COMANDA MICROMOTOARELOR DE CURENT CONTINUU

1. Scopul lucrării

Scopul lucrării este de a prezenta principiile de bază privind posibilităţile de comandă, altele decât cele clasice, a micromotoarelor de current continuu. Se are în vedere în caest scop comanda micromotoarelor prin portul paralel al unui PC şi utilizarea unui circuit integrat dedicat .

2. Consideraţii teoretice

În evoluţia sistemelor mecatronice controlul mişcării ocupă un rol esenţial. Pentru a putea clarifica aspectele legate de controlul mişcării, un utilizator trebuie să aibă abilităţile:

- să înţeleagă utilitatea şi beneficiile controlului mişcării;
- să recunoască şi să analizeze aplicaţiile de control ale mişcării;
- să recunoască şi să analizeze categoriile majore ale sistemelor de control ale mişcării;
- să poată selecta într-o formă preliminară cea mai recomandată soluţie pentru aplicaţia în cauză;

În zilele de debut ale dezvoltării sistemelor mecanice mobile, controlul poziţiei şi a vitezei se asigurau prin soluţii costisitoare şi consum de timp bazate în exclusivitate pe componente mecanice: came, roţi dinţate, robinete etc. Frecvent alte dispozitive – cilindrii hidraulici sau pneumatici, electromagneţi, dispozitive mecanice diverse – s-au utilizat în acelaşi scop. Era perioada mecanizării.

Dezvoltarea industrială ulterioară a necesitat noi forme de control a mişcării în operaţii complexe integrate. Automatizarea rigidă a fost un pas mare realizat nu numai în creşterea productivităţii dar şi în ceea ce priveşte controlul mişcării.

Apariţia microprocesorului a definit debutul unei noi etape cel al automatizărilor flexibile. În sistemele electronice o mare varietate a parametrilor pot fi modificaţi prin schimbarea soft-ului din sistem. De exemplu a modifica o viteză înseamnă acum a introduce câteva linii de cod, sau a selecta un nou profil de viteză din memoria sistemului.

Controlul programabil al mişcării este definit ca şi o aplicaţie hardware şi software – împreună cu elemente senzoriale, actuatoare, dispozitive de reglare – pentru controlul uneia sau a mai multor mişcări liniare sau de rotaţie.

În ultimii 10-15 ani, partea de comandă a suportat mutaţii majore. Ele se refră în principal la creşterea ponderii comenzilor în logică programată, caracterizată prin flexibilitate funcţională ridicată, consumuri de energie reduse, volum şi greutate minime, fiabilitate mărită, posibilitatea implementării unor strategii de comandă performante. Soluţiile de comandă în logică programată implică utilizarea de structuri digitale complexe ca circuite VLSI speciale (ASIC – Application Special Integrated Circuit), microprocesoare (μP) de uz general sau cu funcţii speciale, microcontrolere, procesoare de semnal dar şi echipamente numerice precum automate programabile.

În figura 1 se prezintă sugestiv modul de interconectare a componentelor pentru comanda unui actuator într-o structură conformă cu cereinţele actuale.

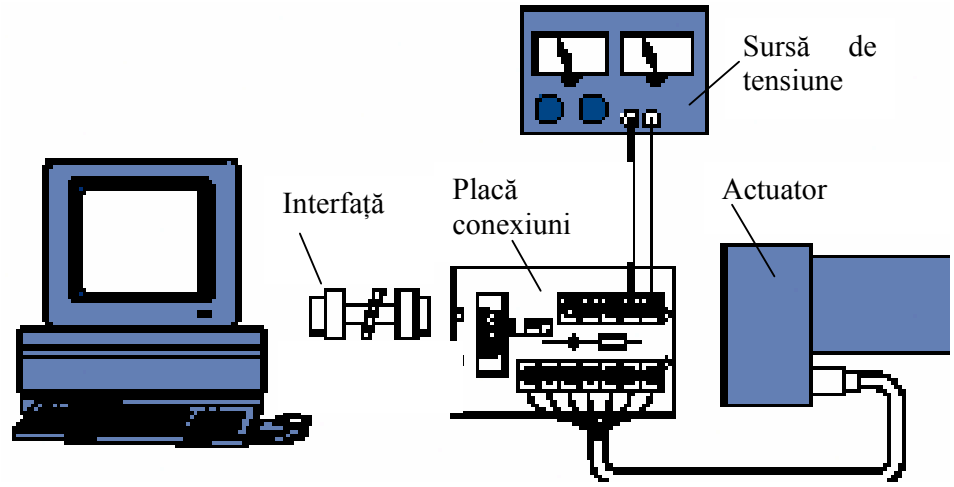

Fig.1

Placa de conexiuni cuprinde componente care realizează funcții de amplificatoare finale pentru alimentarea actuatorului în conformitate cu programal impus.

Circuitul IR8200B, circuitul integrat L298, L293 (fig.2), LMD18201 sunt câteva dintre circuitele utilizate în astfel de aplicații.

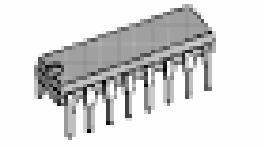

Fig.2

Schema electrică cu evidențierea posibilităților de utilizare ale circuitului L293B este prezentată în figura 3.

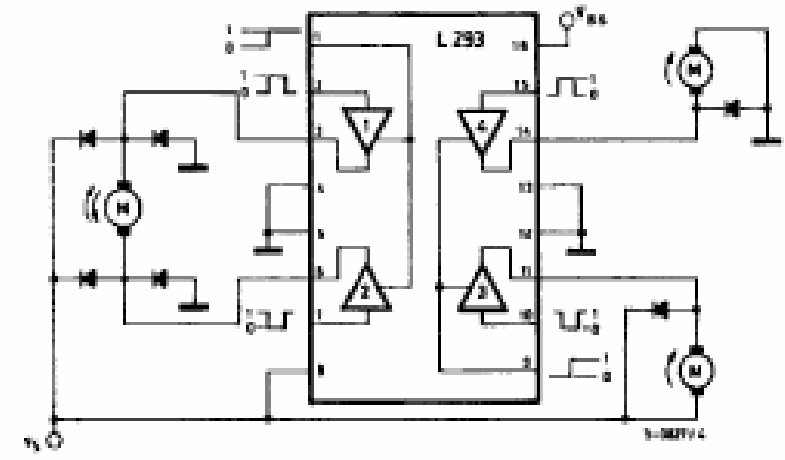

Fig.3

Circuitul are posibilitatea asigurării la ieşire a unui curent între 600mA şi 1A şi alimentarea / comanda a trei motoare: două în acelaşi sens iar unul în ambele sensuri.

O variantă nouă de control a aplicațiilor mecatronice este utilizarea portului paralel al unui PC. Schema de utilizare a portului paralel este ilustrată în figura 4 pentru o sarcină oarecare.

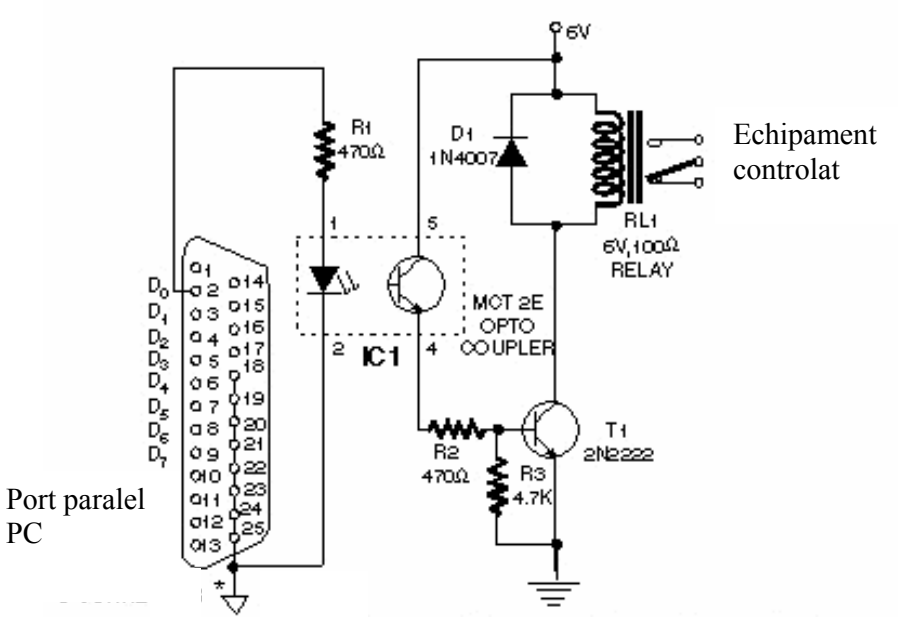

Fig.4

Interfața ilustrată în figura 4 permite controlul unei singure sarcini prin bitul D_0 de pe pinul 2 al portului paralel cu 25 de pini. Circuite identice pentru biti $D_1 – D_7$ disponibili pe pinii 3 – 9 se realizează în mod asemănător. Optocuplorul IC asigură o separare galvanică între PC şi circuitul controlat. S

Schema electrică principială a unui optocuplor este prezentată în figura 5. Un optocuplor este compus din două părți: o sursă de lumină (de ex. o diodă fotoemisivă) şi un detector (de ex. un fototranzistor sau fototiristor). Cea mai răspândită soluție constă în capsularea unei diode fotoemisive de GaAs cu un fototranzistor. Rezistența de limitare se alege astfel încât să nu se depăşească curentul maxim prin diodă. Dacă curentul prin diodă este cunoscut, se poate calcula curentul prin fototranzistor.

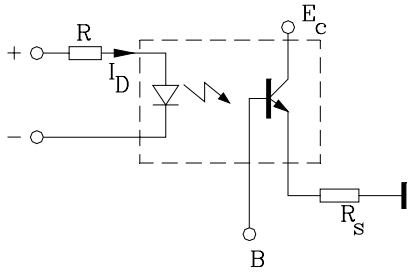

Fig.5

În funcție de semnalul de pe fotodiodă, tranzistorul se va afla în stare de conducție sau blocare. Se poate realiza în acest mod un amplificator din clasa B pentru alimentarea motorului electric.

Un rol important îl constituie partea de software care asigură conlucrarea dintre componentele sistemului.

Programul de lucru pentru comanda motorului prin portul paralel este prezentat în ANEXA 1.

Programul de lucru pentru comanda motorului de curent continuu cu utilizarea circuitului L293 este prezentat în ANEXA 2.

3. Mersul lucrării

În figura 6 se prezintă standul experimental pentru comanda mai multor micromotoare de curent continuu.

3.1. Comanda micromotorului prin portul paralel al unui PC

- Se identifică elementele componente ale standului; se realizează alimentarea circuitului de la o sursă de c.c. de valoare U = 5 V
- Se conectează un osciloscop pe alimentarea motorului comandat (motorul central al standului);
- Se lansează programul de lucru SA.exe (anexa 1);
- Se analizează facilităţile oferite de program;
- Se vizualizează forma tensiunii de alimentare la diverse valori a duratei de conectare; se consemnează concluziile în referat.
- Se vizualizează forma semnalului în tensiune la reversarea mişcării; se consemnează concluzile în cadrul referatului.

3.2. Comanda micromotorului prin intermediul unui PC şi circuitul 293

- Se identifică problema de analizat; se menţine alimentarea circuitului la o tensiune de 5 V;
- Se lansează programul de lucru ***Qbasic.exe \ parallel.bas \ RUN***
- Se analizează facilităţile oferite de program;
- Se comandă cele trei motoare; se vizualizează semnalizările oferite de LED-urile ataşate circuitului; se consemnează starea bitului pe fiecare pin la funcţionarea fiecărui motor.

ANEXA 1

COMANDA MOTORULUI DE C.C. PRIN PORTUL PARALEL

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

void pwm(int, int, int, int);
void main(void){
        int BASEADDR;
        int PORTA, PORTB, PORTC;
        int CNTRL;
        int DutyCycle;
        int SpeedOption;
        int Enable, Phase;

        clrscr ();
        window (5,5,75,30);
   /*   gotoxy(1,1); cprintf("Enter base addres (decimal e.g.) 608 =>\n");
        gotoxy(42,1); scanf("%d", &BASEADDR);
   */
        BASEADDR=888;
        PORTA=BASEADDR;
        PORTB=BASEADDR+1;
        PORTC=BASEADDR+2;
        CNTRL=BASEADDR+3;

        outportb(CNTRL, 128);
        outportb(PORTA,0);
        gotoxy(77,25); cprintf("BCM");
      gotoxy(1,1); cprintf("Universitatea Politehinca Timisoara, Faculatea de
Mecanica");
        gotoxy(1,3); cprintf("Optiuni:");
        gotoxy(1,4); cprintf("f - marire viteza");
        gotoxy(1,5); cprintf("s - micsorare viteza");
        gotoxy(1,6); cprintf("r - inversare directie");
        gotoxy(1,7); cprintf("q - iesire");
        gotoxy(1,9); cprintf("Selectie:");
        gotoxy(14,9);

        DutyCycle = 0;
        Enable=4;
        Phase=0;

        do {
                while (!kbhit()) {
                        pwm(PORTA, DutyCycle, Enable, Phase);
                }
```

```
                SpeedOption = getch();
                switch(SpeedOption){
                case 102:  /* crestere viteza */
                        DutyCycle=DutyCycle+2;
                        if (DutyCycle>100) {
                                DutyCycle=100;
                                gotoxy(1,20);  cprintf("Nu  se  poate  mai
repede!");
                                delay(200);
                                gotoxy(1,20); cprintf("                    ");
                        };
                        gotoxy(1,13); cprintf("Crestere viteza...\n");
                    /*  Enable=4+Phase;*/
                        pwm(PORTA, DutyCycle, Enable, Phase);
                        break;

        case 115:        /* scadere viteza */
                        DutyCycle=DutyCycle-2;
                        if(DutyCycle<0){
                                DutyCycle=0;
                                gotoxy(1,20); cprintf("Nu se poate mai incet!");
                                delay(200);
                                gotoxy(1,20); cprintf("                    ");
                        };
                        gotoxy(1,13);
                        cprintf("Scadere viteza...");
                        pwm(PORTA,DutyCycle,Enable,Phase);
                        break;
        case 114:
                    /*  Enable=4;    */
                        if (Phase==0) Phase=4;
                        else if (Phase==4) Phase=0;
                        pwm(PORTA,DutyCycle,Enable,Phase);
                        gotoxy(1,13); cprintf("Rotatie inversa...");
                        break;
        case 113:
                        gotoxy(1,20); cprintf("Program preluat si adaptat: prof.
Dolga Valer");
                        gotoxy(1,21); cprintf("                    stud. Bacican
Cristian");
                        gotoxy(1,24); cprintf("Pentru iesire apasati o tasta...");
                        outportb(PORTA, 0);
                        outportb(PORTB, 0);
                        outportb(PORTC, 0);
                        sound(600); delay(200); nosound();
                        sound(1000); delay(50); nosound();
                        while (!kbhit()){};
                        exit(0);
                        break;
                };
```

```
                gotoxy(14,9);
        }while(1);
};
void pwm(int PORTA, int DutyCycle, int Enable, int Phase) {
        int OnTime, OffTime, TotalTime;
        TotalTime=100;
        OnTime=(int)(TotalTime*DutyCycle/100);
        OffTime=(int)(TotalTime-OnTime);
        outportb(PORTA,32+Enable+Phase);
        delay(DutyCycle/*OnTime*/);
        if (Phase==4) Enable=-4;
        else Enable=0;
        outportb(PORTA,32+Enable+Phase);
        delay(OffTime);
        gotoxy(1,10); cprintf("Perioada de alimentare este: %3d%%",
DutyCycle);
        return;
};
```

ANEXA 2

COMANDA MOTORULUI DE C.C.
PE BAZA CIRCUITULUI L293

```
        CLS : SCREEN 2
KEY(1) ON: ON KEY(1) GOSUB FINIS
KEY(5) ON: ON KEY(5) GOSUB RETIRE
KEY(10) ON: ON KEY(10) GOSUB ALLON
PORT% = &H378
OUT PORT%, 0
LOCATE 8, 10: PRINT "<- ->"
V$ = STRING$(27, "Ű")
REM Ű obtained by pressing CTRL+ATL+2+1+9 (ASCII)
LOCATE 2, 6: PRINT "        U.P.T. Facultatea de Mecanica departamentul de Mecanica Fina"
LOCATE 22, 71: PRINT "BCM"
LOCATE 5, 6: PRINT V$; SPC(1); "CONTROL PANEL"; SPC(2); V$
LINE (40, 31)-(600, 180), 1, B
LINE (40, 40)-(600, 180), 1, B
LINE (40, 100)-(600, 120), 1, BF
LINE (140, 40)-(460, 110), 1, B
LOCATE 8, 65: PRINT "ON------Q"
LOCATE 12, 65: PRINT "OFF-----W"
LOCATE 17, 15: PRINT "F1"; SPC(24); "F5"; SPC(27); "F10"
LOCATE 19, 10: PRINT "EMERGENCY OFF"; SPC(17); "EXIT"; SPC(24); "ALL ON";
REM LOCATE 22, 10:
D$ = DATE$
J$ = LEFT$(D$, 2)
K$ = MID$(D$, 4, 2)
L$ = RIGHT$(D$, 4)
LOCATE 5, 7: PRINT SPC(1); K$; "-"; J$; L$; SPC(1); ""
STAT:
PSET (145, 85): DRAW "R20U10L20D10"
PSET (185, 85): DRAW "R20U10L20D10"
PSET (225, 85): DRAW "R20U10L20D10"
PSET (265, 85): DRAW "R20U10L20D10"
PSET (305, 85): DRAW "R20U10L20D10"
PSET (345, 85): DRAW "R20U10L20D10"
PSET (385, 85): DRAW "R20U10L20D10"
PSET (425, 85): DRAW "R20U10L20D10"
T$ = TIME$
Y$ = LEFT$(T$, 2)
Y = VAL(Y$)
IF Y < 12 THEN PP$ = "AM" ELSE PP$ = "PM"
IF Y > 12 THEN Y = Y - 12
U$ = MID$(T$, 3, 3)
LOCATE 5, 64: PRINT SPC(1); Y; U$; PP$; SPC(1); ""
```

```
LOCATE 9, 20: PRINT "1"; SPC(4); "2"; SPC(4); "3"; SPC(4); "4"; SPC(4); "5";
SPC(4); "6"; SPC(4); "7"; SPC(4); "8"
LOCATE 12, 19: PRINT AA; SPC(2); SS; SPC(2); DD; SPC(2); FF; SPC(2); GG;
SPC(2); HH; SPC(2); JJ; SPC(2); KK
X$ = INKEY$
X$ = RIGHT$(X$, 1)
N = INP(PORT%)
IF X$ = "K" THEN J = J - 40
IF X$ = "M" THEN J = J + 40
PSET (J + 105, 85): DRAW
"R20U10L20D10R2U10R2D10R2U10R2D10R2U10R2D10R2U10R2D10R2U10R2
D10"
FOR T = 1 TO 400: NEXT
PRESET (J + 105, 85): DRAW
"R20U10L20D10R2U10R2D10R2U10R2D10R2U10R2D10R2U10R2D10R2U10R2
D10"
IF J + 105 < 105 THEN J = 0
IF J >= 360 THEN J = 360
IF (J = 40) AND (X$ = "Q" OR X$ = "q") THEN GOSUB APPLE
IF (J = 40) AND (X$ = "W" OR X$ = "w") THEN GOSUB APPLEOF
IF (J = 80) AND (X$ = "Q" OR X$ = "q") THEN GOSUB BAT
IF (J = 80) AND (X$ = "W" OR X$ = "w") THEN GOSUB BATOF
IF (J = 120) AND (X$ = "Q" OR X$ = "q") THEN GOSUB TALE
IF (J = 120) AND (X$ = "W" OR X$ = "w") THEN GOSUB TALEOF
IF (J = 160) AND (X$ = "Q" OR X$ = "q") THEN GOSUB FLAT
IF (J = 160) AND (X$ = "W" OR X$ = "w") THEN GOSUB FLATOF
IF (J = 200) AND (X$ = "Q" OR X$ = "q") THEN GOSUB FAT
IF (J = 200) AND (X$ = "W" OR X$ = "w") THEN GOSUB FATOF
IF (J = 240) AND (X$ = "Q" OR X$ = "q") THEN GOSUB SILK
IF (J = 240) AND (X$ = "W" OR X$ = "w") THEN GOSUB SILKOF
IF (J = 280) AND (X$ = "Q" OR X$ = "q") THEN GOSUB SEVEN
IF (J = 280) AND (X$ = "W" OR X$ = "w") THEN GOSUB SEVENOF
IF (J = 320) AND (X$ = "Q" OR X$ = "q") THEN GOSUB LAST
IF (J = 320) AND (X$ = "W" OR X$ = "w") THEN GOSUB LASTOF
GOTO STAT '----ALL THE SUBROUTINES ARE BELOW-----
APPLE: SOUND 500, 2
AA = 1
LOCATE 6, 50
Q = 1 OR N
OUT PORT%, Q
RETURN
BAT: SOUND 500, 2
SS = 1
W = 2 OR N
OUT PORT%, W
RETURN
TALE: SOUND 500, 2
DD = 1
Q = 4 OR N
OUT PORT%, Q
```

```
RETURN
FLAT: SOUND 500, 2
FF = 1
Q = 8 OR N
OUT PORT%, Q
RETURN
FAT: SOUND 500, 2
GG = 1
Q = 16 OR N
OUT PORT%, Q
RETURN
SILK: SOUND 500, 2
HH = 1
Q = 32 OR N
OUT PORT%, Q
RETURN
SEVEN: SOUND 500, 2
JJ = 1
Q = 64 OR N
OUT PORT%, Q
RETURN
LAST: SOUND 500, 2
KK = 1
Q = 128 OR N
OUT PORT%, Q
RETURN
TALEOF: SOUND 400, 1
IF DD = 0 THEN RETURN
DD = 0
IF N = 4 THEN P = 0
IF N < 4 THEN P = N
IF N > 4 THEN P = N - 4
OUT PORT%, P
RETURN
APPLEOF: SOUND 400, 1
IF AA = 0 THEN RETURN
AA = 0
IF N = 1 THEN I = 0
IF N > 1 THEN I = N - 1
OUT PORT%, I
RETURN
BATOF: SOUND 400, 1
IF SS = 0 THEN RETURN
SS = 0
IF N = 2 THEN U = 0
IF N > 2 THEN U = N - 2
IF N < 2 THEN U = N
OUT PORT%, U
RETURN
FLATOF: SOUND 400, 1
```

```
IF FF = 0 THEN RETURN
FF = 0
IF N = 8 THEN E = 0
IF N < 8 THEN E = N
IF N > 8 THEN E = N - 8
OUT PORT%, E
RETURN
FATOF: SOUND 400, 1
IF GG = 0 THEN RETURN
GG = 0
IF N = 16 THEN Y = 0
IF N < 16 THEN Y = N
IF N > 16 THEN Y = N - 16
OUT PORT%, Y
RETURN
SILKOF: SOUND 400, 1
IF HH = 0 THEN RETURN
HH = 0
IF N = 32 THEN Y = 0
IF N < 32 THEN Y = N
IF N > 32 THEN Y = N - 32
OUT PORT%, Y
RETURN
SEVENOF: SOUND 400, 1
IF JJ = 0 THEN RETURN
JJ = 0
IF N = 64 THEN U = 0
IF N < 64 THEN U = N
IF N > 64 THEN U = N - 64
OUT PORT%, U
RETURN
LASTOF: SOUND 400, 1
IF KK = 0 THEN RETURN
KK = 0
IF N = 128 THEN z = 0
IF N < 128 THEN z = N
IF N > 128 THEN z = N - 128
OUT PORT%, z
RETURN
ALLON: SOUND 500, 4
OUT PORT%, 255
AA = 1: SS = 1: DD = 1: FF = 1: GG = 1: HH = 1: JJ = 1: KK = 1
RETURN
FINIS: SOUND 400, 2
OUT PORT%, 0
AA = 0: SS = 0: DD = 0: FF = 0: GG = 0: HH = 0: JJ = 0: KK = 0
RETURN
RETIRE:
OUT PORT%, 0
END
```