Aplicaţii

1) Afişare multiplexată pe 4 celule

Schema electronică arată ca în figura de mai jos:

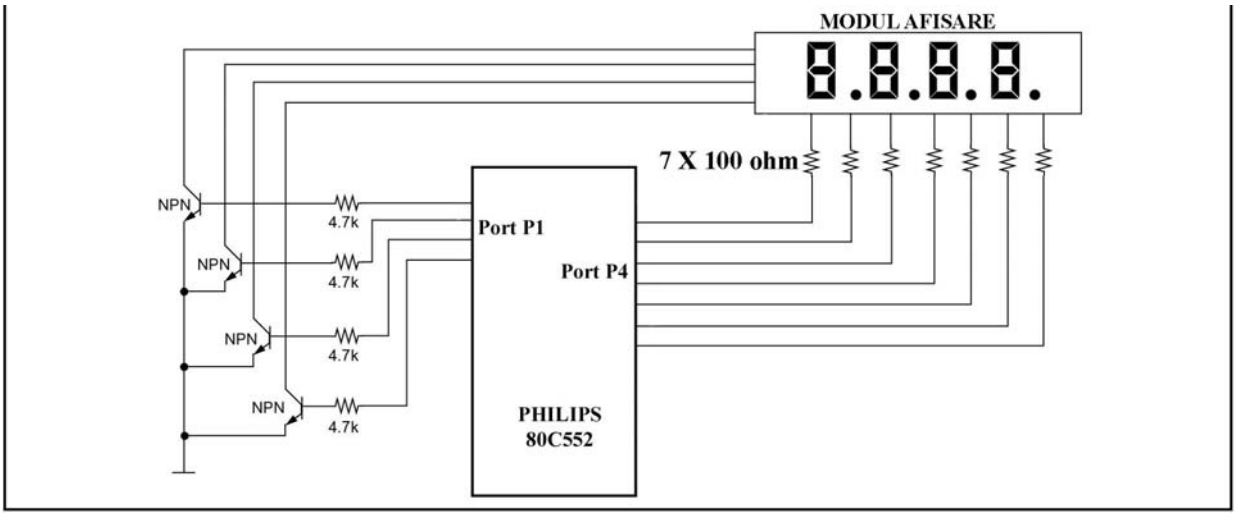


Fig. 1

Afişarea multiplexată se impune ca o necesitate în lucrul cu microcontrolerul. In caz contrar, ar trebui ca fiecare celulă de afişare să fie comandată de un port de ieşire al microcontrolerului. Treaba este simplă daca am porturi destule, însă de obicei nu am , lucru datorat şi faptului că, microcontrolerul, mai trebuie să execute şi alte funcţii de intrare/ieşire într-o aplicaţie dată. După cum se observă din Fig. 1, folosindu-mă doar de 2 porturi pot comanda până la 8 celule de afişare. (în cazul de faţă vom utiliza doar 4 celule).
Portul P4 aplică modulului de afişare cifra ce trebuie afişată (informaţia se aplică tuturor celulelor de afişare), iar portul P1 are rolul de a aprinde pe rând celulele de afişare. Ca urmare, din soft, voi fi „atenţi" să sincronizăm numărul celulei de afişare cu informaţia (cifra) ce trebuie afişată.

În cele ce urmează este prezentat un program de afişare multiplexată, cel mai simplu posibil – din acest motiv foarte rar se întâlneşte în practică, deoarece informaţia ce trebuie afişată se găseşte direct în formă „7 segmente", fără a mai fi nevoie de o translare din BCD în formă „7 segmente".
Funcţionare: din practică s-a observat că e bine ca fiecare modul de afişare să aibă un refresh cuprins între 40Hz şi 200Hz. Dacă este mai mic, atunci apare fenomenul de „pâlpâire" al cifrelor, iar daca este prea mare scade luminozitatea celulelor de afişare. Deci va trebui ca să baleem întregul afişor cu o frecvenţă (de exp.) 50Hz, adică să aprindem afişorul de 50 ori într-o secundă: asta conduce ca timpul pentru o singură baleere să fie de 1sec/50 = 20ms. Datorită faptului că a 4 celule de afişare: 20ms/4 = 5ms. Ca urmare voi aprinde prima celulă 5 ms, apoi o sting pe aceasta şi o aprind pe următoarea tot 5 ms ş.a.m.d. Intervalul de 5 ms îl voi obţine utilizând timerul/numărătorul T0 în lucru ca timer, astfel încât la fiecare întrerupere dată de timerul T0 voi „muta" spre stânga celula care trebuie iluminată.

```
name comanda_afisor_anod_comun_lab_8
ORG 0000H                   ;adresa de inceput al memoriei program
JMP START                   ;salt la eticheta "start"
ORG 000BH                   ;adresa vectorului de intrerupere data de Timerul T0
JMP INTER                   ;salt la eticheta "inter"

START:
        MOV P3,#00H         ;sterge portul P4
        MOV P1,#00H         ;sterge portul P1
        MOV TMOD,#01H       ;timerul T0 sa fie pe 16 biti (2 octeti)
        MOV TH0,#0ECH       ;incarca octetul superior al lui T0
        MOV TL0,#77H        ;incarca octetul inferior al lui T0
        MOV IE,#82H         ;activeaza intreruperile: globala si cea data de timerul T0
;                   gfedcba – caracterele 7 segmente.
        MOV R0,#11111001B   ;incarca registrul R0 (cifra 1)  //0F9H - unde este 0 =LED aprins – merg pe
        MOV R1,#10100100B   ;incarca registrul R1 (cifra 2)  //0A4H        // logica inversata, deoarece am
        MOV R2,#10110000B   ;incarca registrul R2 (cifra 3)  //0B0H        // afisor cu anod comun.
        MOV R3,#10011001B   ;incarca registrul R3 (cifra 4)  // 99H
        MOV P1,#01H         ;activeaza prima celula  (D1)
        MOV P3,R3           ;afiseaza cifra din prima celula
        CLR C               ;sterge flagul carry
        SETB TR0            ;porneste timerul T0
BUCLA:
        JMP BUCLA           ;stai in bucla si iesi de aici doar prin intrerupere

INTER:
        MOV A,P1            ;incarca in acumulator valoarea portului P1
        RL A                ;roteste stanga acumulatorul
        MOV P1,A            ;muta din acumulator inapoi in portul P1
        CJNE A,#10H,D2      ;daca nu esti in a 5-a celula, atunci sari
        MOV P1,#01H         ;activeaza prima celula
        MOV P3,R0           ;afiseaza prima celula
        MOV TH0,#0ECH       ;incarca octetul superior al lui T0
        MOV TL0,#77H        ;incarca octetul inferior al lui T0
        RETI                ;revenire din intrerupere
D2:
        CJNE A,#02H,D3      ;ma aflu in a 2-a celula ? daca da, atunci NU ma ramific
        MOV P3,R1           ;afiseaza valoarea registrului R2
        MOV TH0,#0ECH       ;incarca octetul superior al lui T0
        MOV TL0,#77H        ;incarca octetul inferior al lui T0
        RETI                ;revenire din intrerupere
D3:
        CJNE A,#04H,D4      ;ma aflu in a 3-a celula ? daca da, atunci NU ma ramific
        MOV P3,R2           ;afiseaza valoarea registrului R1
        MOV TH0,#0ECH       ;incarca octetul superior al lui T0
        MOV TL0,#77H        ;incarca octetul inferior al lui T0
        RETI                ;revenire din intrerupere
D4:
        MOV P3,R3           ;sigur ma aflu in celula D4; afiseaza registrul R0
        MOV TH0,#0ECH       ;incarca octetul superior al lui T0
        MOV TL0,#77H        ;incarca octetul inferior al lui T0
        RETI                ;revenire din intrerupere
END                         ;sfarsit
```

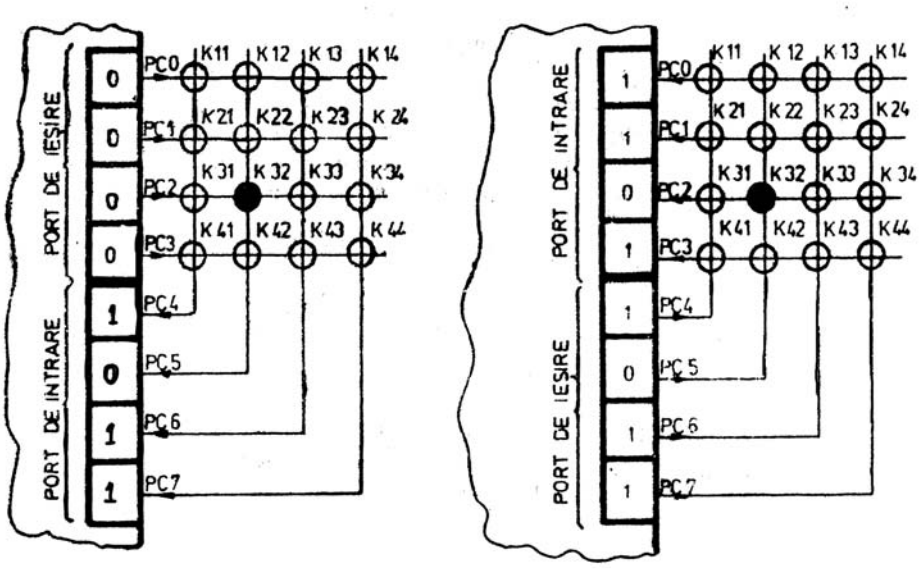
2) Baleere tastatură

Există două modalităţi de „citire” a unei tastaturi:
- prin metoda inversării liniilor şi coloanelor
- prin inspecţie secvenţială

Pentru a înţelege mai uşor fenomenele care apar la „citire”, vom neglija în cele ce urmează oscilaţiile mecanice ale butoanelor şi posibilitatea apăsării a două sau mai multe taste simultan.

Funcţionare metoda 1 (inversarea liniilor şi coloanelor)

Schema electrică este prezentată în Fig. 2.



Identificarea coordonatelor tastei apăsate prin metoda inversării liniilor si coloanelor

Fig. 2

Metoda constă din două etape: în prima etapă (a), „portul” la care sunt conectate liniile tastaturii este programat pentru ieşire iar cel la care sunt conectate coloanele este programat pentru intrare. În această etapă, în registrul de date al „portului” **de ieşire** se scrie peste tot valoarea logică 0. La toate intrările „portului” de intrare, cu excepţia celei corespunzătoarei coloanei pe care se află tasta apăsată se menţine însă valoarea logică 1. **Numai** pe coloana pe care se află tasta apăsată apare un semnal având valoarea logică 0. Astfel în prima etapă, pe baza informaţiei ce se citeşte din „portul” de intrare se identifică coloana pe care se află tasta apăsată (în exemplul ilustrat se află apăsată tasta K32). În cea de-a doua etapă se inversează funcţiile de intrare/ieşire ale celor două „porturi”: „portul” la care sunt conectate liniile, se reprogramează pentru intrare, iar cel la care sunt conectate coloanele – ca „port” de ieşire. Informaţia recepţionată în „portul” coloanelor, din etapa precedentă, este emisă acum pe coloanele matricii determinând recepţionarea valorii logice 0 în bitul „portului” de intrare la care este conectată linia pe care se află tasta apăsată. În toate celelalte poziţii ale „portului” liniilor se recepţionează valoarea logică 1, corespunzând tastelor neapăsate.
În acest moment avem coordonatele (line şi coloană) tastei apăsate.

Funcţionare metoda 2 (inspecţie secvenţială)

Schema electrică este prezentată în Fig. 3.

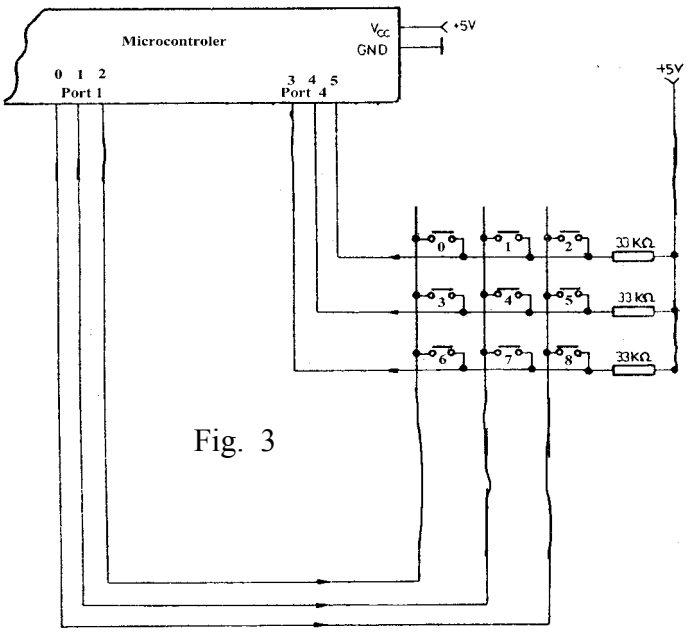


Fig. 3

Această metodă este mai simplă decât precedenta şi se justifică folosirea acesteia acolo unde am puţine taste, deoarece timpul de baleere (verificare dacă o tastă este apăsată) durează direct proporţional cu numărul acestora. După cum se vede şi din desenul alăturat, liniile sunt conectate la 1 logic. Ca urmare, voi pune pe rând coloanele la 0 logic si „citesc" nivelul logic al liniilor: dacă este 1 logic, tasta nu a fost apăsată ; dacă este 0 logic, tasta este apăsată. Chiar din acest moment cunosc coloana căreia i-am aplicat nivel 0 logic, iar linia îmi este dată de existenţa unui nivel 0 logic (datorită legăturii electrice dintre coloană şi linie).

Mai jos este prezentat schematic un soft de baleere de tastatură ce foloseşte această metodă.

```
call tastatura

tastatura:
        CLR P1.0            ;pune in 0 logic prima coloana
        JNB P4.3,TASTA6     ;daca pinul de port P4.3 este zero, atunci sari la eticheta TASTA6
        JNB P4.4,TASTA3     ;daca pinul de port P4.4 este zero, atunci sari la eticheta TASTA3
        JNB P4.5,TASTA0     ;daca pinul de port P4.5 este zero, atunci sari la eticheta TASTA0
        SETB P1.0           ;pune in 1 logic prima coloana

        CLR P1.1            ;pune in 0 logic a doua coloana
        JNB P4.3,TASTA7     ;daca pinul de port P4.3 este zero, atunci sari la eticheta TASTA7
        JNB P4.4,TASTA4     ;daca pinul de port P4.4 este zero, atunci sari la eticheta TASTA4
        JNB P4.5,TASTA1     ;daca pinul de port P4.5 este zero, atunci sari la eticheta TASTA1
        SETB P1.1           ;pune in 1 logic a doua coloana

        CLR P1.2            ;pune in 0 logic a treia coloana
        JNB P4.3,TASTA8     ;daca pinul de port P4.3 este zero, atunci sari la eticheta TASTA8
        JNB P4.4,TASTA5     ;daca pinul de port P4.4 este zero, atunci sari la eticheta TASTA5
        JNB P4.5,TASTA2     ;daca pinul de port P4.5 este zero, atunci sari la eticheta TASTA2
        SETB P1.2           ;pune in 1 logic prima coloana
ret

tasta6: setb P4.0           ;exemplu de functie ce poate fi executata la apasarea tastei 6
        call delay          ;(aprind si sting un LED in functie de intarzierea data de "delay")
        clr P4.0
        call delay
        ret
```