

4.5. Întrebări şi exerciţii

1. Ce este un regulator?
2. În ce constă diferenţa între un regulator analogic şi unul numeric?
3. Ce reprezintă mărimea u_0?
4. Să se scrie algoritmii de reglare pentru regulatoarele **R-P**, **R-PI** şi **R-PD**.
5. Determinaţi pe cale grafică răspunsul unui algoritm **PI** pentru situaţia din figura 4.2.
6. Puteţi spune dacă regulatoarele numerice sunt superioare celor analogice, şi dacă da, de ce?
7. Numiţi cei trei parametri de acordare ai unui regulator PID.
8. Ce înseamnă modul automat al regulatorului SHIMADEN? Dar modul manual?

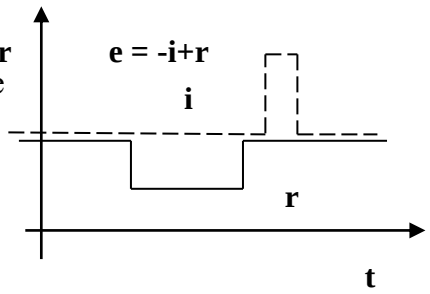


Fig. 4.2

9. Cum se execută comutarea A/M la un regulator analogic? Dar la un regulator numeric?

10. Să se implementeze în mediul Simulink un regulator PI şi un regulator PID,utilizând blocurile din *Linear Library*.

ANEXA 4.1

OPERAREA REGULATORULUI SHIMADEN

Regulatorul **SHIMADEN** are opt moduri de funcţionare (*0, 1, 2,…,7*), fiecare mod având mai multe submoduri. Trecerea de la un mod la altul se face cu ajutorul tastei **MODE**, iar în cadrul unui mod, trecerea de la un submod la altul cu ajutorul tastei **PARA**. Pentru modificarea valorii şi/sau tipului unui anumit parametru se vor utiliza tastele: <(**REMOTE**),V(**MAN**), Λ(**AT**). Validarea oricărei modificări se face apăsând tasta **ENT**. Revenirea din orice mod sau submod la ecranul de bază se face apăsând de două ori tasta **RET(** a doua apăsare trebuie efectuată la cel mult două secunde faţă de prima). Eventualele erori sunt aduse la cunoştinţa utilizatorului prin mesaje corespunzătoare.

În continuare sunt prezentate principalele probleme aferente operării regulatorului.

Comutarea A/M Din ecranul de bază (*mod 0-0*), apăsând tasta **PARA**, se ajunge în ecranul "*out*" (*submodul 0-1*). În acest ecran, apăsând tastele **RET**(**SHIFT**) + V(**MAN**) simultan, se aprinde lampa **MAN** ceea ce indică "cuplarea" modului de comandă manuală. Setarea valorii ieşirii (comenzii) se face folosind tastele <, Λ şi V şi se validează cu tasta **ENT.** Domeniul de ajustare al comenzii: -10.0% … +110.0%.

Pentru introducerea în modul automat, se utilizează aceleaşi două taste, în acelaşi ecran "*out*". După întoarecerea în modul automat, lampa **MAN** se va stinge.

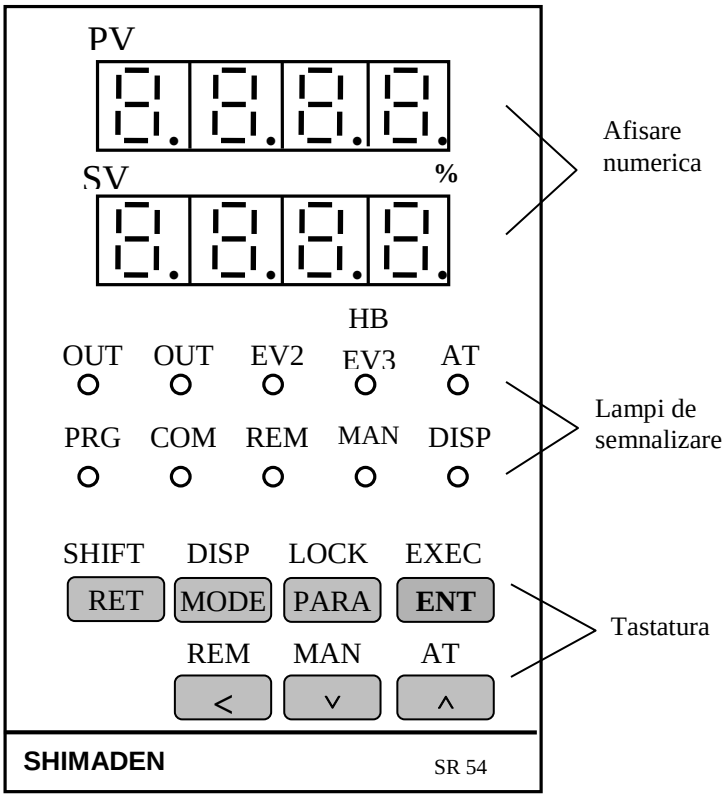
Pentru observarea simultană a valorii din proces **PV** şi a valorii comenzii se specifică modul *0-2* (care are pe afişajul **PV** (de sus) valoarea din proces, iar pe afişajul **SV** (de jos) valoarea ieşirii (comenzii)).

Setarea tipului de algoritm şi modificarea parametrilor de acordare. Atât pentru setarea tipului de algoritm (**P, I, PI, PD, PID**), cât şi pentru setarea corespunzătoare a parametrilor de acordare este necesară parcurgerea submodurilor *0.9* (pentru **BP**), *0.10* (pentru **Ti**) şi *0.11* (pentru **Td**). Acţionând tastele: <,V şi Λ se poate seta o anumită valoare a parametrului de acordare respectiv. În cazul depăşirii limitelor domeniului valid pentru componenta respectivă, în zona de afişare va apare mesajul "*off*". Pentru validarea modificărilor se va apăsa tasta **ENT.**

Modificarea referinţei (SV). În condiţiile ecranului de bază (*modul 0.0*), în modul automat, prin apăsarea tastei **<** se va aprinde cea mai din dreapta cifră a ecranului **SV**, care va începe să clipească, aceasta însemnând că poate fi modificată. Când se apasă din nou tasta <, va începe să clipească a doua cifră din drepta. Pentru modificarea valorii respective se vor utiliza tastele Λ şi V, iar pentru validare tasta **ENT.**

ANEXA 4.2

PANOUL FRONTAL AL REGULATORULUI *SHIMADEN*



Panoul de operare al acestui tip de regulator conţine trei zone şi anume:

- zona de afişare;
- zona de semnalizare;
- zona de operare.

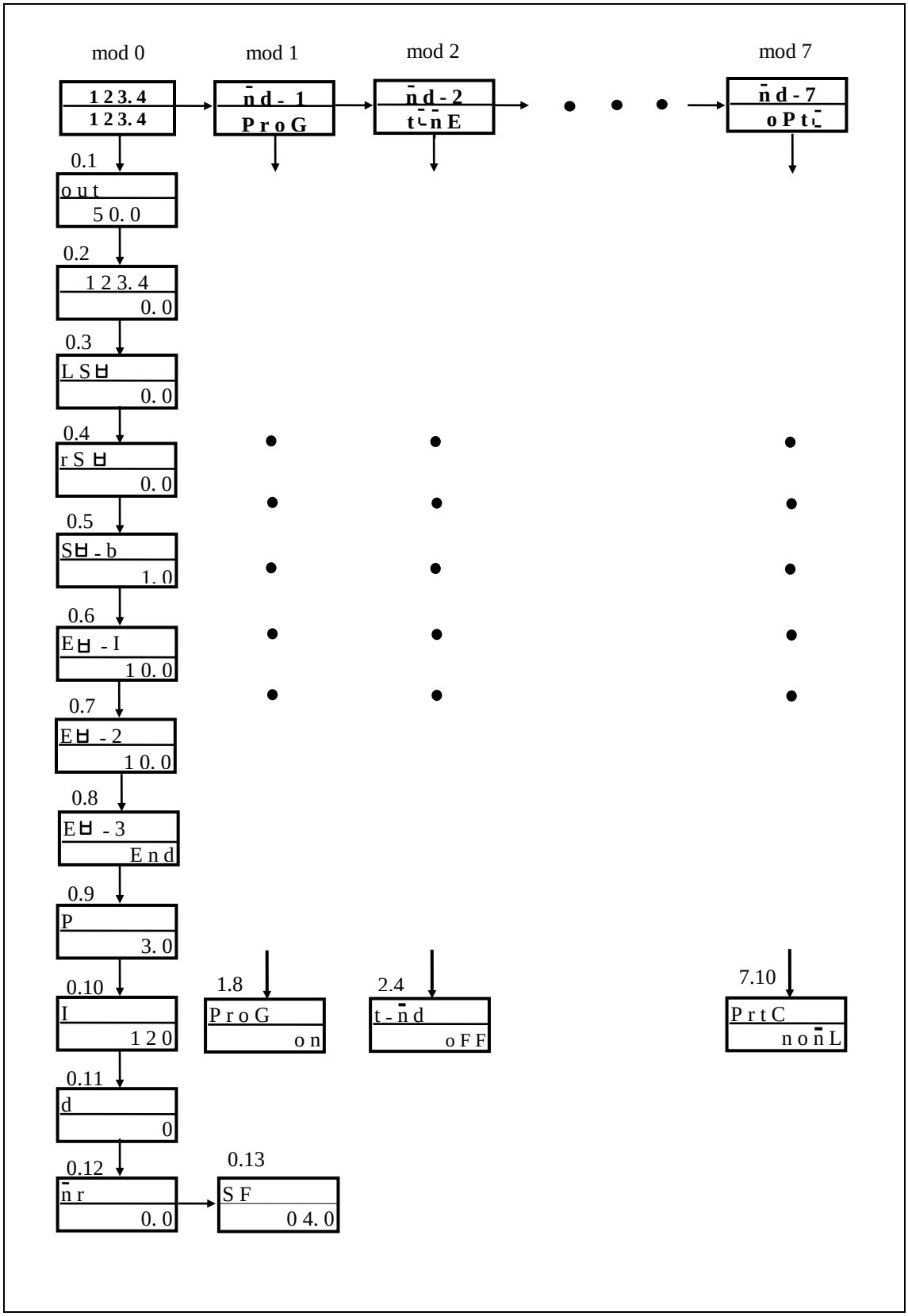
Zona de afişare permite informarea utilizatorului în legătură cu valoarea variabilelor asociate reglării, valoarea unor parametrii ai regulatorului, modurile şi submodurile de lucru. În ecranul de bază (*modul 0.0*) sunt afişate în procente reacţia (**PV** - "Process Value") şi referinţa (**SV** - "Set Value").

Zona de semnalizare realizează avertizarea optică în legătură cu starea variabilelor procesului reglat şi cu configuraţia şi modul de funcţionare ale regulatorului.

Zona de operare permite, prin intermediul a 7 taste cu dublă funcţie, introducerea de date şi modificarea (setarea) configuraţiei regulatorului.

ANEXA 4.3

DIAGRAME DE OPERARE ALE REGULATORULUI *SHIMADEN*



LUCRAREA 5

ELEMENTE DE EXECUŢIE

5.1. Obiectivele lucrării

În această lucrare de laborator se urmăreşte rezolvarea următoarelor obiective:
- cunoaşterea principială a construcţiei şi a funcţionării unui robinet de reglare (RR);
- analiza teoretică şi experimentală a regimului staţionar al RR;
- însuşirea deprinderilor de calcul pentru dimensionarea hidraulică şi alegerea unui RR.

5.2. Prezentarea funcţională a RR

Robinetul de reglare este element de execuţie în cadrul unor sisteme de reglare automată (cu acţiune după abatere sau după perturbaţie), care permite modificarea debitului de fluid care circulă prin el, ca urmare a variaţiei comandate a secţiunii de trecere a sistemului de strangulare.

Mărimea de intrare a RR este un semnal pneumatic unificat (p_c = 0,2 ... 1 bar) sau un semnal electric unificat (i_c = 2 ...10 mA sau i_c = 4 ... 20 mA), iar mărimea de ieşire este debitul de fluid.

Robinetul de reglare se compune din două subansamble: servomotorul S şi organul de reglare OR. În figura 5.1 este prezentată schema pricipială a unui RR cu acţionare pneumatică.

La variaţia într-un sens a presiunii de comandă p_c, se modifică cursa tijei h până când membrana elastică este în echilibru de forţe (forţa elastică a resortului este egală cu forţa exercitată de aerul comprimat) şi în consecinţă se modifică aria secţiunii de trecere obturator – scaun şi în final se modifică debitul de fluid.

Legendă:
S – servomotor pneumatic;
OR – organ de reglare;
1 – resort;
2 – membrană rigidizată;
3 – tijă;
4 – sistem de etanşare;
5 – obturator;
6 – scaun;
7 – corp robinet.

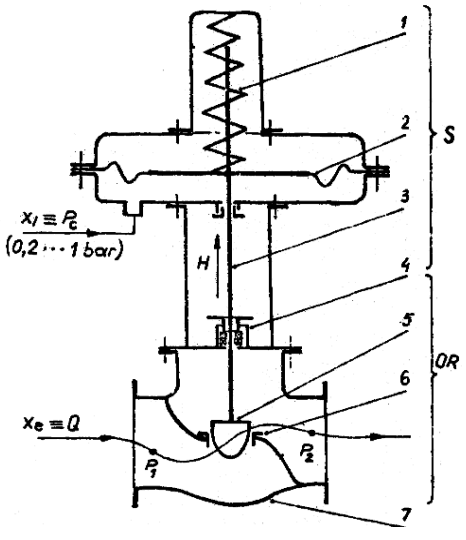


Fig. 5.1. Schema principială a unui robinet de reglare normal închis.

Organul de reglare poate fi acţionat cu servomotor pneumatic, hidraulic sau electric.

Servomotorul pneumatic cu membrană se utilizează la curse relativ mici (10...80 mm), iar cel cu piston la curse mari (50...500 mm). Acestea se caracterizează prin simplitate constructivă şi funcţională, robusteţe şi siguranţă în exploatare, funcţionare fără interdicţii în medii explozive.

5.3. Caracteristicile statice ale RR

Caracteristica statică a unui sistem reprezintă dependenţa în regim staţionar dintre mărimea de ieşire şi mărimile de intrare.

Unui RR i se asociază următoarele caracteristici statice:

- caracteristica statică a servomotorului S, h = f(u), în care u = p_c sau u = i_c;
- caracteristica statică intrinsecă a organului de reglare OR, K_v = f(h);
- caracteristici statice de lucru ale OR, Q = f(h).

Caracteristica statica intrinsecă a OR

Aceasta este o caracteristică hidraulică proprie a OR, care depinde numai de aria şi forma secţiunii de trecere a OR.

Dacă se tratează OR ca o rezistenţă hidraulică, debitul de fluid care trece prin acesta în regim de curgere turbulent, este

$$Q = \alpha \varepsilon A_r \sqrt{\frac{2\Delta p_r}{\rho}} \qquad [m^3/s] \qquad (5.1)$$

în care:

Δp_r - căderea de presiune pe OR;
ρ - densitatea fluidului;
A_r - aria secţiunii de trecere a OR;
α - coeficient de debit;
ε - coeficient de expansiune (detentă).

Pentru simplificarea relaţiei (5.1) se introduce noţiunea de debit specific, ca fiind egal cu debitul de apă cu densitatea $\rho = 1 Kg/dm^3$ care la trecerea prin OR produce o cădere de presiune remanentă $\Delta p_r = 1\ bar$. Expresia acestuia pentru condiţiile etalon precizate, este

$$K_v = \alpha \varepsilon A_r \sqrt{\frac{2\Delta p_r}{\rho}} \qquad (5.2)$$

Dependenţa $K_v = f(h)$ reprezintă caracteristica intrinsecă a OR. În tehnica reglării s-au impus, prin profilarea corespunzătoare a ventilului, următoarele caracteristici intrinseci: logaritmică, liniară şi cu deschidere rapidă (figura 5.2). Reprezentarea grafică

este $K_v / K_{vs} = f(h/h_{100})$, obţinute prin raportarea valorilor curente la valorile lor maxime corespunzătoare stării complet deschis a OR. Debitul specific de scăpări K_{v0}, pentru poziţia închis a OR, arată că RR nu asigură o etanşare perfectă.

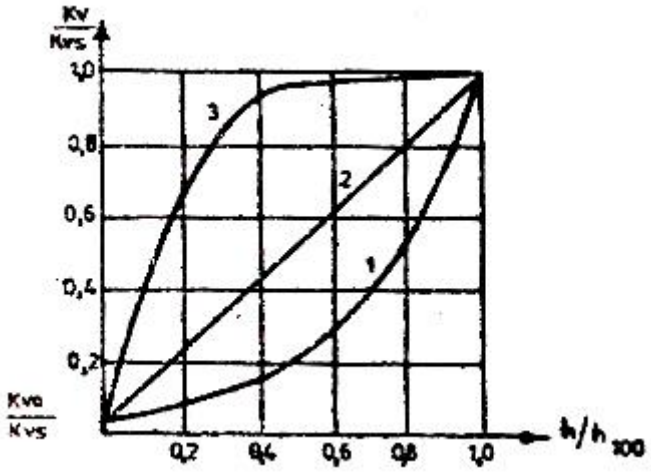


Fig.5.2. Tipuri de caracteristici intrinseci: 1- logaritmică; 2 – liniară; 3 – cu deschidere rapidă.

Caracteristici statice de lucru ale OR

Dependenţa $Q = f(h)$ în regim staţionar, reprezintă caracteristica statică de lucru a OR. Aceasta depinde atât de timpul şi mărimea OR, cât şi de sistemul hidraulic în care este montat acesta.

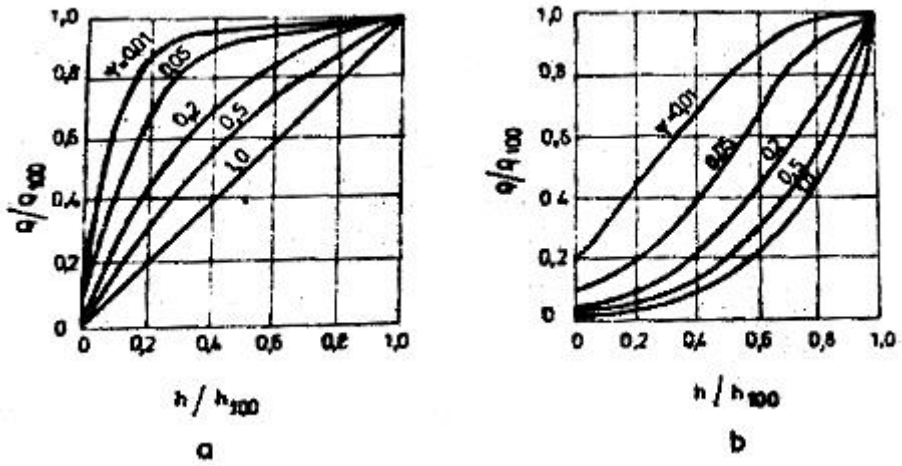


Fig.5.3. Caracteristici de lucru ale RR : a – RR cu caracteristică intrinsecă liniară; b – RR cu caracteristică intrinsecă logaritmică

În figura 5.3 se prezintă caracteristicile statice de lucru ale celor două tipuri de RR pentru diverse valori ale parametrului $\psi = \frac{\Delta p_{r100}}{\Delta p_{s0}}$. Se observă că pentru $\psi = 1 (\Delta p_{s0} = \Delta p_{r100})$, caracteristica statică de lucru se confundă cu caracteristica statică intrinsecă.

5.4. Partea experimentală

Determinările experimentale vor fi efectuate cu ajutorul standului prezentat principial în figura 5.4. Acest stand permite măsurarea următoarelor mărimi:

- valoarea comenzii u, în %;
- cursa H a tijei robinetului de reglare;
- debitul Q al apei vehiculate;
- presiunea P_0 la refularea pompei;
- presiunea P_1 din amontele robinetului de reglare;
- presiunea P_2 din avalul robinetului de reglare.

Cursa H poate fi măsurată direct (în mm) sau indirect prin intermediul sistemului de măsurat deplasarea format din traductorul de deplasare XT şi indicatorul de deplasare XI.

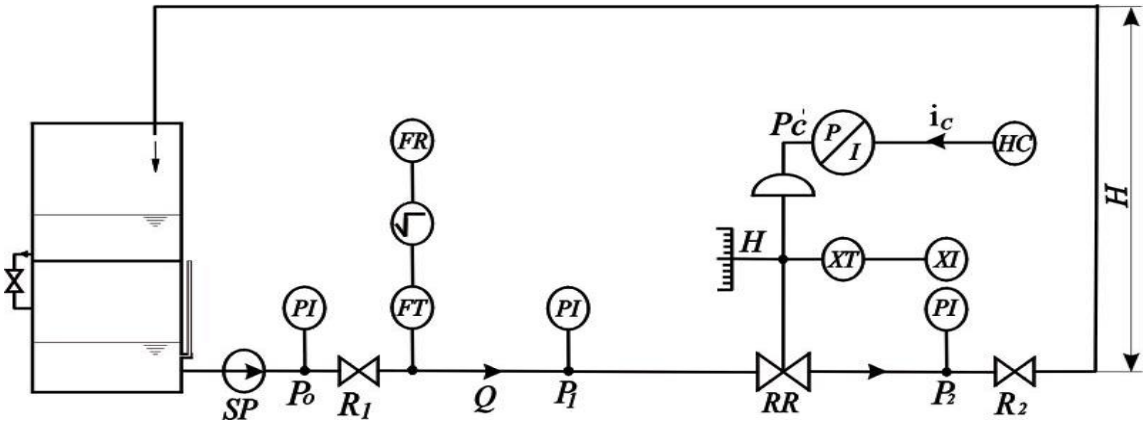


Fig. 5.4. Schema principală a standului pentru determinarea caracteristicilor RR:

SP – *sursa de presiune (pompă centrifugală)*; PI – *manometru*; FT – *trductor de debit*;

FR – înregistrator de debit; HC – element de comandă manuală; XT – traductor de deplasare; XI – indicator de deplasare.

Datele măsurate vor fi trecute în tabelul 5.1.

Tabelul 5.1.

Date experimentale şi calculate

Nr. crt.	Mărimi primare							Mărimi calculate					
	u [%]	H		Q [%]	P_0 [bar]	P_1 [bar]	P_2 [bar]	ΔP_r [bar]	Q [m^3/h]	K_v [m^3/h]	$\frac{H}{H_{100}}$	$\frac{K_v}{K_{v100}}$	$\frac{Q}{Q_{100}}$
		H_d [mm]	H_i $\frac{30}{i_{30}} \cdot i_x$ [mm]										
1	96	30		93	3	1,9	1,5	0,4	10,23	16,17	1	1	1
2	86	29		92	3	1,95	1,45	0,5	10,12	14,31	0,96	0,88	0,98
3	76	28		91	3,05	2	1,4	0,6	10,01	12,92	0,93	0,79	0,97

Modul de lucru

Vor fi efectuate următoarele operaţii:

1. Se va realiza traseul hidraulic pe stand astfel încât să se obţină circuitul din figura 5.4;
2. Cu ajutorul elementului de comandă manuală HC se va deschide complet robinetul de reglare ($H = H_{100}$);
3. Se închide complet robinetul R_1, se porneşte pompa prin acţionarea butonului P (pornire) de pe panoul asociat SRA-D, apoi se deschide complet R_1;
4. Se aduce robinetul R_2 într-o poziţie convenabilă (se recomandă poziţii pentru R_2 care, pentru $H = H_{100}$ să conducă la $P_2 = 1…2$ bar);
5. Având RR cu $H = H_{100}$ (la sfârşitul punctului 4), se face prima serie de citiri pentru tabelul anterior: u, H, Q, P_0, P_1,P_2;
6. Se repetă măsurătorile de la puctul 5 pentru diferitele valori ale comenzii u, respectiv cursei H. în total se vor efectua 7…10 măsurători, relativ uniform distribuite în domeniul $0…H_{100}$ al cursei (distribuite atât pe sensul de închidere cât şi pe cel de deschidere).

Măsurătorile se vor efectua numai în regim staţionar.

Prelucrarea datelor

Relaţii de calcul:

$$\Delta P_r = P_1 - P_2 \quad \text{[bar]}; \qquad Q = \frac{Q[\%]}{100} \cdot Q_{SM100} \quad [m^3/h]; \qquad K_v = \frac{Q}{\sqrt{\frac{\Delta P_r}{\rho}}} = \frac{Q}{\sqrt{\Delta P_r}},$$

unde Q_{SM100} este valoarea maximă a domeniului sistemului de măsurat debitul (Q_{SM100}=11 m^3/h).

În continuare se va ilustra modul de calcul folosind exemplele de măsurători din tabelul 5.1.

Exemplul 1

$$\Delta P_r = P_1 - P_2 = 1{,}9 - 1{,}5 = 0{,}4 \text{ bar};$$
$$Q = \frac{Q[\%]}{100} \cdot Q_{SM100} = \frac{93}{100} \cdot 11 = 10{,}23 \text{ m}^3\text{/h};$$
$$K_v = \frac{Q}{\sqrt{\Delta P_r}} = \frac{10{,}23}{\sqrt{0{,}4}} = 16{,}17 \text{ m}^3\text{/h};$$
$$\frac{H}{H_{100}} = \frac{30}{30} = 1; \quad \frac{K_v}{K_{v100}} = \frac{16{,}17}{16{,}17} = 1; \quad \frac{Q}{Q_{100}} = \frac{10{,}23}{10{,}23} = 1.$$

Exemplul 2

$$\Delta P_r = P_1 - P_2 = 1{,}95 - 1{,}45 = 0{,}5 \text{ bar};$$
$$Q = \frac{Q[\%]}{100} \cdot Q_{SM100} = \frac{92}{100} \cdot 11 = 10{,}12 \text{ m}^3\text{/h};$$
$$K_v = \frac{Q}{\sqrt{\Delta P_r}} = \frac{10{,}12}{\sqrt{0{,}5}} = 14{,}31 \text{ m}^3\text{/h};$$
$$\frac{H}{H_{100}} = \frac{29}{30} = 0{,}96; \quad \frac{K_v}{K_{v100}} = \frac{14{,}31}{16{,}17} = 0{,}88; \quad \frac{Q}{Q_{100}} = \frac{10{,}12}{10{,}23} = 0{,}98.$$

Exemplul 3

$$\Delta P_r = P_1 - P_2 = 2 - 1{,}4 = 0{,}6 \text{ bar};$$
$$Q = \frac{Q[\%]}{100} \cdot Q_{SM100} = \frac{91}{100} \cdot 11 = 10{,}01 \text{ m}^3\text{/h};$$
$$K_v = \frac{Q}{\sqrt{\Delta P_r}} = \frac{10{,}01}{\sqrt{0{,}6}} = 12{,}92 \text{ m}^3\text{/h};$$
$$\frac{H}{H_{100}} = \frac{28}{30} = 0{,}93; \quad \frac{K_v}{K_{v100}} = \frac{12{,}92}{16{,}17} = 0{,}79; \quad \frac{Q}{Q_{100}} = \frac{10{,}01}{10{,}23} = 0{,}97.$$

Cu rezultatele obţinute vor fi construite:

- caracteristica statică intrinsecă $\frac{K_v}{K_{v100}} = f\left(\frac{H}{H_{100}}\right)$;
- caracteristica statică de lucru $\frac{Q}{Q_{100}} = f\left(\frac{H}{H_{100}}\right)$;

(CSI şi CSL vor fi construite pe acelaşi grafic).

În continuare se determină valoarea parametrului Ψ pentru CSL:

$$\Psi = \frac{\Delta P_{r100}}{\Delta P_s} \tag{5.6}$$

unde :

ΔP_{r100} este ΔP_r pentru deschiderea maximă a RR (prima linie din tabelul 5.1);

$\Delta P_s = P_{00} - (P_v + \rho g H)$,

P_{00} – valoarea lui P_0 pentru Q = 0.

Pentru CLS, în cazul sistemelor hidraulice fără ramificaţie este cunoscută relaţia

$$\frac{Q}{Q_{100}} = \frac{1}{\sqrt{1 + \Psi\left(\frac{1}{k_v^2} - 1\right)}} \tag{5.7}$$

în care $k_v = \frac{K_v}{K_{v100}}$.

Se observă că în cazul în care $\Psi = 1$ rezultă $\frac{Q}{Q_{100}} = k_v \Rightarrow$ CLS se confundă cu CSI.

5.5. Întrebări. Exerciţii. Probleme

1. Utilizând relaţia (5.6) se vor determina valorile debitului de scăpări relativ $\frac{Q}{Q_{100}}$ şi absolut pentru cele trei caracteristici, respectiv pentru cele trei valori ale lui Ψ.
2. Ce semnificaţie are K_v şi în ce unităţi de măsură se exprimă acesta?
3. Ce se înţelege prin CSI? Câte tipuri principale de CSI cunoaşteţi?
4. Care este diferenţa dintre CSI şi CSL?

LUCRAREA 6

SISTEME DE REGLARE AUTOMATĂ

6.1. Obiectivele lucrării

În cadrul acestei lucrări de laborator, căreia îi sunt rezervate două ore, se urmăreşte atingerea de către studenţi a următoarelor obiective:

- înţelegerea reglării după abatere;
- studiul experimental al unui sistem de reglare automată după abatere;
- înţelegerea reglării după perturbaţie.

6.2. Prezentarea conţinutului lucrării

Pe parcursul lucrării de faţă se va studia un sistem de reglare automată după abatere a nivelului (SRA-N) şi un simulator pentru un sistem de reglare automată după perturbaţie.

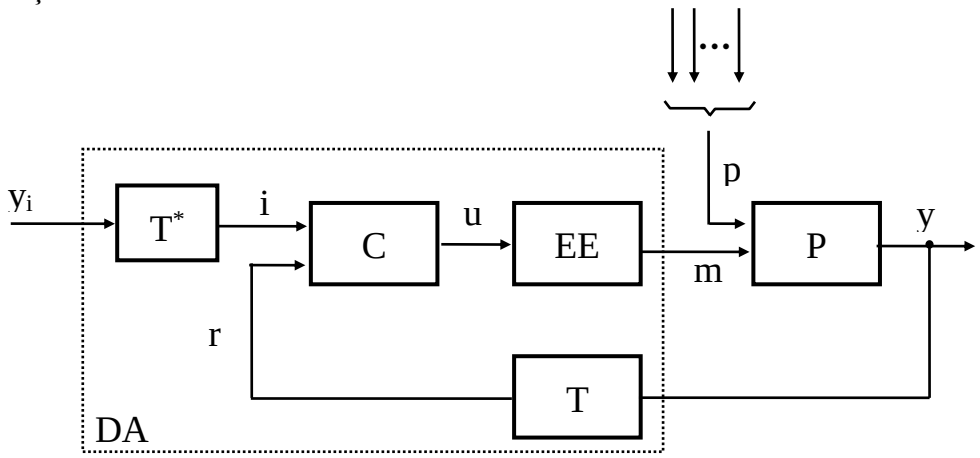


Fig. 6.1. Schema bloc a unui SRA: P- proces; DA - dispozitiv de automatizare; T - traductor; T^*- traductor de intrare; EE - element de executie; y - marime reglata; p - perturbatii; m - marime de executie; r - reactie; i - referinta in semnal; y_i - referinta in unitati ale marimii reglate; u - comanda.

Orice sistem de reglare automată (SRA) este constituit din două parţi principale:

procesul automatizat P şi **dispozitivul de automatizare DA**. SRA care acţionează în baza *legii reglării după abatere* elaborează comanda pe baza prelucrării diferenţei (abaterii) care apare între starea curentă şi o stare de referinţă. După cum se observă din fig. 6.1. în structura DA intră *traductorul, regulatorul şi elementul de execuţie,* care îndeplinesc cele trei funcţii de bază ale oricărui SRA şi anume: *informarea, elaborarea comenzii şi execuţia comenzii.*

În cadrul lucrării de faţă va fi utilizat un stand experimental pentru SRA-N, stand care este prezentat în figura 6.2.

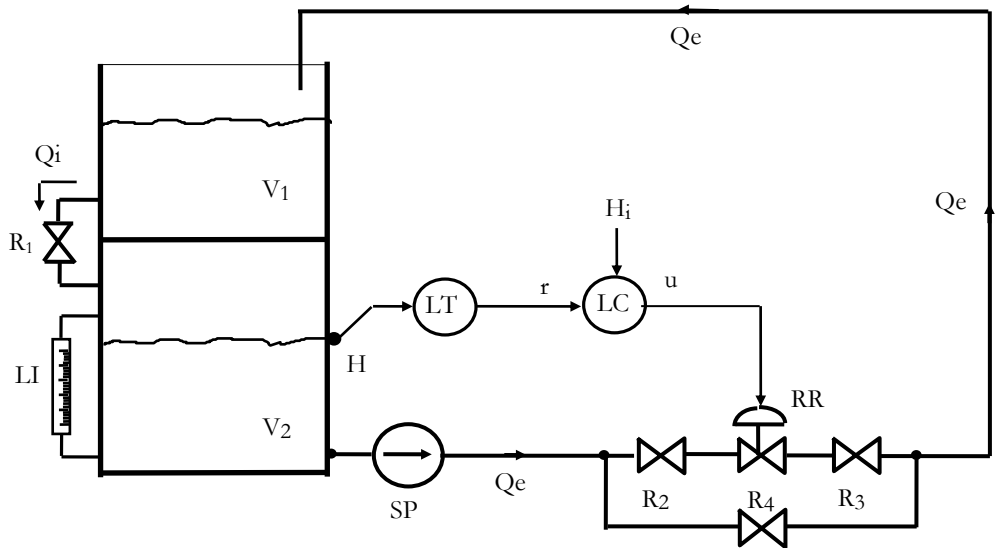


Fig. 6.2. Schema principială a SRA-N existent în laborator:

V_1 ,V_2 - vase cu lichid (suprapuse); R_1…R_4 - robinete de izolare; LI - indicator de nivel (sticla de nivel).

În continuare vor fi discutate toate elementele ce compun SRA-N, în baza cunoştinţelor acumulate la lucrările de laborator anterioare, precum şi a noţiunilor cunoscute la curs.

Robinetul de reglare (RR) este de tip normal închis (NI) şi are rolul de a mări sau micşora debitul Qe în funcţie de modificările comenzii u, primită de la LC.

Observaţie. RR normal închis înseamnă că robinetul de reglare este închis atunci când comanda este minimă.

Regulatorul de nivel (LC) este de tip PID. Scrieţi algoritmul PID care stă la baza funcţionării acestui tip de regulator (a se vedea lucrarea 4). Care este rolul LC în cadrul SRA după abatere?

Traductorul de nivel (LT) este un traductor de tip presiune diferenţială. Spuneţi care este rolul LT (a se vedea lucrarea 3)!

Să se descrie sintetic funcţionarea SRA-N pentru cazul în care Hi=ct. şi Qi creşte cu 2 m^3/h.

Reglarea după perturbaţie va fi discutată cu ajutorul sistemului din fig. 6.3.

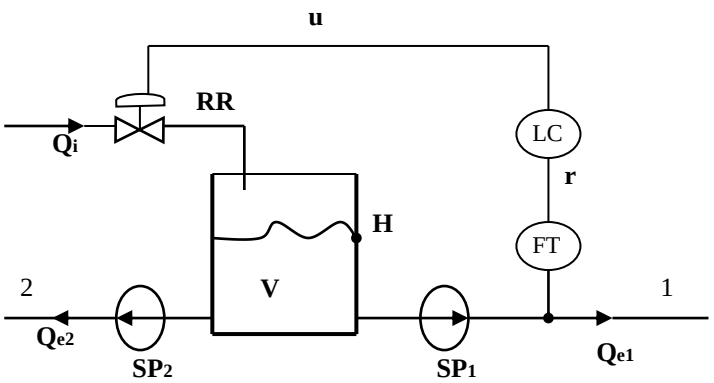


Fig.6.3. Exemplu de sistem de reglare după perturbaţie: V – vas cilindric; FT – traductor de debit; LC – regulator de nivel; RR – robinet de reglare; SP_1, SP_2 – surse de presiune (pompe).

Alimentarea vasului are debitul Q_i. Lichidul din vas este extras prin două conducte, cu debitele Q_{el}, respectiv Q_{e2}. Prin conductele 1 şi 2 curgerea este *forţată*, sub presiunea creată de pompele SP_1 şi SP_2.

Punerea problemei de reglare:

- *obiectivul* sistemului constă din menţinerea constantă a nivelului H, la o anumită valoare, notată în continuare cu H_i;
- debitele Q_{el} şi Q_{e2} au, din punctul de vedere al vasului, variaţii aleatoare în timp (acestea sunt *perturbaţii*);
- debitul Q_i poate fi utilizat drept *comandă,* ceea ce înseamnă că acesta va fi modificat astfel încât să se menţină nivelul H la valoarea H_i ($H = H_i$).

Sistemul de reglare din fig. 6.3 va fi investigat, în cadrul lucrării de laborator, cu ajutorul unui *simulator*.

6.3. Partea experimentală

În laborator, standul experimental care permite studiul SRA-N este reprezentat în fig. 6.2. În continuare vor fi parcurse următoarele etape:

1. Recunoaşterea fizică a elementelor ce compun SRA-N;
2. Cu ajutorul tehnicianului sau a cadrului didactic va fi pus SRA-N în funcţiune;
3. Cu SRA-N funcţionând în modul automat, se va urmări şi consemna în referat modul în care funcţionează SRA-N la modificări ale valorii prescrise H_i şi a debitului de intrare în vas, Q_i. Pentru aceasta vor fi parcurse etapele:
   a. se aduce prescrierea regulatorului la 50% (H_i=50%) şi se aşteaptă până când SRA-N ajunge în regim staţionar (mărimea reglată H a ajuns egală cu prescrisă H_i);
   b. se modifică H_i cu 10% astfel încât H_i=60% şi se urmăreşte modul în care se modifică mărimea reglată H;
   c. se aduce, din nou, prescrierea regulatorului la 50% (H_i=50%) şi se aşteaptă până când SRA-N ajunge în regim staţionar;
   d. se modifică debitul Q_i acţionând robinetul R_1 (se deschide sau se închide cu o tură) şi se urmăreşte modul în care se modifică mărimea reglată H.
7. Se vor face determinări ale formei şi duratei regimului trazitoriu (dinamic) pentru anumite variaţii date prescrierii H_i şi perturbaţiei Q_i.

În acest scop se vor întocmi grafice de forma celor din figura 6.4, unde se va pune în evidenţă aspectul calitativ al răspunsului SRA şi durata regimului tranzitoriu.

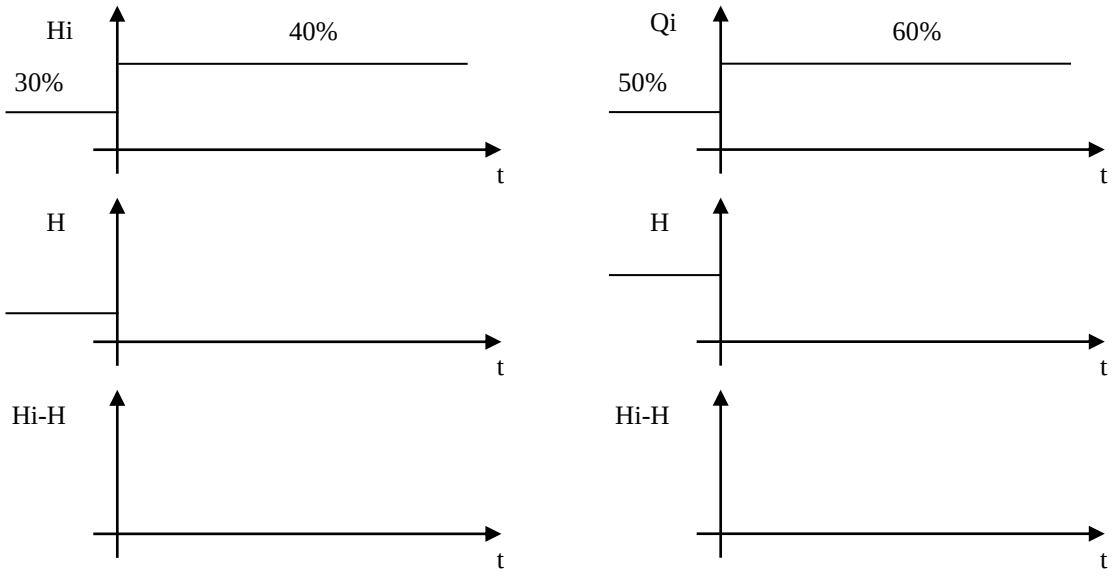


Fig. 6.4. Grafice pentru analiza calitativă a răspunsului în timp pentru SRA-N.

Pentru SRA dupa perturbatie se va utiliza simulatorul existent in directorul *Simulatoare*. După activarea simulatorului, fiecare student va analiza cu atenţie ceea ce oferă acesta cu ajutorul butoanelor, graficelor şi altor elemente afişate pe ecran.

Se vor urmări:

- reglarea automată a nivelului;
- pertubarea sistemului prin debitele Q_{e1} sau Q_{e2} (precizare: debitul Q_{e2} va fi modificat sub forme de treaptă, pozitivă sau negativă, pe durate de timp limitate);
- reglarea manuală a nivelului (dacă această posibilitate este implementată în simulator).

Se recomandă studenţilor să analizeze verosimilitatea fizică a evoluţiei în timp a mărimilor aferente SRA-N, afişate pe ecranul monitorului.

6.4. Intrebări şi exerciţii

1. Care este dezavantajul SRA-N după abatere? Exemplificaţi pe graficele experimentale obţinute.
2. Care este avantajul SRA după abatere?
3. Care sunt domeniile mărimilor de intrare şi de ieşire pentru: LT, LC, EE(RR)?
4. Fie procesul de acumulare a unui gaz într-un vas, ilustrat în figura 6.5.

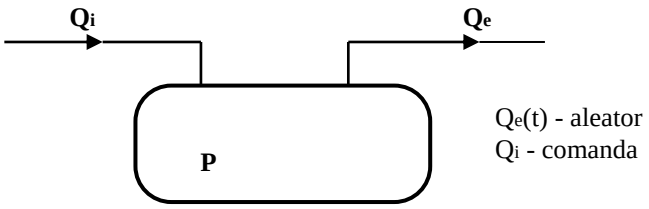


Fig. 6.5. Procesul de acumulare a unui gaz într-un vas.

Să se deseneze structura unui SRA-P după perturbaţie şi apoi a unui SRA-P după abatere.

8. Pentru incinta din fig. 6.6 se cere construirea structurii unui SRA-T după perturbaţie, perturbaţia considerată fiind *temperatura mediului exterior* T_m.

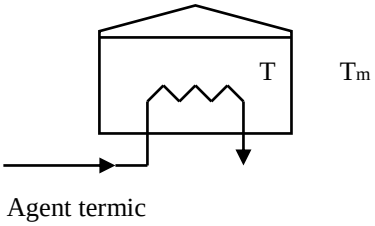


Fig. 6.6. Reglarea temperaturii într-o incintă.

LUCRAREA 7

GESTIUNEA TIMPULUI ŞI GRAFICA ÎN LIMBAJUL C

7.1. Obiectivele lucrării

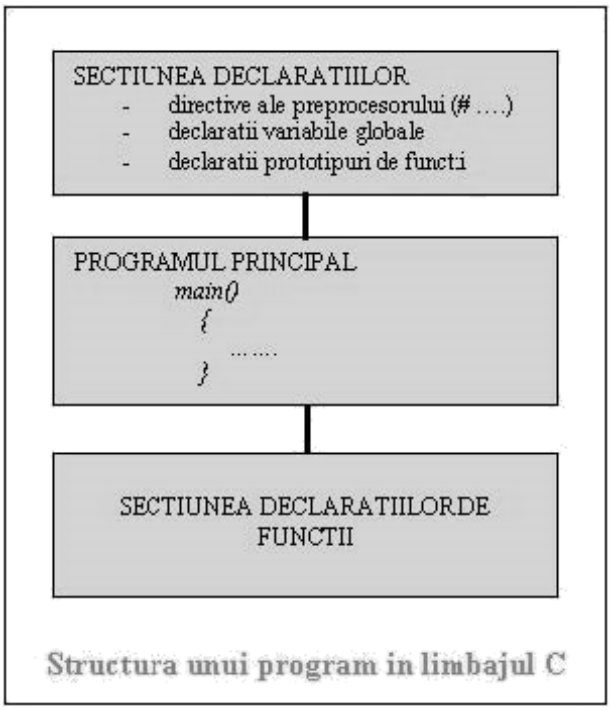
- Însuşirea modului de utilizare a funcţiilor de control C, funcţiilor de control al timpului şi funcţiilor video C in regim alfanumeric si in regim grafic;
- Elaborarea funcţiei pentru trasarea si gradarea unui sistem de axe;
- Elaborarea funcţiei pentru vizualizarea informaţiei in bargraf.

7.2. Prezentarea conţinutului lucrării

Mediul de programare C este unul dintre cele mai populare limbaje de programare. În cadrul acestei lucrări ne propunem să ne familiarizăm cu acest mediu de programare, cu funcţiile de bază utilizate, funcţiile de control, funcţiile de gestionare a timpului şi funcţiile grafice ale limbajului.

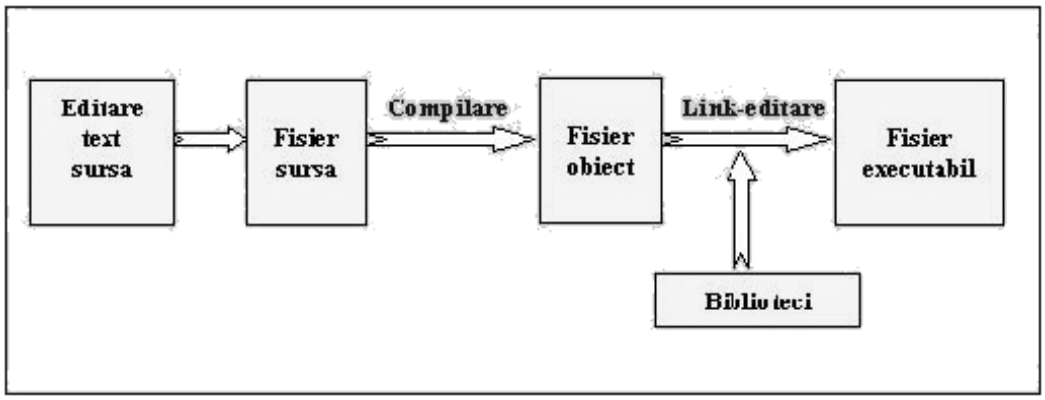
A.Funcţii de intrare-ieşire în limbajul C

Structura generală a unui program C:



Structura unui program in limbajul C

Etape în dezvoltarea unui program simplu în C:



*Directiva preprocesor **#include***

Sintaxa: *#include <nume_fis>*

Efect: Include la compilare fişierul cu numele nume_fis (ca şi cum s-ar "copia" textul acestuia în programul ce conţine directiva #include). În general fişierele ce sunt incluse conţin o serie de definiţii şi declaraţii absolut obligatorii, ca de exemplu:

- *stdio.h* - funcţii standard de I/O
- *conio.h* - extensie a funcţiilor de I/O la consolă
- *stdlib.h* - funcţii standard ale bibliotecilor C;
- *graphics.h* - funcţii de lucru în regim grafic;
- *math.h* - functii matematice

Tipuri de variabile des utilizate

a) variabile de tip caracter (**char**)

b) variabile de tip întreg (**int**)

- -short
- -long
- -unsigned

c) variabile de tip real

- -float: simplă precizie
- -double: dublă precizie

Functia printf()

Efect: afişează un şir formatat pe ecran, în mod text.

Sintaxa: *printf(<sir_formatat>,var1,var2,...);*

<sir_formatat> este un şir de caractere încadrat între ghilimele, care conţine textul ce va fi afişat pe ecran şi specificatorii de format pentru afişarea variabilelor var1,var2,... Fiecăreia din variabilele var1,var2,... trebuie sa îi corespundă un specificator de format:

- %u - întreg fără semn
- %d – întreg
- %ld - întreg lung
- %p – pointer
- %f – real
- %e - real în format exponenţial
- %c – caracter
- %s - şir de caractere
- %x - întreg în format hexazecimal

Caractere "escape" conţinute în *sir formatat*

\n - linie nouă

\t - TAB

\a - BELL

\b - BACKSPACE

\\ - \

Funcţia scanf()

Efect: citeşte de la tastatură valorile variabilelor conţinute în listă.
Sintaxa: *scanf(<şir_formatat>, &var1, &var2,...);*
<sir_formatat> conţine câte un specificator de format pentru fiecare din variabilele var1,var2,... Caracterul & indică faptul că argumentele funcţiei scanf nu sunt variabilele, ci adresele lor.

*Functia **gets()***

Efect: citeşte de la tastatură un şir de caractere
Sintaxa: *gets(şir_de_caractere);*
şir_de_caractere este numele unei variabile vector de tip char. Folosirea lui gets() este preferabilă în locul lui scanf() (spre deosebire de scanf() utilizat pentru a citi şiruri de caractere, gets() recunoaşte şi spaţiile, pe care le stocheaza în *şir_de_caractere*).

Functia clrscr()

Efect: are ca efect ştergerea ecranului
Sintaxa: *clrscr();*

Functia gotoxy()

Efect: poziţionează cursorul-text în linia şi coloana specificate
Sintaxa: *gotoxy(nr_coloană, nr_linie);*

unde:

- nr_coloana= 0 ... 79
- nr_linie= 0 ... 24

Functia cprintf()

Efect: Această funcţie este similară lui printf(), însă afişarea se face începând din poziţia curentă a cursorului-text.

B. Funcţii de control în limbajul C

*Instrucţiunea condiţionată **if ... else** (selecţia simplă)*

Sintaxa:

```
if (expresie_c)
   {
        instrucţiune 1;
        instrucţiune 2;
   }
else
   {
        instrucţiune 1';
        instrucţiune 2';
   }
```

Efect: În funcţie de valoarea de adevăr a expresiei expresie_c se executa unul din seturile de instrucţiuni instrucţiune 1, instrucţiune 2, sau instrucţiune 1', instrucţiune 2' ... . Daca expresie_c este adevărată se execută setul aferent lui if; dacă expresie_c este falsă se execută setul aferent lui else.

*Instrucţiunea condiţionată **switch ... case** (selecţia multiplă)*

Sintaxa:

```
switch(variabilă)
  {
    case constantă_1:
          instrucţiuni1;
          break;
    case constantă_2:
          instrucţiuni2;
          break;
    .
    .
    .
    default:
          instrucţiuni
  }
```

Efect: Este testată valoarea variabilei variabilă şi atunci când ea egalează valoarea unei constante a unui case se vor executa instrucţiunile aferente acelui case; daca variabilă nu egalează nici o constantă a unui case, se execută instrucţiunile aferente lui default (in cazul în care exista un bloc default, prezenta sa nefiind obligatorie).

Variabila de test şi constantele nu pot fi decât de tip int sau char.

Prezenţa lui break la un grup case este necesară atunci când se doreşte numai execuţia instrucţiunilor aferente acelui case. Dacă break lipseşte şi variabilă a egalat o constantă, se vor executa nu numai instrucţiunile aferente acelui case, ci toate instrucţiunile celorlalte case-uri care îl succed (inclusiv default).

*Ciclul **for***

Sintaxa:

```
for(iniţializare; test_condiţie; increment)
  {
   instrucţiuni;
  }
```

Efect: se execută ciclic o secvenţă de instrucţiuni, în funcţie de o condiţie dată.

Secţiunea iniţializare este folosită pentru a da o valoare iniţială variabilei care controlează ciclul, fiind executată o singură dată, înainte ca ciclul să înceapă.

Secţiunea test_condiţie testează variabila de control cu valoarea scop ori de câte ori ciclul se repetă. Daca testul are valoarea de adevăr fals, atunci execuţia ciclului este întreruptă. Acest test este făcut la începutul fiecărui ciclu, la fiecare repetiţie.

Secţiunea increment realizează modificarea variabilei de control a ciclului cu o anumita cantitate; ea este executată la sfârşitul fiecărui ciclu.

Oricare din cele trei secţiuni poate lipsi; de asemenea, testul de condiţie se poate referi la orice altă variabilă (nu numai la variabila de control a ciclului).

*Ciclul **while***

Sintaxa:

```
while(expresie_c)
 {
  instrucţiuni;
 }
```

Efect: se execută ciclic secvenţa de instrucţiuni aferentă, atât timp cât expresia condiţională expresie_c este adevărată. Testul condiţional este efectuat la începutul fiecărui ciclu, în consecinţă, daca expresie_c este falsă de la început ciclul nu se va executa.

*Ciclul **do***

Sintaxa:

```
do
 {
  instrucţiuni;
 }
  while(expresie_c);
```

Efect: se executa ciclic secvenţa de instrucţiuni aferentă atât timp cât expresia condiţională expresie_c este adevărată. Testul condiţional este efectuat la sfârşitul fiecărui ciclu, în consecinţă, ciclul va fi executat cel puţin o dată.

*Instrucţiunile **break** şi **continue***

Au rolul de a controla forţat execuţia ciclurilor for, while şi do. Instrucţiunea **break** are ca efect ieşirea forţată din ciclu. Instrucţiunea **continue** forţează ca următoarea parcurgere a ciclului să aibă loc trecând peste instrucţiunile dintre ea şi ultima instrucţiune a ciclului inclusiv (acestea sunt "sărite").

Funcţia getch()

Sintaxa:ch=getch(); sau getch()
unde:
ch=variabilă de tip char

Efect: Aşteaptă apăsarea unei taste, după care întoarce codul acesteia variabilei ch. Dacă apelul lui getch() se face fără atribuirea valorii întoarse unei variabile, efectul este de oprire a programului până la apăsarea unei taste. Tasta apăsată nu apare pe ecran.

Funcţia getche()

Efect: Este similară cu getch(), singura deosebire fiind aceea că tasta apăsată este afişată (are "ecou") pe ecran.

Funcţia kbhit()

Efect: Detectează dacă a fost apăsată sau nu o tastă. Dacă a fost apăsată o tastă, kbhit() întoarce valoarea "adevărat", în caz contrar întoarce valoarea "fals".

Funcţia delay()

Sintaxa: delay(timp_ms)

Efect: Temporizează execuţia programului cu timp_ms milisecunde.

Observaţie:
Aflarea restului împărţirii numărului întreg a la numărul întreg b:
rest=a%b;

C. Funcţii de gestionare a timpului

Prototipurile funcţiilor de gestionare a timpului-sistem se gasesc în header-ele dos.h şi time.h.

Precizările următoare se referă la funcţiile cu prototipul în header-ul dos.h.

În dos.h se găsesc prototipurile următoarelor funcţii ce permit gestionarea resursei timp:

*void getdate(struct date *datep)*
*void gettime(struct time *timep)*
*void setdate(struct date *datep)*
*void settime(struct time *timep)*
void sleep(unsigned seconds)
void delay(unsigned milliseconds)

în care ***datep** şi ***timep** sunt pointeri la structurile **struct date** şi **struct time**;
seconds, milliseconds sunt numere întregi pozitive.

Structurile aferente datei şi orei sunt de forma:

```
struct date
 {
  int  da_year;        /* pentru an */
  char da_day:       /* pentru  zi */
  char da_ mon;     /* pentru lună */
 };
```

respectiv:

```
struct time
 {
  unsigned char ti_min;     /* pentru minute */
  unsigned char ti_hour;    /* pentru ore */
  unsigned char ti_hund;    /* pentru sec/100 */
  unsigned char ti-sec;       /* pentru secunde */
 };
```

Funcţiile **getdate** şi **gettime** preiau data şi ora curente iar funcţiile **setdate** şi **settime** permit iniţializarea datei şi orei.

Funcţiile **delay** şi **sleep** permit întârzierea execuţiei programului cu numărul specificat de milisecunde, respectiv secunde.

Modurile de utilizare a funcţiilor **getdate**, **gettime** şi **delay** sunt evidenţiate în programele P1 şi P2 ale căror texte sursă se prezintă în continuare.

```
/*
        Program P1 – afisare data si ceas cu getdate() si gettime()
*/
#include <dos.h>
#include <stdlib.h>

struct date d;  /* structura date */
struct time t;   /* structura time */
int i,j;
char c;

void main(void)
{
 clrscr();
 gotoxy(30,5);
 cprintf("E - iesire din program");

 /*  bucla infinita */
 for(;;)
 {
        getdate(&d);   /* se preia data */
        gettime(&t);    /* se preia ora */
        i=t.ti_sec;       /* secunda curenta */
         if(j!=i)          /* sezizare schimbare secunda */
        {
                gotoxy(58,3);     /* tiparire in linia 3 coloana 58 */
                cprintf("%02d-%02d-%4d", d.da_day, d.da_mon, d.da_year);
                cprintf("  %02d:%02d:%02d", t.ti_hour, t.ti_min, t.ti_sec);
                j=i;      /* secunda anterioara */
        }
        if(kbhit()) c=getch();  /* citesc tasta apasata */
        if((c=='e') || (c=='E')) exit(1);
  }       /* sfirsit for */

}       /* sfirsit main */



/*
        Program P2 – varianta a P1, care afiseaza si sutimile de secunda
*/
#include <dos.h>
#include<stdlib.h>

struct date d;
struct time t;
```

```
void main(void)
{
 gotoxy(30,5);
 cprintf("E - iesire din program");
 clrscr();
/* bucla infinita */
 for(;;){
        delay(10);                          /* asteapta 10 ms */
        getdate(&d);                        /* se preia data  */
        gettime(&t);                        /* se preia ora   */
        gotoxy(55,3);                       /* scrie in coloana 55, linia 3 */
        cprintf("%02d-%02d-%4d", d.da_day, d.da_mon, d.da_year);
        cprintf("  %02d:%02d:%02d:%02d", t.ti_hour, t.ti_min, t.ti_sec,t.ti_hund);
         if(kbhit()) c=getch();              /*  test apasare tasta */
        if((c=='e' )|| (c=='E')) exit(1);
  }     /* sfirsit for */
}       /* sfirsit main */
```

Programul P1 afişează data şi ora cu periodicitate de o secundă, iar P2 cu o periodicitate de 10 ms (afişând în plus faţă de P1 sutimile de secundă).

Ieşirea din cele două programe se face prin apăsarea tastei E (funcţia kbhit()).

D. Funcţii video ale limbajului C

Adaptorul video poate opera în două moduri de bază:

- **modul text** (alfanumeric), de regulă implicit;
- **modul grafic**.

Modul text

În modul text unitatea minimă adresabilă este caracterul reprezentat de o matrice de puncte (pixeli).

Stabilirea modului text se poate face cu funcţia

void textmode(int mod);

unde pentru situaţia 80 coloane şi 25 linii *mod=3*, iar numerotarea se face conform fig.7.1.

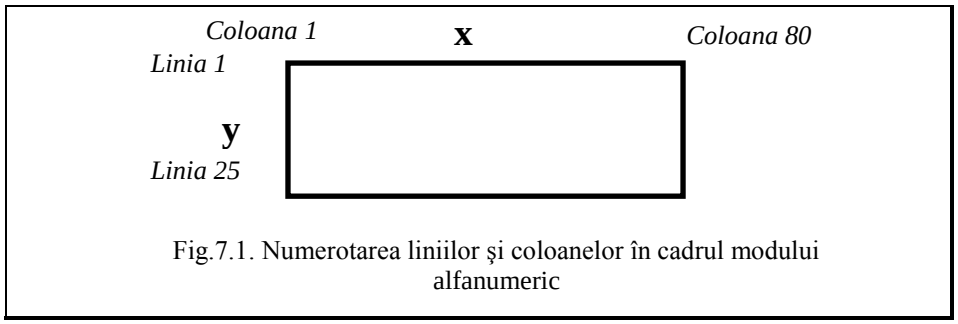


Fig.7.1. Numerotarea liniilor şi coloanelor în cadrul modului alfanumeric

Prototipurile funcţiilor specifice modului text se găsesc în header-ul conio.h ,al cărui conţinut este prezentat în continuare.

CONIO.H

Functions

cgets	clreol	clrscr	cprintf
cputs	cscanf	delline	getch
getche	getpass	gettext	gettextinfo
gotoxy	highvideo	insline	inp
inport	inportb	inpw	kbhit
lowvideo	movetext	normvideo	outp
outport	outportb	outpw	putch
puttext	_setcursortype	textattr	textbackground
textcolor	textmode	ungetch	wherex
wherey	window		

Constants, data types, and global variables

BLINK	COLORS	directvideo	_NOCURSOR
_NORMALCURSOR	_SOLIDCURSOR	text_info	text_modes
_wscroll			

Definirea şi ştergerea unei ferestre

O fereastră este o zonă dreptunghiulară de pe ecran care poate fi gestionată în mod independent şi care se creează cu ajutorul funcţiei **window** cu prototipul:

void window(int stânga, int sus, int dreapta, int jos);

La un moment dat este activă fereastra definită la ultimul apel al funcţiei **window**.

Fereastra activă se şterge cu funcţia **clrscr** cu prototip *void clrscr(void)*;

OBSERVAŢIE: după execuţia funcţiei **textmode** fereastra implicită este reprezentată de tot ecranul.

Gestiunea cursorului

Cursorul se poate plasa pe un caracter al ferestrei active utilizând funcţia gotoxy cu prototipul

void gotoxy(int x,int y);

unde x şi y reprezintă numărul coloanei, respectiv al liniei în fereastra activă.

Poziţia curentă a cursorului in fereastra activă se poate afla cu funcţiile:

int wherex(void);
int wherey(void);

Setarea culorilor

În tabelul 7.1 se prezintă codurile culorilor din paleta principală.

Tabelul 7.1

Codurile culorilor

Culoare	Functie C	Cod
negru	BLACK	0
albastru	BLUE	1
verde	GREEN	2
turcoaz	CYAN	3
roşu	RED	4
purpuriu	MAGENTA	5
maron	BROWN	6
gri deschis	LIGHTGRAY	7
gri închis	DARKGRAY	8
albastru deschis	LIGHTBLUE	9
verde deschis	LIGHTGREEN	10
turcoaz deschis	LIGHTCYAN	11
roşu deschis	LIGHTRED	12
purpuriu deschis	LIGHTMAGENTA	13
galben	YELLOW	14
alb	WHITE	15
clipire	BLINK	128

Pentru setarea culorilor se folosesc funcţiile:

- pentru fond void textbackground(int culoare);
- pentru caractere void textcolor(int culoare);
- pentru atribut void textattr (int atribut);

unde: **culoare** este un întreg în intervalul **[0,7]** *pentru fond* si **[0,15]** *pentru text*;

atribut =16 ∗ cul_fond | cul_text | clipire.

Gestiunea textelor

Pentru afişarea caracterelor color în conformitate cu atributele definite prin relaţia de mai sus se pot folosi funcţiile:

- ***putch*** - afişează un caracter;
- ***cputs*** - afişează un şir (analog cu *puts*);
- ***cprintf*** - afişează date sub controlul formatelor de conversie (analog cu *printf*).

În continuare se prezintă un program care utilizează o parte din funcţiile enumerate.

```
/*
        Program GRAF1.C – functii video in mod text
*/

  #include <dos.h>
  #include <conio.h>

  void sunet(int f1)
   {
        sound(f1),delay(100),sound(2*f1),delay(200),
        sound(f1),delay(100),nosound();
   }

  void main(void)
   {
        clrscr();
        textbackground(BLUE),clrscr();
        window(24,7,60,17);
        textbackground(BLACK),clrscr();
        window(22,6,58,16);
        textattr(RED*16|WHITE|BLINK),clrscr();
        gotoxy(6,2),cprintf("SISTEME CU MICROPROCESOARE");
        textattr(GREEN*16|YELLOW);
        gotoxy(7,5),cprintf("ECHIPAMENTE DE CONDUCERE");
        textattr(BLUE*16|RED);
        gotoxy(14,8),cprintf(“Pentru iesire”);
        gotoxy(10,9),cprintf(“se apasa orice tasta !”);

        for(;;)
        {
                if(kbhit())             /* se iese daca se apasa o tasta */
                                exit(1);
                sunet(550);             /* altfel tiuie potrivit functiei sunet() */
        }
    }
```

Modul grafic

Prototipurile funcţiilor specifice modului grafic se găsesc în header-ul graphics.h, al cărui conţinut este prezentat în continuare.

GRAPHICS.H

Functions

arc	bar	bar3d
circle	cleardevice	clearviewport
closegraph	detectgraph	drawpoly
ellipse	fillellipse	fillpoly
floodfill	getarccoords	getaspectratio
getbkcolor	getcolor	getdefaultpalette
getdrivername	getfillpattern	getfillsettings
getgraphmode	getimage	getlinesettings
getmaxcolor	getmaxmode	getmaxx
getmaxy	getmodename	getmoderange
getpalette	getpalettesize	getpixel
gettextsettings	getviewsettings	getx
gety	graphdefaults	grapherrormsg
_graphfreemem	_graphgetmem	graphresult
imagesize	initgraph	installuserdriver
installuserfont	line	linerel
lineto	moverel	moveto
outtext	outtextxy	pieslice
putimage	putpixel	rectangle
registerbgidriver	registerfarbgidriver	registerbgifont
registerfarbgifont	restorecrtmode	sector
setactivepage	setallpalette	setaspectratio
setbkcolor	setcolor	setfillpattern
setfillstyle	setgraphbufsize	setgraphmode
setlinestyle	setpalette	setrgbpalette
settextjustify	settextstyle	setusercharsize
setviewport	setvisualpage	setwritemode
textheight	textwidth	

Constants, data types, and global variables

arccoordstype	CGA_COLORS	COLORS
EGA_colors	fill_patterns	fillsettingstype
font_names	graphics_drivers	graphics_errors
graphics_modes	HORIZ_DIR	line_styles
line_widths	linesettingstype	MAXCOLORS
line_widths	linesettingstype	MAXCOLORS
palettetype	pointtype	putimage_ops
text_just	text directions	textsettingstype
USER_CHAR_SIZE	VERT_DIR	viewporttype

Setarea modului grafic rezultă din următoarea secvenţa de program:

```
#include <stdio.h>
#include <graphics.h>
#include <stdlib.h>
#include <conio.h>

int gdrv=VGA,gmod=VGAHI,errorcode;      /* adaptor VGA, submod VGAHI
                                           (640 x 480 pixeli, 16 culori) */
void main(void)
{
 initgraph(&gdrv,&gmod,"c:\\bc\\bgi");  /* iniţializare mod grafic */
 errorcode=graphresult();               /* errorcode conţine starea cu
                                           care s-a terminat  funcţia
                                           initgraph si      trebuie  sa fie
                                           grOk pentru o situatie normala */
 if (errorcode != grOk)
 {
        printf("Graphics error: %s\n", grapherrormsg(errorcode));
       exit(1);          /* returneaza codul erorii  */
  }

 outtextxy(100,100,"I S S C");
 getch();
 closegraph();          /* închide modul grafic */
}
```

În cazul utilizării modului grafic sunt utile şi următoarele funcţii de setare:

void restorcrtmode(void); - restabileşte modul text;
void setgraphmode(void); - restabileşte modul grafic;
void closegraph(void); - închide sistemul grafic.
int getgraphmode(void); - întoarce o valoare intre 0 si 5 funcţie de driverul grafic.

În modul grafic ecranul este văzut ca o matrice de **640 * 480** puncte, conform fig. 7.2.

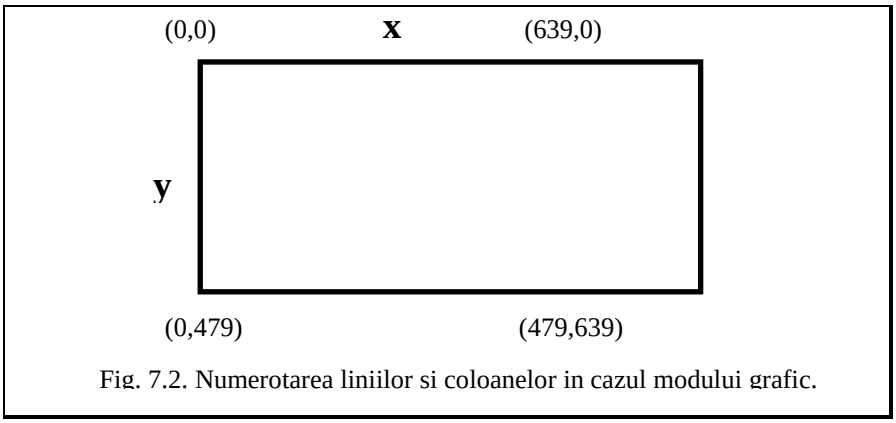


Fig. 7.2. Numerotarea liniilor si coloanelor in cazul modului grafic.

Setarea culorii

Pentru setarea culorii se folosesc funcţiile:

void setbkcolor(int culoare); - setează culoarea fondului;

void setcolor(int culoare); - setează culoarea de desenare;

unde culoare are una din semnificaţiile prezentate in tabelul 7.1.

Desenarea în fereastra activă

În tabelul 7.2 sunt prezentate prototipurile funcţiilor de desenare, iar în tabelele 7.3 şi 7.4 codificările pentru stilurile, respectiv pentru grosimile de linie.

Tabelul 7.2

Funcţii de desenare în cadrul modului grafic

Prototip	Funcţie
void moveto(int x,int y)	fixează poziţia curentă în (x,y)
void putpixel(int x,int y,int culoare);	înscrie pixelul (x,y) cu culoare
void lineto (int x,int y);	linie din poziţia curentă până la (x,y)
void line(int x1,int y1,int x2,int y2);	linie din (x1,y1) în (x2,y2)
void circle(int x,int y,int r);	cerc cu centrul în (x,y) si raza r
void rectangle(int stinga,int sus, int dreapta,int jos)	dreptunghi cu colţurile menţionate
void setlinestyle(int stil_linie,int _model,int grosime);	fixează tipul liniei conform tabelelor de mai jos

Tabelul 7.3

Codificarea stilurilor de linie

Nume	Cod	Descriere
SOLID_LINE	0	linie continuă
DOTTED_LINE	1	linie punctată
CENTER_LINE	2	linie întreruptă (– — –)
DASHED_LINE	3	linie intreruptă (– – –)
USERBIT_LINE	4	stil definit de utilizator

Tabelul 7.4

Codificarea grosimilor de linie

Nume	Cod	Descriere
NORM_WIDTH	1	1 pixel latime
THICK_WIDTH	3	3 pixel latime

În tabelul 7.2 parametrul model are valoarea nenulă pentru tipul USERBIT.

Colorarea şi haşurarea figurilor

Colorarea figurilor închise, rezultate prin utilizarea funcţiilor din tabelul 7.2, se poate realiza cu funcţia

void floodfill(int x, int y, int culoare_contur);

unde: *x, y* - coordonatele unui punct care aparţine interiorului figurii;
culoare_contur - culoarea conturului figurii.

Specificarea modelului şi a culorii de hasurare se fac cu funcţia:

void setfillstyle(int model, int color);

unde: *color* - culoarea conform tabelului 7.1;
model - modelul conform tabelului 7.5.

Ferestre grafice

În modul grafic ferestrele sunt cunoscute sub denumirea de viewport, pentru care în cele ce urmează se prezintă funcţiile uzuale de lucru.

Funcţia

setviewport(int stânga, int sus, int dreapta, int jos, int clip);

creează o fereastra cu coordonatele indicate.

Variabila de tip întreg clip reglementează amplasarea în viewport a unor desene sau texte. Daca clip=0 figura (textul) poate depaşi limitele ferestrei, în caz contrar nu. La referire, colţul stânga-sus are coordonatele (0,0), iar clip se defineşte la începutul programului.

Funcţia

clearviewport();

realizează ştergerea ferestrei curente.

Daca fereastra curentă este reprezentată de tot ecranul, atunci pentru ştergerea acestuia se utilizeaza funcţia:

cleardevice();

Afişarea textului în mod grafic

Reprezentarea caracterelor unui text în modul grafic se poate face prin:

- descrierea imaginii fiecărui caracter prin câte o matrice de 8x8 pixeli (8x8 *bit-mapped font*);
- descrierea printr-un set de vectori (*stroked font*).

Funcţia

void settextjustify(int orizontal, int vertical);

permite alinierea textului referitor la poziţia curentă **(PC)** conform precizărilor din tabelul 7.6. În mod implicit, alinierea orizontală este la stânga *(LEFT_TEXT*) iar cea verticală este la bază (*BOTTOM_TEXT*).

. Tabelul 7,5

Codificarea modelelor de haşurare

Nume	Cod	Descriere
EMPTY_FILL	0	umple cu culoarea de fond
SOLID_FILL	1	umple uniform toţi pixelii
LINE_FILL	2	haşură orizontală
LTSLASH_FILL	3	haşură ///
SLASH_FILL	4	haşură /// linii groase
BKSLASH_FILL	5	haşură \\\ linii groase
LTBKSLASH_FILL	6	haşură \\\
HATCH_FILL	7	haşură în cruce +++
XHATCH_FILL	8	haşură în cruce oblică
INTERLEAVE_FILL	9	haşură cu întreţesere
WIDE_DOT_FILL	10	umple cu puncte rare
CLOSE_DOT_FILL	11	umple cu puncte dese
USER_FILL	12	model utilizator

Tabelul 7.6

Alinierea textului in modul grafic

Descriere	Nume	Valoare	Acţiune
Orizontal	LEFT_TEXT CENTER_TEXT RIGHT_TEXT	0 1 2	PC la stânga PC in centru PC la dreapta
vertical	BOTTOM_TEXT CENTER_TEXT TOP_TEXT	0 1 2	PC la baza PC in centru PC deasupra

Alegerea dimensiunii, a tipului de caracter şi a direcţiei textului se face cu funcţia:

void settextstyle(int font, int directie, int mărime);

unde parametrii sunt explicitaţi in tabelele 7.7 şi 7.8.

Tabelul 7.7

Tipuri de caractere (fonturi) asociate modului grafic

Nume	Valoare	Descriere
DEFAULT_FONT	0	8x8 mapped font
TRIPLEX_FONT	1	strk.trip.font
SMALL_FONT	2	strk. small font
SANS_SERIF_FONT	3	strk.sans serif font
GOTHIC_FONT	4	strk. gothic font

Tabelul 7.8

Codificarea direcţiei textului pentru modul grafic

Nume	Valoare	Descriere
HORIZ_DIR	0	stinga-dreapta
VERT_DIR	1	jos-sus

Precizările anterioare sunt valabile pentru afişarea textului cu una din funcţiile:

void outtext(" char text "); în poziţia curentă

void outtextxy(int x,int y," char text "); începând cu punctul (x,y).

Daca textul afişat are nevoie de formatare se poate folosi funcţia sprinf conform exemplului de mai jos.

```
char x[20];
sprintf("Valoarea lui x este:%d", x);
outtextxy(10, 20, x);
```

Observaţii

1. elementele prezentate în tabelele 7.1 ÷ 7.8 pot intra în functiile utilizate fie cu *numele complet*, fie cu *codul;*
2. pentru detalii in legatură cu utilizarea modului grafic se poate consulta documentaţia mediului de programare.

În programul de mai jos sunt utilizate o parte din funcţiile specifice modului grafic, descrise anterior.

```
/*
        Program graf2.c – functii video in modul grafic
*/

#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <stdlib.h>
#include <dos.h>

#define CLIP_ON 1

void sunet(int f1)
 {
  sound(f1),delay(100),sound(2*f1),delay(200),sound(f1),delay(100),nosound();
 }

void main()
{
  int i,j;
  int gdrv=VGA,gmod=VGAHI,errorcode;  /* adaptor VGA, submod VGAHI
                                         (640 x 480 pixeli, 16 culori) */
  initgraph(&gdrv, &gmod, "c:\\bc\\bgi");    /* initializare mod grafic */
  errorcode=graphresult();

 if (errorcode != grOk)
  {
        printf("Graphics error: %s\n", grapherrormsg(errorcode));
        exit(1);
  }

  cleardevice();          /* sterge ecranul */

  setbkcolor(RED);
  settextstyle(TRIPLEX_FONT,HORIZ_DIR,3);
  outtextxy(90,50,"SUNTEM IN MODUL GRAFIC ! APASATI O TASTA !");
  rectangle(100,100,400,400);
  sunet(500);
  getch();
  setfillstyle(1,BLUE);
  floodfill(200,200,WHITE);
  sunet(600);
  getch();
  circle(250,250,150);
  sunet(650);
  getch();
  setfillstyle(1,CYAN);
  floodfill(250,250,WHITE);
```

```
sunet(700);
getch();
setviewport(175,175,325,325,CLIP_ON);clearviewport();
floodfill(250,250,WHITE);
setfillstyle(1,MAGENTA);
sunet(450);
getch();
settextstyle(4,0,4);
outtextxy(30,70,"ECHIPAMENTE");
outtextxy(40,80,"NUMERICE");
sunet(850);
getch();
restorecrtmode();
textattr(RED*16|WHITE|BLINK);
gotoxy(5,5);
cprintf(" SUNTEM IN MODUL ALFANUMERIC ! ");
sunet(750);
getch();
window(10,10,40,20);
textbackground(BLUE);
textcolor(YELLOW);
for(i=1; i<=10; i++)
  {
   for(j=1; j<=10; j++)
     {
       gotoxy(j,i);
       cprintf("~");
     }
  }
gotoxy(3,5);cprintf("E.N.");
sunet(1000);
getch();
setgraphmode(getgraphmode());  /* se restaureaza modul grafic */
setbkcolor(LIGHTBLUE);
settextstyle(TRIPLEX_FONT,HORIZ_DIR,3);
outtextxy(90,50,"SUNTEM DIN NOU IN MODUL GRAFIC !");
outtextxy(90,60,"APASATI O TASTA!");
bar3d(100,100,300,300,20,1);
sunet(700);
getche();
}
```

7.3. Modul de lucru:

I. Etape pregătitoare:

a) Se pregătesc matrice de lucru (640x480) şi (80x25) pentru lucrul în regim grafic, respectiv alfanumeric. Matricele vor fi pregătite pe hârtie de calc şi se vor multiplica ulterior;

b) Se vor construi pentru modul grafic functiile:

- void axe(x0, y0, deltax, deltay)

- void bargraf(stinga, jos, sus, param, delta, culoare)

unde: x0, y0 - sunt coordonatele originii;
deltax, deltay - lungimile în pixeli ale axelor;
stinga, jos, sus - coordonatele bargrafului;
param - valoarea în pixeli a parametrului care se vizualizează pe bargraf;
delta - lăţimea în pixeli a bargrafului;
culoare - culoarea de umplere a bargrafului;
(Toate variabilele sunt de tip întreg).

c) Se construiesc programele de apel ale funcţiilor axe şi bargraf.

II. Desfăşurarea lucrării

a) Se porneşte sistemul;
b) Se intra în subdirectorul de lucru al subgrupei ;
c) Se lansează mediul de programare;
d) Se compilează, linkeditează şi se lansează în execuţie programele P1, P2, GRAF1, GRAF2.
e) Se va observa diferenţa între modurile alfanumeric şi grafic;
f) Se vor edita, compila, linkedita şi executa programele realizate in cadrul etapei I;
g) Lucrarea se consideră încheiată când toate programele menţionate sunt funcţionale, iar referatul va cuprinde principalele idei din breviar şi codurile sursă ale programelor create.

7.4. Întrebări şi exerciţii

1. Să se scrie un program care să citească de la tastatură un număr întreg, apoi să scrie pe ecran următoarele informaţii:
Aţi introdus numărul ... (numărul introdus)
Numărul este ... (negativ, zero, pozitiv)
Modulul sau este ... (modulul numărului introdus)
2. Să se scrie un program care să afişeze numerele de la 1 la 100, câte 5 pe un rând. De fiecare dată când se trece la un rând nou acest fapt să fie semnalat printr-un semnal sonor scurt.
3. Să se scrie un program care să ceara introducerea de la tastatură a unui şir de caractere pe care apoi să-l afişeze literă cu literă până la întâlnirea primului spaţiu, moment în care programul va trebui să se oprească. Daca şirul nu conţine nici un spaţiu, el va fi afişat integral.
4. Să se scrie un program care să citească de la tastatură un şir de caractere şi o literă, apoi sa numere apariţiile literei respective în şirul introdus. Mesajul dat trebuie să fie de forma:
Litera ... apare de ... ori in şirul introdus.

5. Să se elaboreze un program care să testeze dacă un număr natural introdus de la tastatură este număr prim. După efectuarea testului utilizatorul va fi chestionat dacă doreşte să testeze un alt număr (să se răspundă cu D/N).

6. Să se scrie un program care să simuleze funcţionarea unui contor, astfel:

- la apăsarea tastei S contorul să înceapă numărarea, de la ultima valoare afişată;
- la apăsarea tastei O contorul să se oprească, rămânând afişată valoarea acestuia
- la apăsarea tastei R contorul să revină la zero si dacă anterior funcţiona, să se oprească;
- la apăsarea tastei E să se iasă din program.

LUCRAREA 8

PROGRAMAREA SECTIUNII ANALOGICE A INTERFEŢEI AX 5411 PENTRU INTRĂRI ŞI IEŞIRI ANALOGICE

8.1.Obiectivele lucrării

- Însuşirea modului de utilizare a funcţiilor driver-ului interfeţei de proces AX 5411;
- Determinarea caracteristicilor statice pentru subsistemul intrărilor şi ieşirilor analogice (SADA) al interfeţei AX 5411;
- Vizualizarea comportării dinamice a subsistemelor intrărilor şi ieşirilor analogice, cuplate;
- Elaborarea unor programe pentru vizualizarea comportării dinamice (în planul **u-t** şi pe bargraf) a subsistemului intrărilor şi al subsitemului ieşirilor analogice.
- Elaborarea unui program pentru determinarea caracteristicii statice a ansamblului celor două subsisteme.

8.2. Prezentarea conţinutului de lucru

CARACTERISTICI PRINCIPALE ALE SUBSISTEMULUI DE INTRĂRI ANALOGICE PENTRU INTERFAŢA AX 5411

AX5411 este o interfaţă multifuncţională care conţine toate cele 4 subsisteme (SADA, SADN, SDCA, SDCN) şi care poate fi montată într-un slot disponibil al unui sistem IBM PC AT sau compatibil.

În continuare se prezintă principalele specificaţii ale echipamentului.

Subsistemul de achiziţie a datelor analogice (SADA)

Număr de intrări	16 simple (AI0 - AI15);
Rezoluţie CAN	12 bit;
Frecvenţă de achiziţie	max. 60 kHz ;
Timpul de conversie A/D	max 15 microsec. ;
Timpul de achiziţie	max 5 microsec./canal;
Domenii de intrare	+/-10V,+/-5V,+/-2.5V,+/-1.25V, +/-0.625V,+/- 0.3125V (selectabile software);
Impedanţa de intrare	>10 MΩ,50pF;
Neliniaritate	+/- 1 LSB;
Eroare inerentă	+/- 1 LSB.

Subsistemul de generare a comenzilor analogice (SDCA)

Număr de ieşiri	2 (DO0,DO1);
Rezoluţie	12 bit;
Frecvenţă	max 33 kHz ;
Domenii de ieşire	0…5V,0…10V (selectabile hardware);
Curent de ieşire	5 mA max.

Subsistemul intrarilor/ieşirilor numerice (SADN/SDCN)

Intrări numerice	24 dintre care disponibile 8 (DI0 - DI7);
Ieşiri numerice	24 dintre care disponibile 8 (DO0 - DO7).

Nivele intrare/ieşire compatibile TTL;

Conector intrare/ieşire 50 pini

Caracteristici de interfaţare cu calculatorul

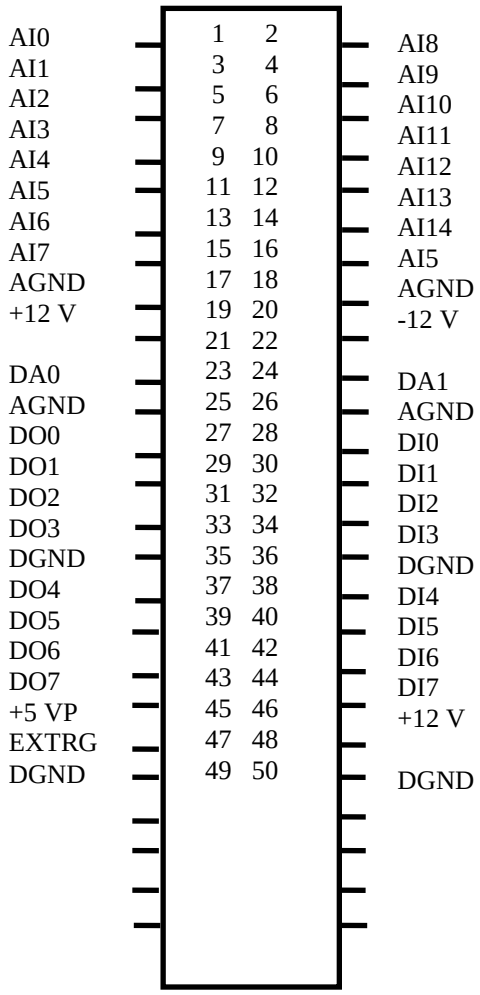
Magistrala	compatibilă IBM PC AT;
Biţi adresă utilizaţi	A9 - A0;
Adresa de bază (IOPORT)	0300 hexa;
Locaţii necesare	16 (octeţi);
Nivele de întrerupere	2,3,4,5,6,7 (controlabile soft);
Sursa de întreruperi	FINISH bit conversie A/D;
Opţiuni DMA	DMA1 sau DMA3 selectabile hard;

În fig. 8.1 se prezintă asignarea pinilor în conectorul interfeţei AX 5411.

CARACTERISTICILE DRIVER-ULUI C PENTRU SUBSISTEMUL INTRĂRILOR ANALOGICE PENTRU INTERFAŢA AX 5411

- ***sintaxa:***

  flag=ax5411(fun, dio, ary1, ary2);



Nume	I/O	Funcţie
AIx	Intrare	Canal x Int.Anal
DAx	Ieşire	Canal x Ieşire Anal.
Dix	Intrare	Canal x Int. Num.
DOx	Ieşire	Canal x Ieşire Num.
+12v	Sursă	-
-12v	Sursă	-
AGND	Masă	Analogic
DGND	Masă	Numeric
+5vp	Sursă	PC +5v
+12v	Sursă	PC +12v
-12v	Sursă	PC -12v
EXTRG	Intrare	Trigger extern

Fig. 8.1. Asignarea şi semnificaţiile semnalelor din conectorul interfeţei AX 5411.

```
if (flag == 0)
  procesare normala;
else
  procesare cu erori;
```

- *parametri:*

int fun : va conţine numele sau numărul funcţiei din driver (funcţiile ce se vor utiliza în lucrările de laborator sunt prezentate in tabelul 8.1);

int dio[ ] : tablou de întregi (maximum 7 elemente), care conţine toţi parametrii necesari funcţiei respective;

int far *ary1: tablou de întregi (maximum 32 Kwords);

int far *ary2 : tablou de întregi (maximum 32 Kwords);

În continuare sunt detaliate funcţiile din acestă lucrare, urmând ca celelalte să fie prezentate pe măsura utilizării.

Tabelul 8.1

Codificarea principalelor funcţii din driver

Număr	Funcţie	Descriere
0	INIT	Iniţializează funcţiile driver-ului AX5411
1	SET_CH	Setează canalul de achiziţie
2	SET_GAIN	Setează domeniul pentru toate canalele
3	SFT_TRG	Realizează o achiziţie pe un canal
13	DO_BYTE	Înscrie un octet în canalele DO0...DO7
14	DI_BYTE	Citeşte un octet din canalele DI0...DI7
15	DAC_ONE	Trimite un semnal analogic pe un canal
16	DAC_TWO	Trimite semnale analogice pe ambele canale

Funcţia INIT : iniţializare AX 5411

Scop:

1. Setează adresa de bază conform dio[0];
2. Setează nivelul de întreruperi conform dio[1];
3. Setează nivelul DMA conform dio[2];
4. Verifică prezenţa modulului AX 5411;
5. Dezactivează alte întreruperi externe.

Aceasta funcţie se va executa o singură dată, înaintea oricărei alte funcţii.

Intrări:

fun - INIT (sau 0);
dio[0] - adresa portului de bază (IOPORT);
dio[1] - nivel IRQ (de la 2 la 7);
dio[2] - canal DMA (1 sau 3);
dio[3] - neutilizat;
dio[4] - neutilizat;
dio[5] - neutilizat;
dio[6] - neutilizat;
∗ary1 - neutilizat;
∗ary2 - neutilizat.

Ieşiri:

flag = 0 - fără erori;
= 2 - funcţie în afara domeniului;
= 3 - eroare de setare a IOPORT;
= 4 - nivel IRQ setat în afara domeniului;
= 5 - canal DMA setat în afara domeniului;
= 6 - defect hardware.

Exemplu de utilizare a funcţiei INIT:

```
#include <ax5411.h>
#include <stdio.h>
#define IOPORT 0x300
//
int fun, flag;
int far *ary1;
int far *ary2;
unsigned int dio[7];
//
main( )
{
 fun=INIT;
 dio[0]=IOPORT;
 dio[1]=3;
 dio[2]=3;
 flag=ax5411(fun,dio,ary1,ary2);
 if(flag != 0)
  printf("Eroare de initializare\n");
 else
  printf("Initializare reusita");
 getch();
}
```

Funcţia SET_CH : setare canale achiziţie

Scop:

1. Setează numărul primului canal conform dio[0];
2. Setează numărul ultimului canal conform dio[1].

Aceasta funcţie se va executa o singură dată, înaintea funcţiei de achiziţie.

Intrări:

fun - SET_CH (sau 1);
dio[0] - numărul canalului de start (0..15);
dio[1] - numărul canalului de stop (0..15);
dio[2] - neutilizat;
dio[3] - neutilizat;

dio[4] - neutilizat;
dio[5] - neutilizat;
dio[6] - neutilizat;
*ary1 - neutilizat;
*ary2 - neutilizat.

Ieşiri:
flag = 0 - fără erori;
= 1 - driver neiniţializat;
= 2 - funcţie în afara domeniului;
= 7 - nr. canal în afara domeniului (0..15).

Exemplu de utilizare a funcţiei SET_CH:

```
.
fun=SET_CH;                          // funcţia set canal
dio[0]=0;                            // nr. canal start
dio[1]=15;                           //nr. canal stop
flag=ax5411(fun,dio,ary1,ary2);
if(flag != 0)
  printf("Eroare de setare canal\n");
else
  printf("Setare reusita/n");
    .
```

Funcţia SET_GAIN : setare domeniu canale achiziţie

Scop:
Setează domeniul pentru toate canalele conform dio[0].

Aceasta funcţie se va executa o singură dată, înaintea funcţiei de achiziţie.

Intrări:
fun - SET_GAIN (sau 2); *)
dio[0] - domeniul (1,2,4,8,16);
dio[1] - neutilizat;
dio[2] - neutilizat;
dio[3] - neutilizat;
dio[4] - neutilizat;
dio[5] - neutilizat;
dio[6] - neutilizat;
*ary1 - neutilizat;
*ary2 - neutilizat.

*)Valoare	Domeniu
1	-5/+5 V
2	-2.5/+2.5 V
4	-1.25/+1.25 V
8	-0.625/+0.625 V
16	-0.3125/+0.3125 V

Ieşiri:

flag = 0 - fara erori;
= 1 - driver neiniţializat;
= 2 - funcţie în afara domeniului;
= 22 - eroare setare domeniu.

Exemplu de utilizare a funcţiei SET_GAIN:

```
.
fun=SET_GAIN;          // functia set domeniu
dio[0]=1;              // domeniu selectat
flag=ax5411(fun,dio,ary1,ary2);
if(flag != 0)
  printf("Eroare de setare domeniu\n");
else
  printf("Setare reusita/n");
.
```

Funcţia SFT_TRG : realizează conversia A/N

Scop:

Această funcţie realizează o achiziţie şi conversia analog/numerică pentru canalele şi domeniile selectate. Declanşarea conversiei (start conversie) se realizează pe cale software (**S**o**FT**ware_**TR**i**G**ger).

Funcţia se va executa înaintea unei conversii analog-numerice

Intrări:

fun - SFT_TRG (sau 3);
dio[0] - neutilizat;
dio[1] - neutilizat;
dio[2] - neutilizat;
dio[3] - neutilizat;
dio[4] - neutilizat;
dio[5] - neutilizat;
dio[6] - neutilizat;
∗ary1 - neutilizat;
∗ary2 - neutilizat;

Ieşiri:

flag = 0 - fără erori;
= 1 - driver neiniţializat;
= 2 - funcţie în afara domeniului;
= 8 - eroare start conversie;
dio[0] = număr de cuante în zecimal -2048 la +2047;
dio[1] = numărul canalului pe care se achiziţionează.

Exemplu de utilizare a funcţiei SFT_TRG:

```
    .
fun=SFT_TRG;                          // functia software trigger
flag=ax5411(fun,dio,ary1,ary2);
if(flag != 0)
  printf("Eroare de achizitie\n");
else
  printf("Data=%x, Canal=%x\n",dio[0],dio[1]);
    .
```

UTILIZAREA DRIVER-ULUI C PENTRU AX 5411 ÎN APLICAŢII PENTRU SUBSISTEMUL INTRĂRILOR ANALOGICE

În vederea utilizării driver-ului C pentru interfaţa AX 5411, în aplicaţii se va proceda după cum urmează:

a) se editează programul utilizator;

b) se construieşte un proiect (extensia .prj) conform exemplului de mai jos;

c) se înscrie numele proiectului in funcţia **project** a mediului de programare;

d) se efectuează compilarea, linkeditarea şi execuţia conform metodologiei cunoscute.

Exemplu: program de achiziţie de pe canalul **0**, domeniu **-5/+5V** şi tipărirea numărului de cuante. Acest program se găseşte în subdirectorul de lucru sub numele ***TESTAI.CPP***.

```
//  PROGRAM TESTAI TESTARE ACHIZITIE PE CANALUL 0
//
    #include <ax5411.h>
    #include <stdio.h>
    #include <conio.h>
    #include <dos.h>
    #define  IOPORT  0X300          // adresa port de baza pentru AX 5411
    int fun,flag;
    int far *ary1;
    int far *ary2;
    unsigned int dio[7];
    int main( )
    {
     clrscr();
      // initializare AX5411
     fun=INIT;                              // functia de initializare
     dio[0]=IOPORT;
     dio[1]=3;
     dio[2]=3;
     flag=ax5411(fun,dio,ary1,ary2);
     if(flag != 0)
    printf("Eroare de initializare \n");
     // setare numar canal
```

```
 fun=SET_CH;                    //functia set canal
 dio[0]=0;                      //nr. canal de start
 dio[1]=0;                      //nr. canal de stop
 flag=ax5411(fun,dio,ary1,ary2);
 if(flag != 0)
printf("Eroare de setare canal\n");

 // setare domeniu
 fun=SET_GAIN;                  //functia set domeniu
 dio[0]=1;                      //domeniu selectat -5/+5V
 flag=ax5411(fun,dio,ary1,ary2);
 if(flag != 0)
printf("Eroare de setare domeniu\n");
 textattr(GREEN*16|YELLOW);

 // bucla infinita
 for(;;)
  {
    delay(100);                 //asteapta 100 ms

    // achizitie
    fun=SFT_TRG;                //functia software trigger
    flag=ax5411(fun,dio,ary1,ary2);
    if(flag != 0)
     printf("Eroare de achizitie\n");
    else

    //  tiparire
     gotoxy(20,20),
     cprintf("Data=%d, Canal=%d\n",dio[0],dio[1]);
    if(kbhit()) break;          // la apasarea unei taste iesire din ciclu
  }
getch( );                       // iesire din program
return 0;
}
```

CARACTERISTICI PRINCIPALE ALE SISTEMULUI IEŞIRLOR ANALOGICE PENTRU INTERFAŢA AX 5411

În continuare se prezintă funcţiile specifice subsistemului ieşirilor analogice. Pentru detalii privind caracteristicile sistemului de interfaţa, vezi precizările anterioare.

Subsistemul ieşirilor analogice

Număr de ieşiri	2 (DO0,DO1);
Rezoluţie	12 bit;
Frecvenţă	max. 33 kHz ;
Domeniu de ieşire	0..5V;
Curent de ieşire	max 5 mA .

CARACTERISTICILE DRIVER-ULUI C PENTRU AX 5411 ÎN SISTEMUL IEŞIRILOR ANALOGICE

Pentru detalii privind:
- utilizarea driver-ului;
- codificarea funcţiilor;

funcţiile *INIT,SET_CH,SET_GAIN,SFT_TRG*,
vezi subsistemul intrărilor analogice.

În continuare se prezintă funcţiile aferente ieşirilor analogice.

Funcţia DAC_ONE : realizează conversia N/A pe un canal

Scop:

Aceasta funcţie realizează o conversie numeric-analogica (N/A) şi generează tensiunea rezultată pe canalul specificat în domeniul setat pe cale hardware (0..5V pentru prezenta lucrare).

Valoarea tensiunii generate se obţine cu relaţia:

$$U_{gen} = \frac{Nr.cuante}{4096} * 5$$

Intrări:

fun - DAC_ONE (sau 15);
dio[0] - canal nr. 0 sau 1;
dio[1] - nr. cuante care se aplică CNA (intre 0 si 4095);
dio[2] - neutilizat;
dio[3] - neutilizat;
dio[4] - neutilizat;
dio[5] - neutilizat;
dio[6] - neutilizat;
*ary1 - neutilizat;
*ary2 - neutilizat;

Ieşiri:

flag = 0 - fără erori;
= 1 - driver neiniţializat;
= 2 - functie în afara domeniului;
= 21 - nr. cuante în afara domeniului;
= 20 - nr. canal în afara domeniului (0 sau 1).

Exemplu de utilizare a funcţiei DAC_ONE:

```
.
fun=DAC_ONE;                    // conversie N/A pe un canal
dio[0] = 0;                     // canal 0
dio[1] = 1024;                  // nr. cuante
flag=ax5411(fun,dio,ary1,ary2);
if(flag != 0)
  printf("Eroare de conversie !,flag=%d\n",flag);
else
  printf("Canal=%d, Tens=%.2f V\n",dio[0],dio[1]*4096/5);
```

Funcţia DAC_TWO : realizează conversia N/A pe ambele canale.

Scop:

Aceasta funcţie realizează o conversie N/A şi generează tensiunea rezultată pe ambele canale (0 şi1), în domeniul setat pe cale hardware (0..5V pentru prezenta lucrare).

Valoarea tensiunii generate se obţine cu relaţia:

$$U_{gen} = \frac{Nr.cuante}{4096} * 5$$

Intrări:

fun - DAC_TWO (sau 16);
dio[0] - nr. cuante pe canalul nr. 0 (0..4095);
dio[1] - nr. cuante pe canalul nr. 1 (0..4095);
dio[2] - neutilizat;
dio[3] - neutilizat;
dio[4] - neutilizat;
dio[5] - neutilizat;
dio[6] - neutilizat;
*ary1 - neutilizat;
*ary2 - neutilizat.

Iesiri:

flag = 0 - fără erori;
= 1 - driver neiniţializat;
= 2 - funcţie în afara domeniului;
= 21 - nr. cuante în afara domeniului.

Exemplu de utilizare a funcţiei DAC_TWO:

```
.
fun=DAC_TWO;                    // conversie A/N pe un canal
dio[0] = 1024;                  // nr. cuante pe canal 0
dio[1] = 2048;                  // nr. cuante pe canal 1
flag=ax5411(fun,dio,ary1,ary2);
if(flag != 0)
  printf("Eroare de conversie !,flag=%d\n",flag);
else
  printf("Canal 0 Tens=%.2f V\n",dio[0]*4096/5);
  printf("Canal 1 Tens=%.2f V\n",dio[0]*4096/5);
```

UTILIZAREA DRIVER-ULUI C PENTRU AX 5411 ÎN APLICAŢII PENTRU SISTEMUL IEŞIRLOR ANALOGICE

În vederea utilizării driver-ului C pentru AX 5411 în aplicaţii se va proceda după cum urmează:

a) se editează programul utilizator;

b) se construieşte un proiect (extensia .prj) conform exemplului de mai jos;

c) se înscrie numele proiectului în funcţia *project* a mediului de programare;

d) se efectuează compilarea, linkeditarea şi execuţia conform metodologiei cunoscute.

Exemplu:

Program de generare pe canalul de ieşire 0 de tensiuni din 0.5 V în 0.5 V , achiziţia acestora pe canalul de intrare 0 şi tipărirea atât a tensiunii generate cât şi celei achiziţionate. Trecerea de la o valoare la alta se face prin apăsarea tastei c. Acest program se găseşte în subdirectorul de lucru sub numele *TESTAO.CPP*.

```
 //
 //  PROGRAM TESTARE IESIRI ANALOGICE PE CANALUL AO0
 //  SI ACHIZITIA VALORII GENERATE PE CANALUL AI0
 //
#include  <ax5411.h>
#include  <stdio.h>
#include  <conio.h>
#include  <dos.h>
#define IOPORT 0x300
int fun,flag,i;
int far *ary1;
int far *ary2;
float uach[20],ugen[20];
unsigned int dio[7];

main( )
{
 // initializare AX 5411
 fun=INIT;                                  // functia de initializare
 dio[0]=IOPORT;
 dio[1]=3;
 dio[2]=3;
 flag=ax5411(fun,dio,ary1,ary2);
 if(flag != 0)
  printf("Eroare de initializare\n");

 // setare numar canal
 fun=SET_CH;                  // functia set canal
 dio[0]=0;                          // nr. canal start
 dio[1]=0;                          // nr. canal stop
 flag=ax5411(fun,dio,ary1,ary2);
 if(flag != 0)
  printf("Eroare de setare canal\n");

 // setare domeniu
 fun=SET_GAIN;                      // functia set domeniu
 dio[0]=1;                          // domeniu selectat -5/+5V
 flag=ax5411(fun,dio,ary1,ary2);
 if(flag != 0)
```

```
  printf("Eroare de setare domeniu\n");
// tipareste cap de tabel
  textbackground(BLACK);
 clrscr( );
 textattr(BLUE*16|YELLOW);
 gotoxy(27,5);cprintf(" * DEMO SUBSISTEME D/A + A/D * ");
 gotoxy(30,7);cprintf(" i  ugen[V]   uach[V] ");
 gotoxy(30,8);cprintf(" ==================== ");
   // bucla cu 9 pasi
   for(i=1;i<=9;i++)
  {
   // generare tensiune
   dio[0]=0;                          // canal iesire nr. 0
   ugen[i]=0.5*i;
   dio[1]=(int)(409.6*i);
   fun=DAC_ONE;
   flag=ax5411(fun,dio,ary1,ary2);
   if(flag != 0)
    printf("Eroare de conversie D/A\n");
   else
    // achizitie tensiune
    fun=SFT_TRG;                      // functia software trigger
    flag=ax5411(fun,dio,ary1,ary2);
   if(flag != 0)
    printf("Eroare de achizitie\n");
   else
    uach[i]=dio[0]*0.00244;
    // tiparire
    gotoxy(30,8+i);
    cprintf(" %d    %.2f     %.2f   \n",i,ugen[i],uach[i]);
     // avans pentru o noua generare
    getch( );
  }
    // iesire din program
   getch( );
}
```

8.3 Partea experimentală

A. Descrierea platformelor de lucru

Laboratorul **“CALCULATOARE DE PROCES” conţine** următoarele componente funcţionale importante:

- panoul complex IAN;

- platformele de lucru P1, P2, P3;
- stand pentru reglarea presiunii.

Platformele de lucru P_1, P_2, P_3 sunt conectate la panoul I.A.N. prin intermediul cablurilor prezentate punctat în fig. 8.2 şi pozate în interiorul unor bare casetate. Conectarea fizică la panourile interfeţelor de proces se realizează prin şirurile de conectori SC_1 , SC_2, SC_3, care au configuraţia din fig. 8.3.

Acest laborator este interconectat din punct de vedere funcţional cu laboratorul "Automatizarea proceselor chimice" de unde se preiau semnale în curent de la traductoarele de debit (platforma P2) şi nivel (platforma P1) şi către care se trimit semnale în curent la elementele de execuţie aferente SRA debit şi SRA nivel.

Comanda pompelor centrifugale ale celor două SRA se poate realiza de la panoul IAN.

La platforma P1 sunt conectate traductorul şi elementul de execuţie amplasate pe standul pentru reglarea presiunii.

B. Desfăşurarea lucrării

a) Se realizează pe rând montajul din fig.8.2 respectiv cel din fig.8.3 pentru fiecare platforma P_i (i=1,2,3);

b) Se porneşte sistemul ;

c) Se intră în subdirectorul de lucru al subgrupei;

d) Se lansează mediul de programare**;**

e) Se editează proiectul după cum urmează:

- pentru subsistemul intrărilor analogice:
- se tastează F10 şi se selectează meniul *Project*;
- se selectează submeniul *Open project...*;
- se introduce numele proiectului;
- se apasă tasta *Insert* şi se adaugă fişierele :

testai.cpp (existent în subdirectorul curent)

ax5411cl.lib (existent în subdirectorul C:\BC\LIB)

- se salvează proiectul din meniul *Options*, submeniul *Save...*

- pentru subsistemul ieşirilor analogice:

- se tastează F10 şi se selectează meniul *Project;*
- se selectează submeniul *Open project...*;
- se introduce numele proiectului (LUC6.PRJ);
- se apăsa tasta *Insert* şi se adaugă fişierele :

testao.cpp (existent în subdirectorul curent)

ax5411cl.lib (existent în subdirectorul C:\BC\LIB)

- se salvează proiectul din meniul *Options*, submeniul *Save...*

f) Se alimentează panoul cu tensiune;

g) Se compilează proiectul;

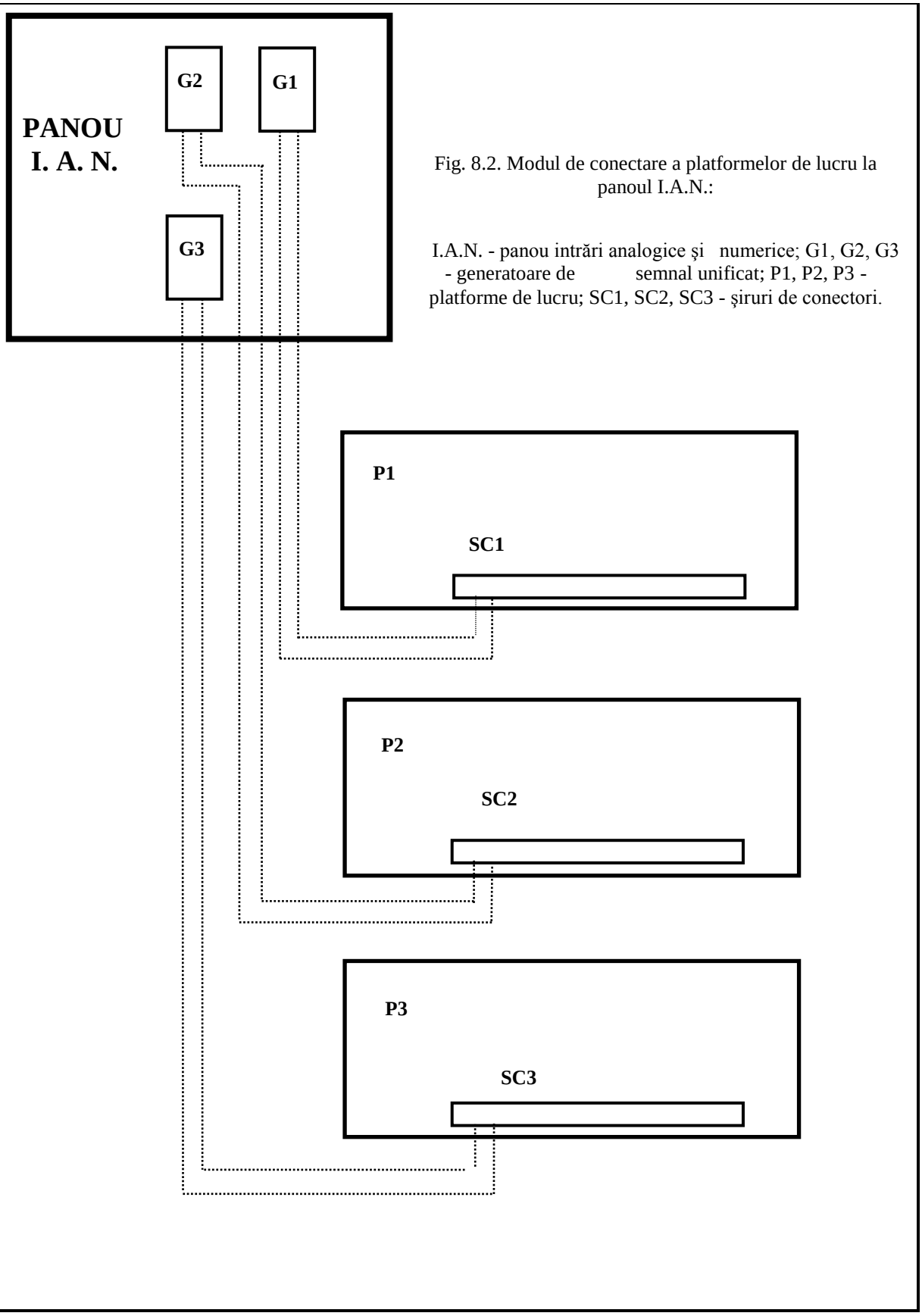


Fig. 8.2. Modul de conectare a platformelor de lucru la panoul I.A.N.:

I.A.N. - panou intrări analogice şi numerice; G1, G2, G3 - generatoare de semnal unificat; P1, P2, P3 - platforme de lucru; SC1, SC2, SC3 - şiruri de conectori.

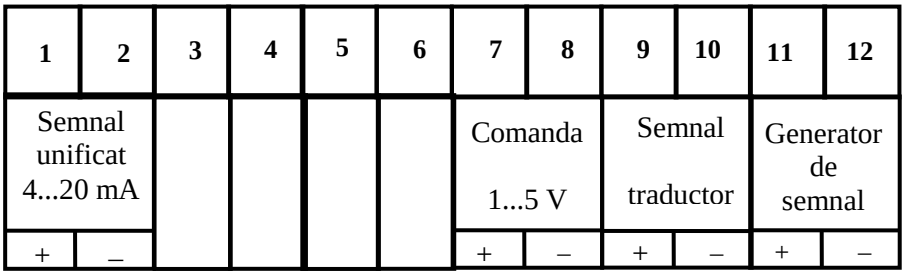


Fig. 8.3. Configuraţia şirului de conectori.

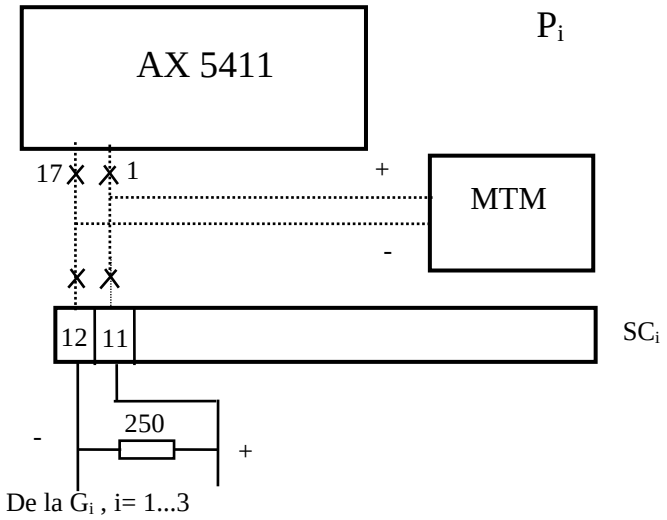


Fig. 8.4.Montajul utilizat pentru testarea intrărilor analogice:

SC_i - sir de cleme; G_i - generator de semnal unificat; MTM - multimetru;

h) Se linkeditează şi se lansează în execuţie proiectul creat;

i) Pentru verificarea programului se modifică tensiunea de la panou şi se urmăresc valorile afişate pe ecran;

j) Se lansează din subdirectorul de lucru programul *pr2_itf.exe* şi se determină caracteristica statică a canalului **0.** Se notează tensiunile de intrare şi ieşire, coeficienţii dreptei de regresie, abaterea standard, dispersia şi se trasează pe hârtie milimetrică caracteristica statică;

k) Se lansează în execuţie din subdirectorul de lucru programul *pr4_itf.exe* şi prin modificări ale tensiunii de la panou se urmăreşte comportarea dinamică a achiziţiei pe canalul 0;

l) Se completează programul *TESTAI* în aşa fel încât să permită vizualizarea tensiunii achiziţionate într-un sistem de axe **u-t** şi pe bargraf. În acest scop se vor utiliza funcţiile axe si bargraf elaborate în cadrul lucrării anterioare. Achiziţia se va realiza în 500 paşi cu temporizare 200 ms;

m) După funcţionarea programului se va nota sau, după caz, se va lista textul sursă la imprimantă;

n) Referatul va cuprinde principalele idei din breviar (referitoare la interfaţă şi la driver), caracteristicile statice ale SADA şi forma sursă a programului *TESTAI* completat.

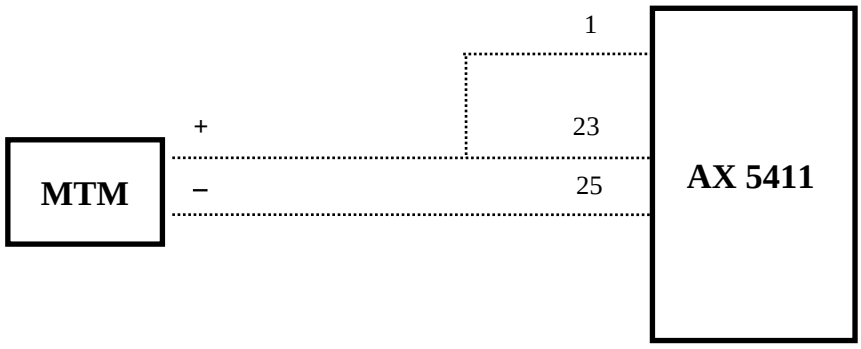


Fig. 8.5. Montajul utilizat pentru testarea ieşirilor analogice :

MTM - multimetru numeric.

o) Prin program se generează pe canalul *DAO* tensiuni crescătoare între 0.5 şi 4.5 V, trecerea de la o tensiune la alta efectuându-se prin apăsarea unei taste. Se verifică concordanţa dintre valorile afişate pe ecran şi cea indicată de multimetru;

p) Se lansează din subdirectorul de lucru programul *pr3_itf.exe* şi se determină caracteristica statică a canalului DA0. Se notează tensiunile de intrare şi ieşire, coeficienţii dreptei de regresie, abaterea standard, dispersia şi se trasează pe hârtie milimetrică caracteristica statică;

r) Se lansează în execuţie din subdirectorul de lucru programul *pr5_itf.exe* şi se urmăreşte comportarea dinamică a canalelor cuplate DA0 şi AI0 pentru diferite forme de variaţie a tensiunii generate (liniară, sinusoidală, exponenţială);

s) Se completează programul *TESTAO.CPP,* astfel încât să permită trasarea caracteristicii statice $\mathbf{U_{ach} = f(U_{gen})}$ a ansamblului celor două canale. În sistemul de axe se vor marca punctele experimentale şi se va trasa dreapta obţinută prin regresie liniară. Se vor calcula şi afişa de asemenea abaterea standard şi dispersia.

LUCRAREA 9

VERIFICAREA UNOR TEOREME ŞI AXIOME ALE ALGEBREI LOGICE

9.1.Obiectivele lucrării

- Revederea cunoştinţelor de programare în limbajul C;
- Elaborarea de programe în limbajul C, prin care să se testeze proprietăţile funcţiilor logice şi să se verifice teoremele şi axiomele algebrei binare,

9.2. Prezentarea conţinutului lucrării

Desfăşurarea lucrării de laborator presupune deţinerea de cunoştinţe în domeniul algebrei binare, dintre care, în continuare, se vor prezenta succint cele de bază.

Fie mulţimea B = {x | x = 0; 1 }, unde prin definiţie:

0 = elementul nul;

1 = elementul unitate (universal).

În exprimarea curentă, referirea la unul din cele două elemente ale mulţimii B se face prin noţiunea de bit.

Pe mulţimea B se definesc 3 operatori principali:

- complementarea (negaţia, *NOT)*\
- disjuncţia logică (suma logică, *OR):*
- conjuncţia logică (produs logic, *AND).*

O modalitate comodă de definire a unei funcţii logice o reprezintă utilizarea TABELELOR DE ADEVĂR . În continuare se prezintă simbolurile şi implementarea cu contacte (fig. 9.1) şi tabelele de adevăr pentru cele trei funcţii logice fundamentale (tabelul 9.1).

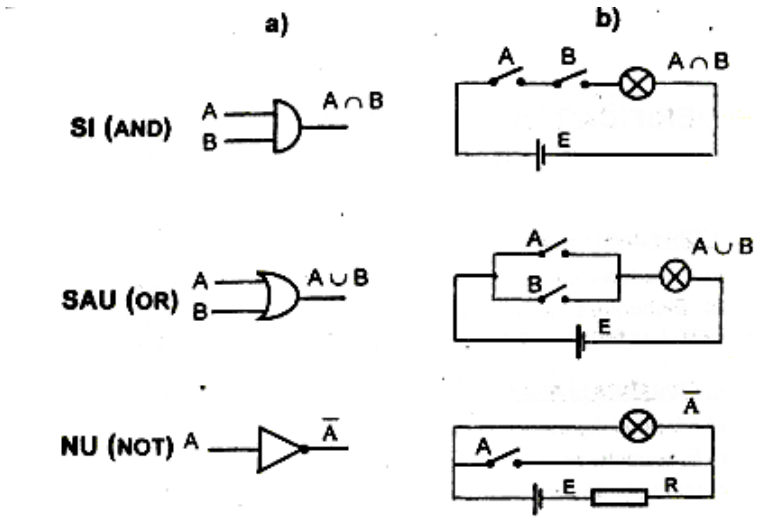


Fig. 9.1. Funcţii logice elementare:

a - simbol; **b** - implementare cu contacte.

Tabelul 9.1

Funcţii logice elementare (tabela de adevăr)

A	**B**	**A∩B**	**AUB**	**/A**
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Referitor la mulţimea B şi la operaţiile definite mai cus sunt valabile 6 axiome şi 5 teoreme care sunt prezentate sintetic în tabelul 9.2.

Tabelul 9.2

Axiomele şi teoremele algebrei binare

Tip	**Denumire**	**Forma sumă**	**Forma produs**
A_1	Închiderea mulţimii B în raport cu operaţiile +, •	$x+y \in B \Rightarrow x \cdot y \in B$	$x \cdot y \in B \Rightarrow x+y \in B$
A_2	Asociativitate	$x+(y+z)=(x+y)+z$	$x \bullet (y \bullet z) = (x \bullet y) \bullet z$
A_3	Comutativitate	$x+y=y+x$	$x \bullet y = y \bullet x$
A_4	Element neutru	$x+0=0+x$	$x \bullet 1 = 1 \bullet x$
A_5	Distributivitate	$x+y \bullet z=(x+y) \bullet (x+z)$	$x \bullet (y+z)=x \bullet y+x. \bullet z$
A_6	Existenta complementului	$x+\bar{x}=1$	$x \cdot \bar{x}=0$
T_1	Idempotenţa (tautologia)	$x+x=x$	$x \bullet x = x$
T_2	Element neutru	$x+1=1$	$x \bullet 0 = 0$
T_3	Teorema dublei negaţii	$\bar{\bar{x}}=x$	$\bar{\bar{x}}=x$
T_4	Absorbţia	$x+x \bullet y=x$	$x \bullet (x+y) = x$
T_5	Teoremele De Morgan	$\overline{x+y}=\bar{x} \cdot \bar{y}$	$\overline{x \cdot y}=\bar{x}+\bar{y}$

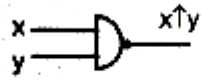
Una din metodele de demonstrare a teoremelor este aceea în care sunt utilizate tabelele de adevăr.

Perechile de operatori (NOT, AND) şi (NOT, OR) formează fiecare câte un sistem complet, în sensul ca orice relaţie definită pe mulţimea B poate fi exprimată numai cu operatorii unei perechi. Circuitul fizic care implementează un operator logic poartă denumirea generală de poartă logică.

Sisteme complete au fost realizate şi sub forma unor porţi logice denumite NAND (NOT-AND), NOR (NOT-OR) pentru care sunt valabile următoarele funcţii logice

$x \uparrow y = \overline{x \cdot y}$

$x \downarrow y = \overline{x + y}$



Alta poartă logică des utilizată este cea care implementează operaţia SAU EXCLUSIV (EXCLUSIVE OR), căreia îi este specifică funcţia logică de mai jos.

$x \oplus y = \bar{x} \cdot y + x \cdot \bar{y}$.

Observaţii:

1. Acest operator poate fi utilizat pentru determinarea coincidenţei a doi operanzi.
2. Daca unul din operanzi este 1 atunci SAU EXCLUSIV poate fi utilizat ca operator de negaţie.

În continuare se prezintă un program demonstrativ, în limbajul C, pentru testarea funcţiei logice ŞI.

```
/*
 Program care implementeaza funcţia SI
  - se citesc valori pentru variabilele binare a si b
  - se tipăreşte valoarea funcţiei f=ab
  - pentru reluare se apasă tasta y
*/
#include <sfdio.h>
#include <conio.h>
#include <std!ib.h>

void main(void)
{
    int a,b,f;
    char c;
    clrscr();
    while(1)
      {
            printf("a="); scanf("%d", &a);          /*  se citeşte a */
            printf("b="); scanf("%d", &b);          /* se citeşte b */
            f=a & b;                                /* se determina f */

            printf("%d AND %d =%d\n", a, b,f);    /*  se tipăresc a.b.f */
           printf("alt test ? (y/n)");          /* se testează opţiunea de
          continuare */
            c=getche();
           if((c=='n')||(c=='N'))  exit(1);
       }
}
```

9.3. Desfăşurarea lucrării

1. Se porneşte sistemul.
2. Se intra în subdirectorul de lucru al subgrupei;
3. Se lansează mediul de programare ;
4. Se introduce programul demonstrativ ;
5. Se compilează, se linkeditează şi se lansează în execuţie acest program;
6. Lucrarea se consideră încheiată în momentul în care toate programele (demonstrative şi dezvoltate sunt funcţionale). Referatul va cuprinde principalele idei din breviar şi textele sursă ale programelor.

9.4. Întrebări şi exerciţii

1. Se *completează* programul în aşa fel încât să se realizeze şi testarea funcţiilor SAU**,** NU, SAU EXCLUSIV (la intrarea în program se va testa opţiunea privind funcţia);
2. Se va *scrie şi se va executa un* program care sa permită implementarea funcţiilor logice SI, SAU, NU cu ajutorul porţilor NAND;
3. Se va scrie şi se va executa un program care sa permită demonstrarea echivalenţei funcţiilor logice f1 şi f2 din reprezentările de mai jos.