REALIZAREA UNUI SISTEM CU COMANDA PROGRAMATĂ . EXEMPLU

SCOPUL LUCRĂRII

Parcurgerea etapelor de proiectare HW şi SW a unui sistem cu comandă programată pornind de la funcţiile impuse acestuia.

1. DESCRIEREA PROGRAMULUI DE SIMULARE

1.1.Prezentare generală

Programul simuleaza funcţionarea unui sistem cu comandă programată realizat cu următoarele blocuri funcţionale:

a.- unitatea centrală (UCP): microprocesor, memorie de programe (ROM), memorie de date (RAM), blocul de control al subsistemelor de intrare/ieşire;

b.- porturi de intrare/ieşire (pentru a realiza interfata dintre echipamentele de intrare/iesire şi unitatea centrală):

- PLED (intrare/ieşire);
- PIO (intrare/ieşire);
- PEDAF0, PEDAF1 (ieşire);
- CTC (contor de timp programabil);

c.- dispozitive şi echipamente de intrare/ieşire (dispozitive comandate, prin program, de către UCP prin intermediul porturilor de intrare/ieşire):

- 8 diode LED conectate la portul PLED (L7...L0);
- 2 afişoare pe 7 segmente conectate la portul PEDAF0,respectiv PEDAF1;
- 8 comutatoare (D7...D0) conectate la portul PIO
- 1 comutator (STB) pentru generarea întreruperilor asincrone (de la portul PIO);
- 1 diodă LED (LINE) conectată la portul PIO;
- un "terminal" pentru afişarea de mesaje alfanumerice ("DISPLAY") "conectat" cu unitatea centrală

Acest terminal ("DISPLAY") permite afişarea a 8 linii fiecare cu câte 80 caractere; după scrierea celor 8 linii, afişajul terminalului se şterge în mod automat pentru a permite afişarea liniilor următoare (vezi procedurile PRINT , PRINTH , PRINTM).

Circuitul CTC (contor de timp programabil) este utilizat pentru generarea întreruperilor periodice către unitatea centrală.

In figura 1 este prezentată schema bloc a sistemului cu comandă programată simulat. Această schemă bloc este afişata pe ecranul calculatorului în momentul apelării programului de simulare. În urma execuţiei unor programe de aplicaţie, starea diodelor LED L7...L0 şi a comutatoarelor D7...D0 este afişată pe ecran; comutatoarele D7...D0, STB pot fi acţionate în orice moment (acţionarea lor fiind analoagă cu cea a unor comutatoare reale) de la tastele F1 - F9 după cum urmează:

F1=D7; F2=D6; F3=D5; F4=D4; F5=D3; F6=D2; F7=D1; F8=D0;F9=STB
(stânga-'0' logic; dreapta-'1' logic).

Starea LED-urilor, afişoarelor DAF0, DAF1, LINE poate fi modificată prin executarea unor programe de aplicaţii ce conţin instrucţiuni corespunzătoare de lucru cu porturile PLED, PEDAF0, PEDAF1, PIO definite în cadrul programului de simulare.

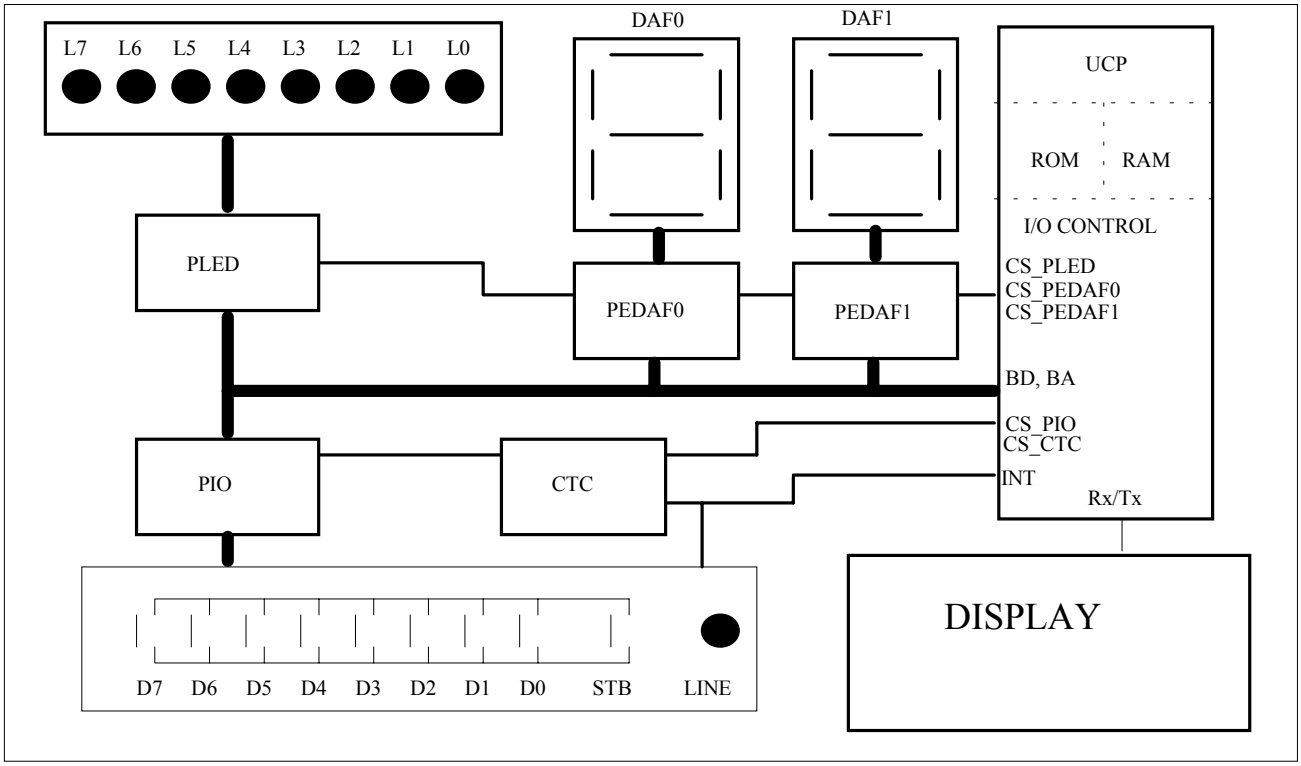


Figura 1. Schema bloc a SCP simulat

1.2.Structura programului de simulare

Programul de simulare are următoarele module (subprograme) ilustrate în figura 2 .

1.- subprogram de iniţializare (SCP_INIT) :

- permite iniţializarea variabilelor implicite şi eventual declararea şi iniţializarea altor variabile decât cele implicite;
- asigură legatura (interfaţa) între subprogramul de aplicaţie (dorit de utilizator) şi subprogramul de grafică ce actualizează starea sistemului cu comandă programată.

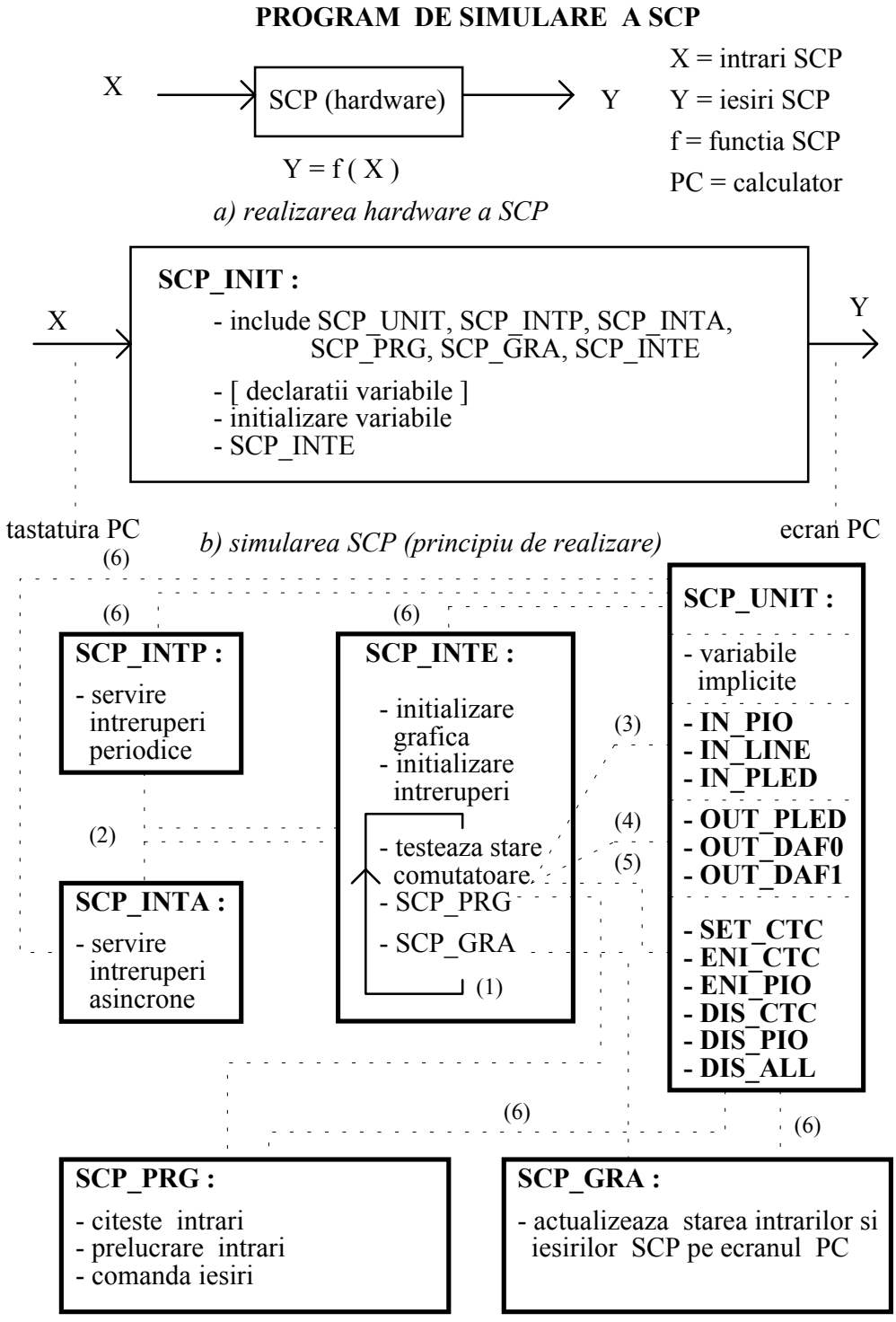
Acest subprogram (SCP_INIT) trebuie lansat în execuţie numai după scrierea subprogramului de aplicaţie.El constituie punctul de lansare în execuţie a programului de simulare a SCP.

2.- subprograme de aplicaţie (SCP_PRG, SCP_INTP, SCP_INTA):

Reprezintă proceduri care conţin aplicaţia dorită de utilizator.Aceste subprograme pot utiliza orice instrucţiune PASCAL şi de asemenea, procedurile definite pentru lucrul cu subsistemul de intrare/ieşire:

- SCP_PRG - aplicaţia (fără tratarea întreruperilor);
- SCP_INTP - rutina de tratare a întreruperilor periodice;
- SCP_INTA - rutina de tratare a întreruperilor asincrone.

Toate aceste subprograme sunt scrise utilizând proceduri PASCAL.



c) subprogramele programului de simulare a SCP

(1) - iesire din bucla cu ESC (2) - intreruperile trebuie validate in SCP_PRC
(3) - instructiuni de intrare (4) - instructiuni de iesire
(5) - instructiuni de lucru cu intreruperi
(6) - SCP_INTE, SCP_GRA, SCP_PRG, SCP_INTP, SCP_INTA, utilizeaza variabile, functii si proceduri definite in SCP_UNIT

Figura 2. Structura simulatorului

3.- colecţie de proceduri şi funcţii pentru lucrul cu subsistemul de intrare/ieşire al sistemului cu comandă programată (SCP_UNIT):

- conține declarația variabilelor implicite:
  a. - contori: i0 - i9 de tip întreg;
  b. - variabile: a0 - a9 de tip întreg;
  c. - registre: z[i,j] cu i,j = 0..49 ,de tip întreg.

Se recomandă a fi utilizate variabilele implicite .

- conține următoarele **funcţii si proceduri de lucru cu echipamentele de intrare/ieşire:**

- **funcţii:**

a.- funcția IN_PIO: - se citeşte starea comutatoarelor D0...D7 şi se întoarce un număr întreg (comutator acționat - 1);

b.- funcția IN_PLED: - se citeşte starea LED-urilor L0...L7 şi se întoarce un număr întreg (LED aprins - 1);

c.- funcția IN_LINE: - se citeşte starea LED-ului LINE şi se întoarce un număr întreg (LED-ul LINE aprins -1);

- proceduri:

a.- CLS - şterge DISPLAY
- apel: CLS;

b.- PRINTM- afişeaza un mesaj pe DISPLAY
- apel: PRINTM ('MESAJ', 0/1)
0/1: se alege 0 dacă nu se trece la
linie nouă şi 1 dacă se trece la
linie nouă;

c.- PRINT - afişeaza o variabilă de tip întreg pe DISPLAY în zecimal
- apel: PRINT (v,0/1)
0/1: are semnificația de la pct.b;

d.- PRINTH- afişeaza o variabilă în hexazeci mal pe DISPLAY
- apel: PRINTH (v,0/1);
0/1: are semnificația de la pct.b.

Pentru toate procedurile de afişare pe DISPLAY ,variabila din apel va fi afişată începând din poziția cea mai din stânga a "ecranului".

Opțiunea 0 în apelul acestor proceduri poate fi utilizată numai pentru o singură linie afişată pe tot "ecranul" dacă variabila de afişat nu se modifică.

e.- OUT_PLED- "aprinde" LED-urile conectate la
portul PLED (corespunzător parametrului procedurii)
- apel: OUT_PLED (n), unde n este întreg;

f.- OUT_DAF0- "aprinde" segmentele afişorului conectat la portul PEDAF0

(corespunzător parametrului procedurii)
- apel:OUT_DAF0(n);

g.- OUT_DAF1- "aprinde" segmentele afişoruluiconectat la portul PEDAF1
(corespunzător parametrului procedurii)
- apel:OUT_DAF1(n);

h.- OUT_LINE- "aprinde" LED-ul LINE corespunzător cu valoarea parametrului
- apel:OUT_LINE(n);

Observaţie: "aprinde" semnifică faptul că LED-urile corespunzătoare vor fi aprinse la următorul apel al subprogramului de grafică SCP_GRA.

i.- SET_CTC - setează constanta de timp pentru generarea întreruperilor periodice
- apel: SET_CTC (m)
Ex.: pentru m=20 se generează întreruperi periodice cu T=1sec.

j.- ENI_CTC - validează întreruperile de la CTC
- apel: ENI_CTC;

k.- ENI_PIO - validează întreruperile de la PIO
- apel: ENI_PIO;

l.- DIS_CTC - invalidează întreruperile de la CTC
- apel: DIS_CTC;

m.- DIS_PIO - invalidează întreruperile de la PIO
- apel: DIS_PIO;

n.- DIS_ALL - invalidează întreruperile de la CTC şi de la PIO.

4.- subprogram de afi**şare grafica SCP_GRA:**

- asigură ştergerea blocului funcţional DISPLAY la apăsarea tastei 'spaţiu';
- asigură actualizarea ecranului calculatorului (starea SCP): comutatoarele D0-D7, LED-urile L0-L7, LINE, cele două afişoare,DISPLAY;
- asigură controlul grafic asupra semnalelor : CS_PLED,CS_PEDAF0/1, CS_PIO, CS_CTC, INT, în modul următor:
  - stare inactivă a semnalului - culoare "darkgray";
  - stare activă a semnalului - culoare "white".

Semnificaţiile semnalelor sunt:

- CS_PLED - validare port PLED;
- CS_PEDAF0/1 - validare port PEDAF0/1;
- CS_PIO - validare PIO;
- CS_CTC - validare CTC;
- INT - tratarea unei cereri de întrerupere

În acest mod se pune în evidență ori de câte ori se execută o operație de intrare-ieşire sau o cerere de întrerupere.

5.- subprogramul SCP_INTE:

- asigură legatura între programele de aplicație şi programul grafic care actualizează starea sistemului cu comandă programată pe ecranul calculatorului;
- citeşte tastatura pentru a modifica în mod corespunzător starea comutatoarelor D7-D0; codul oricărei taste acționate de la tastatura calculatorului este memorat în variabila globală "display" de tip caracter.În momentul în care o tastă a fost actionată, codul acesteia este preluat de către calculator prin servirea unei cereri de întrerupere (generate de tastatură) şi depus într-o locație de memorie; conținutul acestei locații este citit de subprogramul SCP_INTE. Această modalitate de citire a tastaturii permite simularea acționării comutatoarelor prin acționarea unor taste (deoarece nu se aşteaptă apăsarea unei taste de la claviatura calculatorului).
- asigură ieşirea din programul de simulare la acționarea tastei "ESC".

Acest subprogram conține o buclă în care se execută pe rând procedura de aplicație SCP_PRG şi procedura de afişare grafică SCP_GRA. Din această cauză aplicația SCP_PRG (SCP_INTP, SCP_INTA) nu trebuie să conțină bucle.

1.3.Utilizarea programului de simulare

Următoarele subprograme trebuie modificate pentru a realiza o aplicație:

- SCP_INIT → inițializarea variabilelor;
- SCP_PRG (SCP_INTP, SCP_INTA - dacă există întreruperi)→ pentru a defini aplicația.

După realizarea modificărilor în aceste subprograme, se lansează în execuție SCP_INIT care prin intermediul subprogramului SCP_INTE va executa aplicația
SCP_PRG şi va actualiza starea SCP pe ecranul calculatorului. Dacă s-au programat întreruperi, se vor executa subrutinele specifice de tratare a întreruperilor; şi în acest caz, starea SCP se va actualiza pe ecran.
În continuare este prezentat fiecare subprogram de aplicație :

Subprogramul SCP_INIT:

```
{$i g:\l_scp\scp_unit}          {include variabilele , funcțiile şi procedurile predefinite}
{$i g:\l_scp\lscp1\scp_intp}  {procedura de tratare a întreruperilor periodice}
{$i g:\l_scp\lscp1\scp_inta}   {procedura de tratare a întreruperilor asincrone}
{$i g:\l_scp\scp_gra}           {procedura de actualizare grafică a ecranului}
{$i g:\l_scp\lscp1\scp_prg}   {procedura aplicație utilizator}
{$i g:\l_scp\scp_inte}          {procedura de interfață HW - calculator}
{declaratii variabile,funcții,proceduri suplimentare}
begin
{inițializari variabile}
scp_inte;                {apel program aplicație , actualizare grafică , stare HW}
end.
```

Subprogramul SCP_PRG:

```
procedure scp_prg;
begin
```

{instrucţiuni PASCAL şi instucţiuni de intrare - ieşire definite anterior}
end;

Subprogramul SCP_INTA:

```
procedure scp_inta;
begin
{instructiuni PASCAL si instuctiuni de intrare - ieşire definite anterior}
end;
```

Subprogramul SCP_INTP:

```
procedure scp_intp;
begin
{instructiuni PASCAL si instuctiuni de intrare - ieşire definite anterior} end;
```

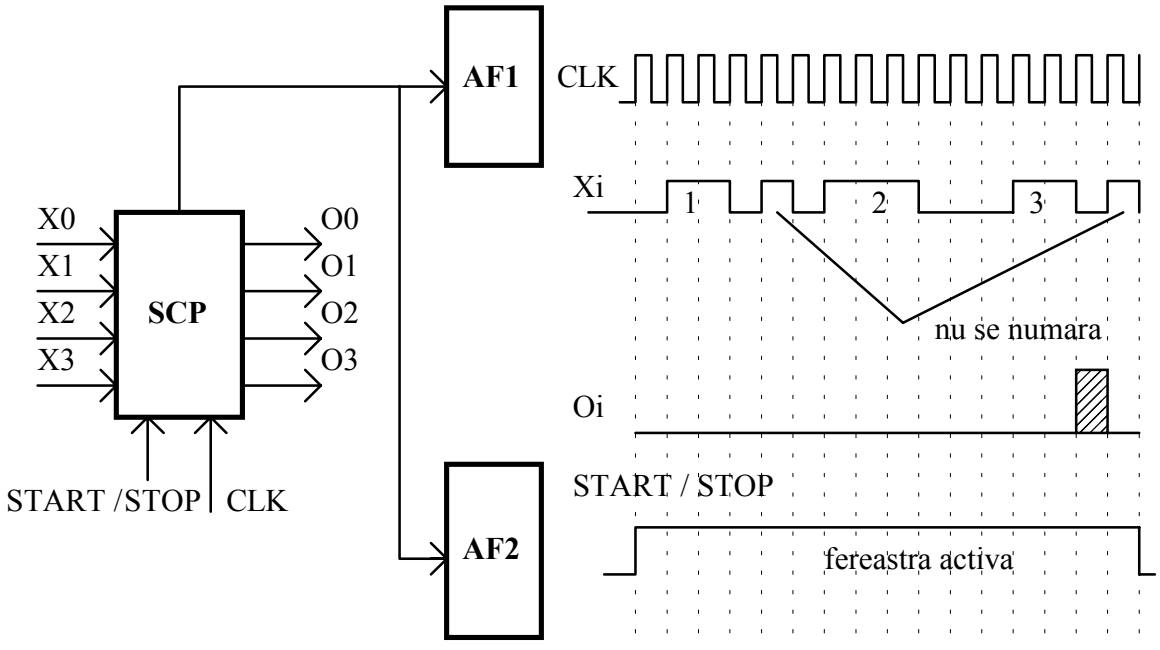
Toate liniile de program subliniate trebuie introduse de către utilizator; celelalte linii există şi nu trebuie şterse.

2. DESCRIEREA FUNCŢIILOR SISTEMULUI CU COMANDĂ PROGRAMATĂ

Să se realizeze un sistem cu comandă programată cu următoarele funcţii:

- citeşte intrările X0, X1, X2, X3;
- contorizează toate impulsurile cu durata mai mare de 2sec. care apar pe cele patru intrări;
- dacă pe o intrare apar mai mult de trei impulsuri cu durata mai mare decât 2sec. , pe ieşirea Oi se generează impulsuri periodice cu durata de 1 sec.;
- citirea intrărilor se va face pe durata unei ferestre temporale (marcate de semnalul START/STOP);
- la expirarea ferestrei temporale active se afişează, pe două dispozitive alfanumerice, numărul maxim, respectiv numărul minim de impulsuri care au apărut pe intrările Xi.

Schema sistemului cu comandă programată este prezentată în fig.3.



Pulsul hasurat indica faptul ca Oi este in mod repetitiv 1 si 0 logic

Figura.3. Schema sistemului cu comandă programată

Notații: CLK = ceas de citire a intrărilor Xi (pe frontul pozitiv sau negativ)

Observație: un impuls neterminat la sfârşitul ferestrei temporale, se consideră încheiat.

Se cere:

- schema hardware a sistemului cu comandă programată;
- organigrama programului de comandă.

3. STRUCTURA HARDWARE

Structura sistemului cu comandă programată trebuie să conțină, în conformitate cu funcțiile impuse, următoarele elemente:

- un port de intrare (pentru citirea intrărilor Xi);
- un port de ieşire (pentru generarea ieşirilor Oi);
- un controler de întreruperi, dacă programul de comandă utilizează întreruperi;
- două porturi de ieşire pentru cele două afişoare;
- unitatea centrală de prelucrare (microprocesorul şi circuitele sale auxiliare);
- memoria asociată programului şi datelor utilizate de program.

Rezultă următoarea schemă bloc generală a sistemului cu comandă programată (figura 4):

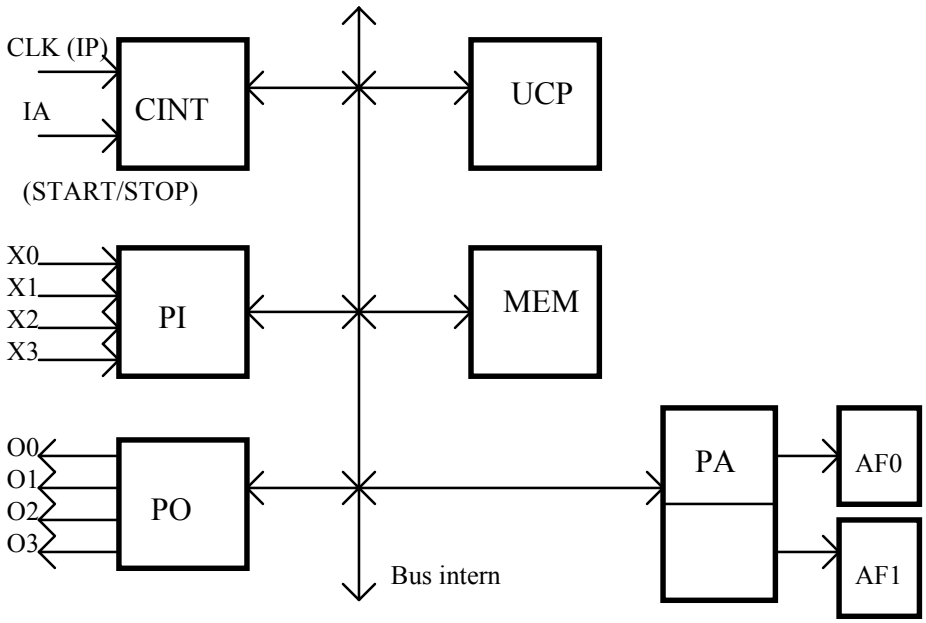


Figura 4. Schema bloc a SCP

Notații: CINT = controler de întreruperi
IP = întrerupere periodică
IA = întrerupere asincronă

PI = port de intrare (pentru X0,X1,X2,X3)
PO = port de ieşire (pentru O0,O1,O2,O3)
UCP = unitatea centrală de prelucrare
MEM = memorie
PA = porturi de ieşire pentru afişoare
AF0,AF1 = afişoare

Se presupune că se lucrează cu două nivele de întreruperi:

- întreruperi periodice (de la CLK), pentru măsurarea duratelor de timp;
- întreruperi asincrone pentru generarea semnalelor START/STOP.

4. REALIZAREA PROGRAMULUI DE COMANDĂ

4.1. Definirea proceselor

Se definesc procesele:

- P0,P1,P2,P3 - procese care citesc respectiv intrările X0,X1,X2,X3;
- P4 - proces care testează apariția unui număr de impulsuri mai mare decât 3, pe linia de intrare i (i=0,1,2,3) şi generează în mod corespunzător ieşirea Oi;
- P5 - procesul de calcul al minimului şi maximului numărului de impulsuri cu durata mai mare de 2 sec. care au apărut pe liniile X0,X1,X2,X3.

Cele şase procese definite au grafurile hibride de tranziții prezentate în figurile 3,4 şi 5.Notațiile utilizate sunt următoarele:

- CNTi - contor de timp pentru lățimea impulsului curent pe linia i (i=0,1,2,3);
- NRi - număr de impulsuri cu durata mai mare de 2 sec. pe linia i (i=0,1,2,3);

Perioada întreruperilor periodice este de 1 sec.

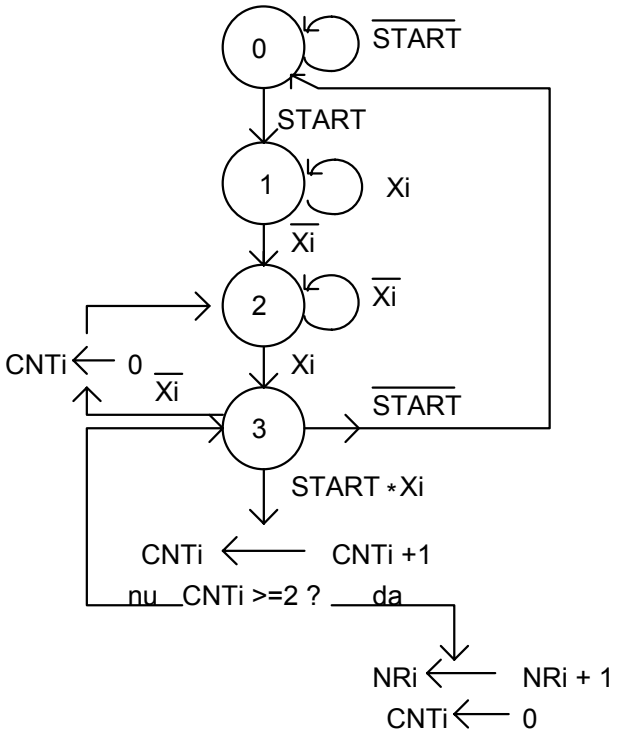


Figura.5. Graful hibrid de tranziții pentru procesele Pi (i=0,1,2,3)

Notații: START = fereastra temporală activă
/START = fereastra temporală inactivă
Xi = intrarea Xi=1
/Xi = intrarea Xi=0

Pentru procesul P4 se presupune că durata de timp în care fereastra temporală este inactivă, este de cel puțin 2sec., pentru a permite generarea corectă a semnalului pe linia Oi.

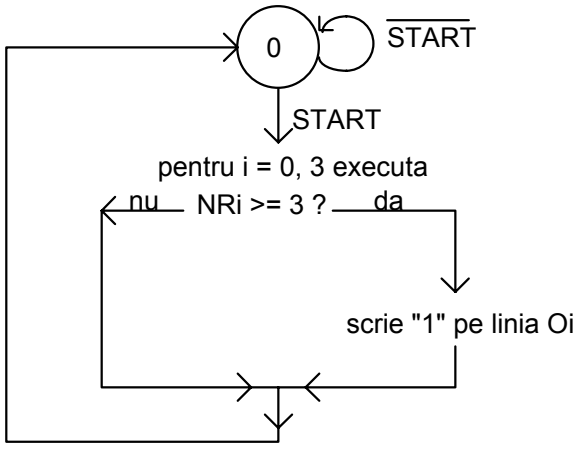


Figura 6. Graful hibrid de tranziții pentru procesul P4

Dacă NRi >= 3, se scrie "1" pe linia Oi timp de 1sec. (după 1sec. linia Oi trebuie adusă la zero).

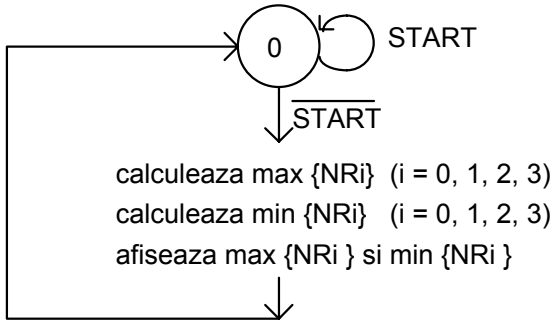


Figura.7. Graful hibrid de tranziții pentru procesul P5

4.2. Organigramele programului de comandă

Cele cinci procese (P0,P1,P2,P3,P4) vor fi executate pe rând de către un program de planificare cu diviziune uniformă în timp (se alocă câte o cuantă de timp pentru fiecare proces)

Organigrama planificatorului este următoarea:

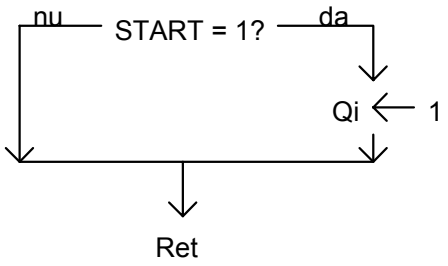
IP : CALL P0;
(1sec.) CALL P1;
(întreruperi CALL P2;
periodice) CALL P3;
CALL P4;
CALL P5;
Revenire în programul principal (RET);

În continuare sunt prezentate organigramele pentru fiecare proces în parte:

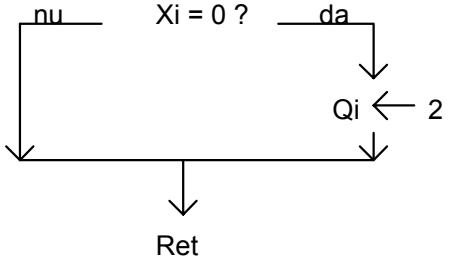
Procesele Pi(i=0,1,2,3) |

Selecteaza Qi dintre

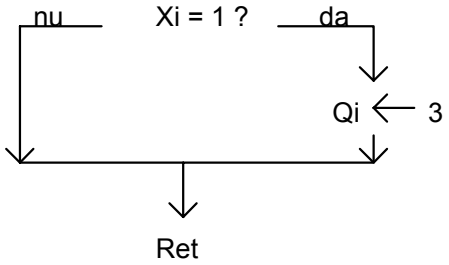
0:



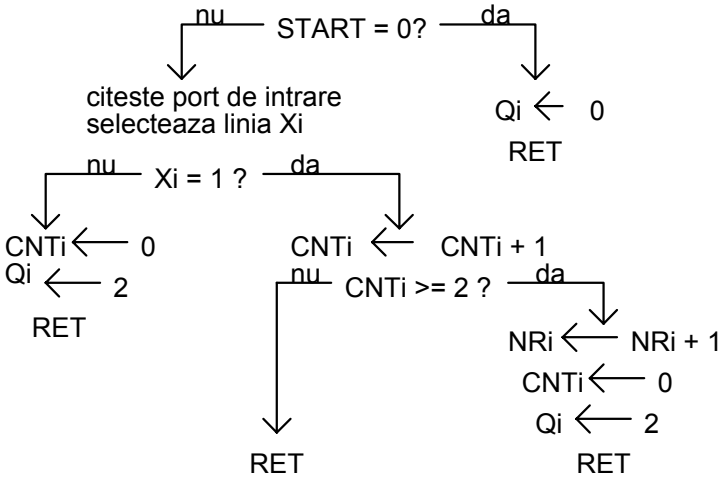
1: citeste port de intrare;
selecteaza linia Xi;



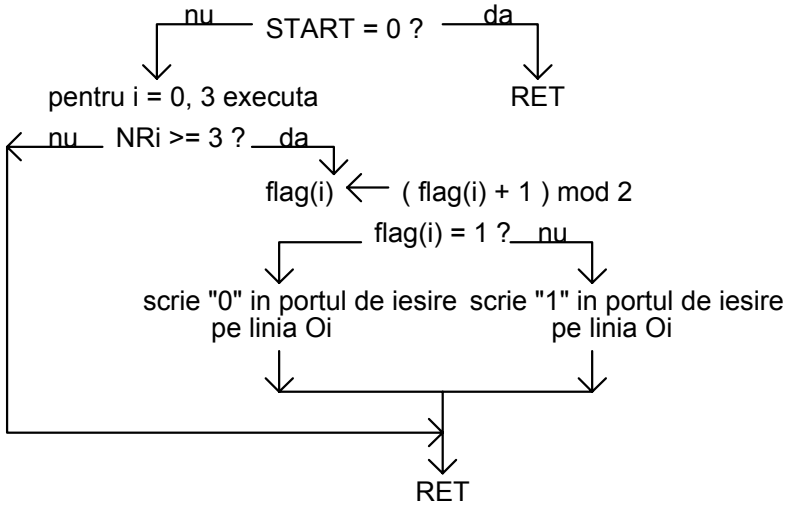
2:citeste port intrare;
selecteaza linia Xi



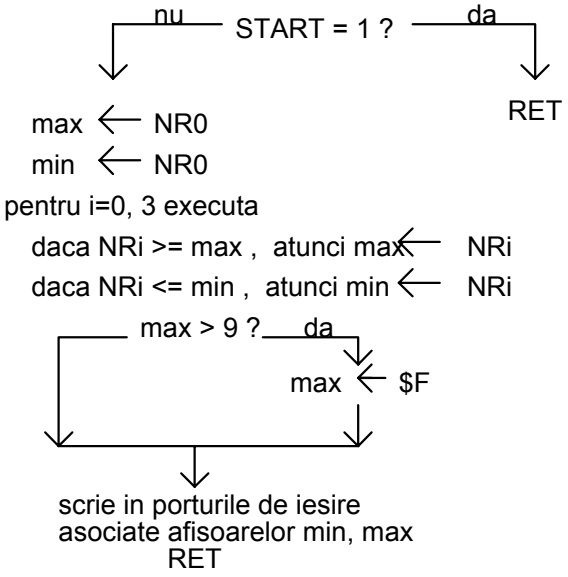
3:



Procesul P4



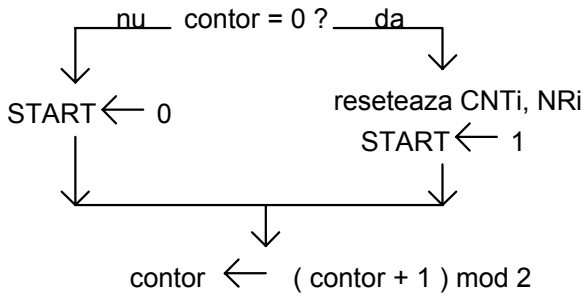
Procesul P5



Observații: Qi = starea procesului Pi (i=0,1,2,3)
START = variabilă care indică faptul că fereastra de timp este activă (START=1)
flg(i)= indică faptul că s-a scris "1" pe ieşirea Oi (flg(i)=1).Această variabilă este necesară pentru înscrierea pe linia Oi a unui impuls de 1sec.

Dacă numărul maxim de impulsuri cu o durată mai mare de 2sec. depăşeşte cifra 9, atunci se marchează acest fapt afişându-se cifra hexazecimala "F". Variabila START este setată în rutina de servire a întreruperii asincrone (IA):

IA :

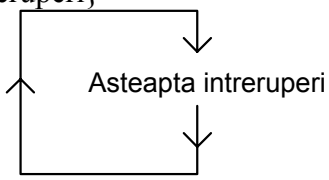


START comută între 0 şi 1 la fiecare întrerupere asincronă IA.

Programul principal are următoarea organigramă:

PP :

Q0=0; Q1=0; Q2=0; Q3=0;
contor=0; flg(0)=0; flg(1)=0; flg(2)=0; flg(3)=0
START=0; CNTi=0; NRi=0;
{initializari}
programare întreruperi periodice la 1sec.
validare întreruperi periodice
validare întreruperi asincrone
{programare întreruperi}



Programul este următorul:

SCP_INIT:

```
{$i  g:\l_scp\scp_unit}          {include variabilele , functiile si}
                                 {procedurile predefinite}

procedure proces(i:integer);

begin
case z[0,i] of

0: begin
  if i1=1 then z[0,i]:=1;{daca start este 1 trec in starea 1}
  end;

1: begin
  a1:=in_pio;{citesc portul de intrare}
  a2:=a1 and z[3,i];{selectez linia i}
  if a2=0 then z[0,i]:=2;{daca linia i este 1 trec in starea 2}
  end;
```

```
2: begin
  a1:=in_pio;  {citesc portul de intrare}
  a2:=a1 and z[3,i];{selectez linia i}
  if a2=z[3,i] then z[0,i]:=3;{daca linia i este 1 trec in starea 3}
  end;

3: begin
  if i1<>0 then begin {daca start este 1}
            a1:=in_pio;{citesc portul de intrare}
            a2:=a1 and z[3,i];{selectez linia i}
            if a2=z[3,i] then begin {daca linia i este 1 }
                  z[1,i]:=z[1,i]+1;{incrementez contorul de timp}
            if z[1,i] >=2 then begin
            z[2,i]:=z[2,i]+1;{incrementez contorul de inpulsuri}
            z[1,i]:=0;          {resetez contorul de timp}
            z[0,i]:=1;          {trec in starea 1}
            end;
            end;
            if a2=0 then begin {daca linia i este 0}
            z[1,i]:=0;{resetez contorul de timp}
            z[0,i]:=2;{trec in starea 2}
            end;
            end else z[0,i]:=0;{trec in starea 0}

  end;
end;    {end case}
end;    {end proces(i)}

procedure proces4(i:integer);

begin

if i1<>0 then begin {daca start este1}
  if z[2,i]>=3 then begin {daca numărul de inpulsuri >=3}
      z[4,i]:=(z[4,i]+1) mod 2;{flag pentru semnalizare intermitenta}
      if z[4,i]<>1 then begin  {daca flag este 0}
      a3:=in_pled;
      a4:=a3 or z[3,i];{selectez led-ul pentru intrarea i}
      out_pled(a4);{aprind led-ul corespunzator intrării i}
      end   else
      begin
      a3:=in_pled;
      a4:=a3 and (not z[3,i]);{selectez led-ul pentru intrarea i}
           out_pled(a4);     {sting led-ul corespunzator intrării i}
      end;
      end;
      end;
      end;

procedure proces5;
{calculeaza numărul max si min de impulsuri sosite pe intrări}
```

```
begin

if i1=0 then begin
a5:=z[2,0];  {max}
a6:=z[2,0];  {min}
if z[2,1]>=a5 then a5:=z[2,1];
if z[2,2]>=a5 then a5:=z[2,2];
if z[2,3]>=a5 then a5:=z[2,3];
if z[2,1]<=a6 then a6:=z[2,1]; if z[2,2]<=a6 then a6:=z[2,2];
if z[2,3]<=a6 then a6:=z[2,3];
if a5>9 then a5:=15;    {pentru un număr mai mare de 9 impulsuri afisez F}
out_daf0(z[9,a5]);      {afisez maxim}
out_daf1(z[9,a6]);      {afisez minim}
end;
end;

{$i g:\l_scp\lscp4\intp}    {procedura de tratare a întreruperilor}
                            {periodice}
{$i g:\l_scp\lscp4\inta}    {procedura de tratare a întreruperilor}
                            {asincrone}
{$i g:\l_scp\scp_gra}           {procedura de actualizare grafica a}
                            {ecranului}
{$i g:\l_scp\lscp4\prg}     {procedura aplicatie utilizator}
{$i g:\l_scp\scp_inte}          {procedura de interfata HW - calculator}

begin

{initializari variabile}
z[0,0]:=0;    { z[0,i] starea procesului i , cu i de la 0 la 3}
z[0,1]:=0;
z[0,2]:=0;
z[0,3]:=0;
i0:=0;       {contor întreruperi asincrone}
z[3,0]:=1;   {masti}
z[3,1]:=2;
z[3,2]:=4;
z[3,3]:=8;

z[1,0]:=0;z[1,1]:=0;z[1,2]:=0;z[1,3]:=0;  {contori de timp - CNTi}
z[2,0]:=0;z[2,1]:=0;z[2,2]:=0;z[2,3]:=0;  {numărator de impulsuri - NRi}
z[4,0]:=0;z[4,1]:=0;z[4,2]:=0;z[4,3]:=0;  {flag pentru afisare}
z[9,0]:=$3f;z[9,1]:=6;z[9,2]:=$5b;z[9,3]:=$4f;z[9,4]:=$66;
  {coduri cifre pentru DAF-uri}
z[9,5]:=$6d;z[9,6]:=$7d;z[9,7]:=7;z[9,8]:=$7f;z[9,9]:=$6f;
z[9,15]:=$71;
i1:=0;          {start}
scp_inte;                   {apel program aplicatie , actualizare}
end.
```

SCP_PRG

```
procedure scp_prg;
begin
set_ctc(20);
eni_ctc;
eni_pio;
out_pled(i1*128);       {afiseaza start}

printm(concat(chr(z[2,3]+$30),' ',chr(z[2,2]+$30),' ',
chr(z[2,1]+$30),' ',chr(z[2,0]+$30)),1);    {afiseaza NR3,NR2,NR1,NR0}
end;
```

SCP_INTP

```
procedure scp_intp;

begin
proces(0);
proces(1);
proces(2);
proces(3);
proces4(0);
proces4(1);
proces4(2);
proces4(3);
proces5;
end;
```

SCP_INTA

```
procedure scp_inta;
begin
if i0=0 then begin
   z[1,0]:=0;z[1,1]:=0;z[1,2]:=0;z[1,3]:=0;
   z[2,0]:=0;z[2,1]:=0;z[2,2]:=0;z[2,3]:=0;
   z[4,0]:=0;z[4,1]:=0;z[4,2]:=0;z[4,3]:=0;
   i1:=1;
   end
   else
   i1:=0;
   i0:=(i0+1) mod 2;
end;
```

5. Desfăşurarea lucrării

1. Să se analizeze grafurile de tranziții pentru fiecare proces definit anterior .
2. Să se execute programul şi să se urmarească funcționarea sa (conform specificațiilor impuse la punctul 2).

Temă : Să se realizeze o versiune de program fără a utiliza întreruperi sau planificare de procese (cu algoritm specializat).