# MODULUL 4
# Aplicații care integrează CLC, CLS și PS

Să se proiecteze un modul software (realizat ca proces secvențial PS) pentru blocul de afișare al unei centrale termice de apartament, cu următoarele specificații:

***Intrări***

| SW7 | SW6 | SW5 | SW4 | SW3 | SW2 | SW1 | SW0 |
|---|---|---|---|---|---|---|---|
| PRG | RST | AC | I | SET | SF | SG | SP |

***Ieşiri***

| LED7 | LED6 | LED5 | LED4 | LED3 | LED2 | LED1 | LED0 |
|---|---|---|---|---|---|---|---|
| AC | I | TempAC | | TempI | | Err | |

**SW7-SW3** reprezintă butoanele de comandă active în 0 cu revenire (apăsat și eliberat):

**PRG** – programare mod de lucru
**RST** – inițializare
**AC** – comandă apă caldă
**I** – comandă încălzire
**SET** – programare temperatură apă caldă și încălzire

**Modul de lucru** poate fi:

- Programare temperaturi pentru apă caldă sau încălzire
- Operare (mod apă caldă / mod încălzire și apă caldă)

**SW2-SW0** reprezintă senzori activi în 1, astfel:

**SF** – senzor fum
**SG** – senzor lipsă gaz
**SP** – senzor lipsă presiune apă

**Programarea temperaturii** pentru apa caldă (încălzire) se face astfel:

1. **PRG apăsat** – trecerea în mod programare
2. **AC/I apăsat** – selecție temperatură apă caldă sau încalzire
3. **SET apăsat** – modifica la fiecare apăsare temperatura (mică, medie, mare)
4. **PRG apăsat** – ieșire din modul de programare

Pentru **modul de operare:**

**AC apăsat** – furnizare AC și monitorizare erori
Se vor afișa temperatura AC și erorile
**I apăsat** – furnizare AC și I și monitorizare erori
Se vor afișa temperaturile AC și I și erorile

Din modul de operare se poate trece în modul de programare prin apăsarea butonului PRG.

Se poate comuta din modul AC în modul I prin apăsarea butonului I.

Se poate comuta din modul I în modul AC prin apăsarea butonului AC.

La apariția unei erori, sistemul se blochează. Se revine în funcționarea normală prin apăsarea butonului RST, care va fi semnalizat prin aprinderea scurtă și apoi stingerea LED7-LED0.

**LED7-LED0** se aprind pe 0 și reprezintă modul de lucru astfel:

| Afisare | | |
|---|---|---|
| **AC** | **I** | |
| 0 | 0 | mod programare |
| 0 | 1 | mod AC (I inactiv) |
| 1 | 0 | mod I(AC inactiv) |
| 1 | 1 | asteptare comanda |

| TempAC(TempI) | |
|---|---|
| 10 | temp mica |
| 01 | temp medie |
| 00 | temp mare |

| Err | |
|---|---|
| 11 | fara erori |
| 10 | eroare P |
| 01 | eroare G |
| 00 | eroare F sau orice combinatie |

Semnalizarea erorilor se va realiza printr-un CLC.
Temperaturile vor fi stabilite utilizând un CLS.

Se cer:
- Tabela de adevăr pentru CLC și determinarea măștilor necesare
- Graful asociat CLS
- Tabelele de semnale relevante pentru CLS și determinarea măștilor necesare
- Graful PS
- Codul în limbaj C pentru întregul modul software (CLC, CLS și PS), fără secvențele de inițializare a porturilor și a timer-ului.

Se va considera următoarea configurare:

**PORTD** – intrare
**PORTB** – ieșire
**TIMER0** – programat pe întreruperi periodice la 20ms.

# REZOLVARE

## Tabela de adevăr pentru CLC și determinarea măștilor necesare

Circuitul logic combinațional (CLC) va fi folosit pentru semnalizarea erorilor.

Erorile sunt generate de către cei trei senzori (fum, lipsă gaz, lipsă presiune apă), astfel că SW2-SW0 vor fi considerate intrări ale circuitului logic combinațional, ceea ce determină un CLC cu trei biți de intrare.

Erorile sunt semnalizate cu ajutorul LED1-LED0 și pot avea patru valori, conform cerinței:

11 – fără erori
10 – eroare lipsă presiune apă
01 – eroare lipsă gaz
00 – eroare fum sau orice altă combinație

Calcularea măștilor pentru fiecare bit de intrare în parte se poate realiza conform principiului prezentat în laborator. Masca specifică unui bit de intrare se obține prin atribuirea valorii 1 pe poziția bitului pentru care se calculează masca, respectiv 0 pe pozițiile celorlalți biți.

Datorită faptului că biții de intrare ai CLC sunt deja poziționați pe cei mai puțin seminificativi trei biți ai registrului de intrare, iar deplasările nu mai sunt necesare, este suficientă determinarea unei singure măști. Pentru demonstrație, se vor prezenta în continuare ambele variante:

Varianta 1 – rezolvarea cu calculul unei singure măști

| **INTRARE** | PRG | RST | AC | I | SET | ***SF*** | ***SG*** | ***SP*** |
|---|---|---|---|---|---|---|---|---|
| **MASCA** | 0 | 0 | 0 | 0 | 0 | ***1*** | ***1*** | ***1*** |
| **INDEX** | 0 | 0 | 0 | 0 | 0 | ***SF*** | ***SG*** | ***SP*** |

& **= 0x07**

Varianta 2 – rezolvarea cu calculul celor trei măști

| **INTRARE** | PRG | RST | AC | I | SET | ***SF*** | ***SG*** | ***SP*** |
|---|---|---|---|---|---|---|---|---|
| **MASCA_SF** | 0 | 0 | 0 | 0 | 0 | ***1*** | 0 | 0 |
| **MASCA_SG** | 0 | 0 | 0 | 0 | 0 | 0 | ***1*** | 0 |
| **MASCA_SP** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ***1*** |

= **0x04**
= **0x02**
= **0x01**

| **INTRARE** | PRG | RST | AC | I | SET | ***SF*** | ***SG*** | ***SP*** |
|---|---|---|---|---|---|---|---|---|
| **MASCA_SF** | 0 | 0 | 0 | 0 | 0 | ***1*** | 0 | 0 |
| **TMP_SF** | 0 | 0 | 0 | 0 | 0 | ***SF*** | 0 | 0 |

&

| **INTRARE** | PRG | RST | AC | I | SET | ***SF*** | ***SG*** | ***SP*** |
|---|---|---|---|---|---|---|---|---|
| **MASCA_SG** | 0 | 0 | 0 | 0 | 0 | 0 | ***1*** | 0 |
| **TMP_SG** | 0 | 0 | 0 | 0 | 0 | 0 | ***SG*** | 0 |

&

| **INTRARE** | PRG | RST | AC | I | SET | ***SF*** | ***SG*** | ***SP*** |
|---|---|---|---|---|---|---|---|---|
| **MASCA_SP** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ***1*** |
| **TMP_SP** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ***SP*** |

&

Deplasările nu sunt necesare, astfel că se poate calcula direct indexul pentru tabela de adevăr:

**INDEX = TMP_SF | TMP_SG | TMP_SP**

| **TMP_SF** | 0 | 0 | 0 | 0 | 0 | ***SF*** | 0 | 0 |
|---|---|---|---|---|---|---|---|---|
| **TMP_SG** | 0 | 0 | 0 | 0 | 0 | 0 | ***SG*** | 0 |
| **TMP_SP** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ***SP*** |
| **INDEX** | 0 | 0 | 0 | 0 | 0 | ***SF*** | ***SG*** | ***SP*** |

(SAU)

Intrarea CLC-ului, având trei biți, poate fi descrisă prin 8 valori, astfel:

| SF | SG | SP |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |
| 1 | 1 | 1 |

și ținând cont că senzorul este activ în 1 =>

| SF | SG | SP | Semnificație |
|---|---|---|---|
| 0 | 0 | 0 | - |
| 0 | 0 | 1 | activ SP |
| 0 | 1 | 0 | activ SG |
| 0 | 1 | 1 | activ SG,SP |
| 1 | 0 | 0 | activ SF |
| 1 | 0 | 1 | activ SF,SP |
| 1 | 1 | 0 | activ SF,SG |
| 1 | 1 | 1 | activ SF,SG,SP |

Pe baza semnificației celor 8 combinații de valori ale biților de intrare, se pot determina ieșirile, astfel:

| **INTRARE** | | | | **IEȘIRE** | | | |
|---|---|---|---|---|---|---|---|
| **SF** | **SG** | **SP** | **Semnificație** | **LED1** | **LED0** | **Semnificație** | **Val Hexa** |
| 0 | 0 | 0 | - | 1 | 1 | *Fără erori* | 0x03 |
| 0 | 0 | 1 | *Activ SP* | 1 | 0 | *Eroare P* | 0x02 |
| 0 | 1 | 0 | *Activ SG* | 0 | 1 | *Eroare G* | 0x01 |
| 0 | 1 | 1 | *Activ SG,SP* | 0 | 0 | *Eroare F sau orice combinație* | 0x00 |
| 1 | 0 | 0 | *Activ SF* | 0 | 0 | *Eroare F sau orice combinație* | 0x00 |
| 1 | 0 | 1 | *Activ SF,SP* | 0 | 0 | *Eroare F sau orice combinație* | 0x00 |
| 1 | 1 | 0 | *Activ SF,SG* | 0 | 0 | *Eroare F sau orice combinație* | 0x00 |
| 1 | 1 | 1 | *Activ SF,SG,SP* | 0 | 0 | *Eroare F sau orice combinație* | 0x00 |

Tabela de adevăr a CLC-ului va fi definită ca vector TAB cu 8 elemente, după cum urmează:

**char TAB[8] = { 0x03, 0x02, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00 }**

Deoarece monitorizarea erorilor se va face pentru oricare dintre cele două moduri de operare, CLC poate fi implementat ca o funcție ce poate fi apelată în interiorul codului la momentele precizate:

Varianta 1 – rezolvarea folosind o singură mască

```
char in;
char eroare;
char TAB[] = {0x03,0x02,0x01,0x00,0x00,0x00,0x00,0x00};

void CLC(void) {
        char selectie;
        selectie = in & 0x07;
        eroare = TAB[selectie];
}
```

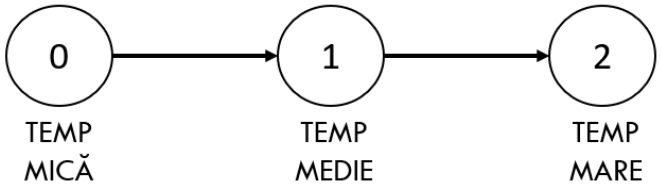
Varianta 2 – rezolvarea folosind trei măști

```
char in;
char eroare;
char TAB[] = {0x03,0x02,0x01,0x00,0x00,0x00,0x00,0x00};

void CLC(void) {
        char SF, SG, SP, selectie;
        SF = in & 0x04;
        SG = in & 0x02;
        SP = in & 0x01;
        selectie = SF | SG | SP;
        eroare = TAB[selectie];
}
```

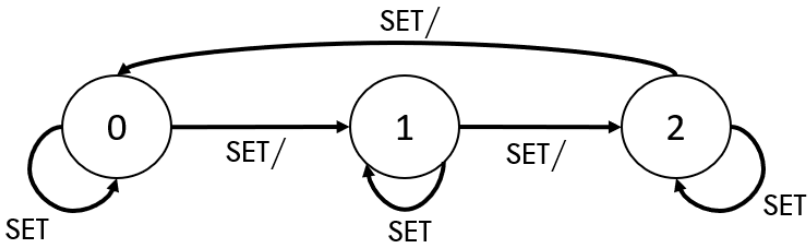
# REZOLVARE

## Graful asociat CLS

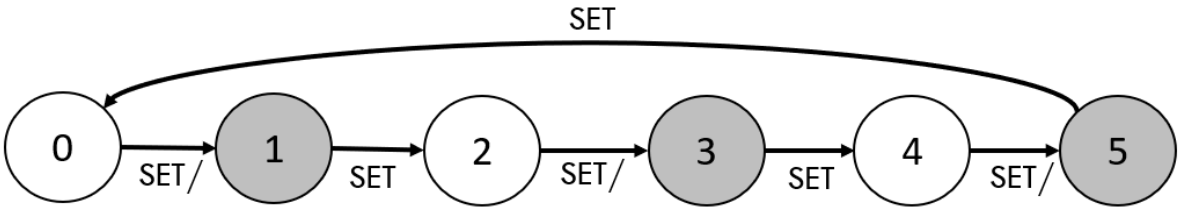
Circuitul logic secvențial (CLS) este utilizat pentru stabilirea temperaturii AC/I.
Pentru crearea grafului asociat CLS, este necesară definirea stărilor, respectiv modul de tranziție dintr-o stare în alta. Conform cerinței, modificarea temperaturii se realizează în mod ciclic și poate fi setată în trei trepte: temperatură mică, medie sau mare. Astfel, se pot defini trei stări, conform temperaturii AC/I:




Trecerea dintr-o stare în alta este realizată cu ajutorul butonului SET (SW3). Astfel, condiția care determină tranziția este reprezentată de apăsarea butonului SET. Dacă butonul SET este apăsat (SET/), atunci se efectuează tranziția, altfel se păstrează starea, iar graful devine:




În plus, cerința menționează faptul că butoanele sunt cu revenire (apăsat și eliberat), astfel că apăsarea unui buton este reprezentată, de fapt, prin două tranziții: una corespunzătoare apăsării butonului, respectiv una corespunzătoare revenirii. Ținând cont de acest aspect, graful asociat CLS devine:




Apăsarea butonului determină tranziția, respectiv schimbarea temperaturii AC/I, iar eliberarea este o consecință a apăsării butonului. Din acest motiv, valoarea ieșirii (temperatura nou setată) se va modifica la apăsarea butonului SET, adică în stările 1, 3 și 5. În stările determinate de eliberarea butonului SET valoarea ieșirii nu se modifică. Fiecare stare în parte, semnificația acesteia, respectiv valoarea ieșirii sunt prezentate în următorul tabel:

| Stare | Semnificaţie | Valoare ieşire | |
|---|---|---|---|
| | | Binar | Zecimal |
| 0 | Stare iniţială **TEMP MICĂ** | 10 | 2 |
| 1 | Apăsare SET **TEMP MEDIE** | 01 | 1 |
| 2 | Eliberare SET | 01 | 1 |
| 3 | Apăsare SET **TEMP MARE** | 00 | 0 |
| 4 | Eliberare SET | 00 | 0 |
| 5 | Apăsare SET **TEMP MICĂ** | 10 | 2 |

# REZOLVARE

## Tabelele de semnale relevante pentru CLS și determinarea măștilor necesare

Conform teoriei CLS, fiecărei stări Qi I se asociază un tabel de semnale relevante Ai. Graful CLS a determinat un număr de 6 stări (0-5) pentru care se asociază 6 tabele de semnale relevante. Trecerea dintr-o stare în alta este determinată de apăsarea butonului SET (SW3).

**TAB**

| Stare | Adresa tabelei de semnale relevante |
|---|---|
| 0 | A0 |
| 1 | A1 |
| 2 | A2 |
| 3 | A3 |
| 4 | A4 |
| 5 | A5 |

Formatul unui tabel de semnale relevante este următorul: $Ai = \{X_{i0}, Q_{i0}, X_{i0}, Q_{i0}, \dots, T, i\}$. În tabelul de semnale relevante asociat stării i, se trec perechi $(X_{ik}, Q_{ik})$, reprezentând semnalul relevant *k* în starea *i*, respectiv identificatorul următoarei stări *i*, dacă la intrare a apărut semnalul relevant *k*. Atunci când nu mai există semnale relevante pentru o stare, se va trece în tabela de semnale relevante perechea $(T, i)$, reprezentând terminatorul de tablou, respectiv starea asociată tabelei de semnale revelante.

Conform detaliilor prezentate anterior, cele șase tabele de semnale relevante asociate CLS pot fi descrise astfel:

$$A0[\,] = \{0x00\,, 1\,, T, 0\}$$
$$A1[\,] = \{0x08\,, 2\,, T, 1\}$$
$$A2[\,] = \{0x00\,, 3\,, T, 2\}$$
$$A3[\,] = \{0x08\,, 4\,, T, 3\}$$
$$A4[\,] = \{0x00\,, 5\,, T, 4\}$$
$$A5[\,] = \{0x08\,, 0\,, T, 5\}$$

Semnalul relevant asociat stărilor poate avea două valori, în funcție de starea butonului SET – apăsat sau neapăsat. Dacă butonul SET (SW3) este apăsat, atunci valoarea acestuia este 0, obținându-se, astfel, valoarea hexazecimală 0x00. În caz contrar, dacă butonul SET nu este apăsat, atunci are valoare 1, obținându-se valoarea hexazecimală 0x08.

Ieșirile asociate fiecărei stări au fost prezentate anterior, în cadrul exercițiului care prezintă graful CLS.

Pentru prelucrarea intrării în scopul comparării acesteia cu semnalul relevant, masca asociată realizează selecția bitului corespunzător butonului SET (SW3):

| Intrare | PRG | RST | AC | I | SET | SF | SG | SP | |
|---|---|---|---|---|---|---|---|---|---|
| Masca SET | 0 | 0 | 0 | 0 | **1** | 0 | 0 | 0 | **= 0x08** |

Astfel, comparația se va realiza între rezultatul $IN\ \&\ 0x08$ și valorile semnalelor relevante – $0x00$ sau $0x08$.

Ca și în cazul circuitului logic combinațional, și CLS-ul poate fi implementat ca o funcție ce va fi apelată în cazul setării temperaturii pentru apă caldă și/sau încălzire. Pentru construcția codului, se pot considera următoarele aspecte:

- Valoarea terminatorului de tablou poate fi definită ca o constantă cu ajutorul directivei #define;
- Numărul de stări ale CLS-ului poate fi definit ca o constantă cu ajutorul directive #define;
- Intrarea, ieșirea, respectiv starea CLS pot fi declarate ca variabile globale de tipul char;
- Tabela de adrese va fi definită ca o variabilă pointer de tipul char;
- Tabelele de semnale relevante vor fi definite ca matrice de tipul char;
- Ieșirile asociate CLS vor fi definite cu ajutorul unei matrice de tipul char;

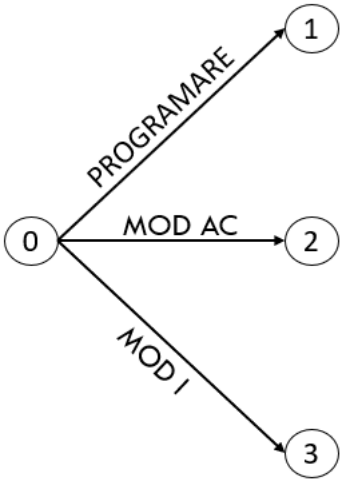
Codul posibil asociat circuitului logic secvențial:

```
#define T    0x10
#define NR 6

char Q;
char in;
char out;

char *TABA[NR];

char A0[] = {0x00,1,T,0};
char A1[] = {0x08,2,T,1};
char A2[] = {0x00,3,T,2};
char A3[] = {0x08,4,T,3};
char A4[] = {0x00,5,T,4};
char A5[] = {0x08,0,T,5};

char Tout[] = {0x02, 0x01, 0x01, 0x00, 0x00, 0x02};

TABA[0]=A0;
TABA[1]=A1;
TABA[2]=A2;
TABA[3]=A3;
TABA[4]=A4;
TABA[5]=A5;

Q=0;

void cls(void){
        char i;
        char *adr;
        char ready;

        adr=TABA[Q];
        i=0;
        ready=0;

        while (!ready){
                if ((IN&0x08) == *(adr+i)) {Q=*(adr+i+1); ready=1;}
                else if (*(adr+i)==T) ready=1;
                else i=i+2;
        }

        out = Tout[Q];
}
```

# REZOLVARE

## Graful PS

Funcționarea modulului software al centralei termice de apartament se bazează pe un proces secvențial. Pentru crearea grafului asociat PS, se poate considera o stare inițială (starea 0) în care centrală este pornită și așteaptă comenzi. Din această stare, sunt posibile trei tranziții, după cum urmează: programare temperatură, mod operare AC, respectiv mod operare I:

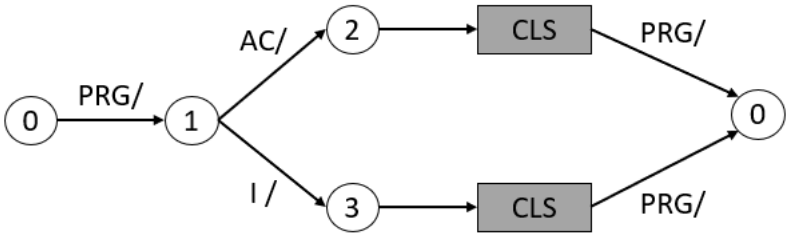



**Mod programare temperatură AC/I**

Conform cerinței, programarea temperaturii se face astfel:

- Trecerea în modul de programare prin apăsarea butonului PRG (SW7)
- Selecția temperaturii de programat (AC sau I) prin apăsarea butonului AC (SW5) sau I (SW4)
- Modificarea temperaturii în trei trepte (mică, medie, mare) prin apăsarea butonului SET (SW3)
- Ieșirea din modul de programare prin apăsarea butonului PRG (SW7)

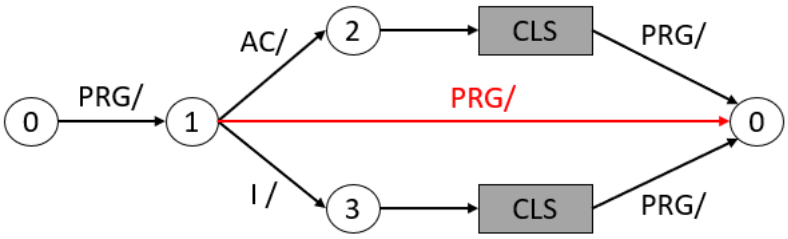
În plus, ținând cont de faptul că modificarea temperaturii se realizează prin intermediul circuitului logic secvențial (CLS), graful asociat programării temperaturii devine:



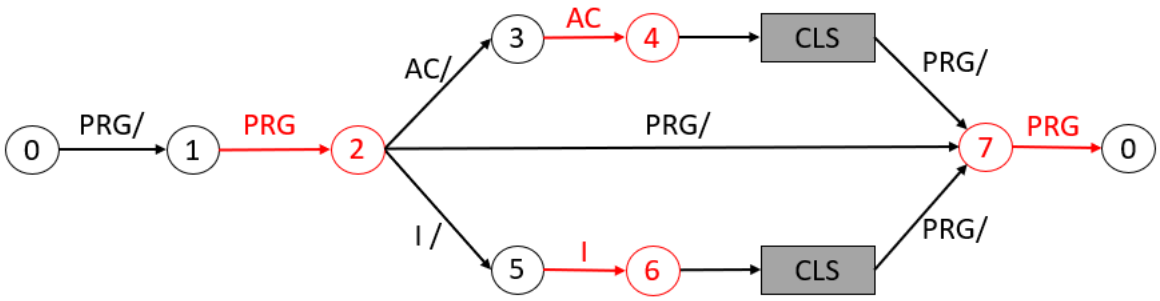

Stările au următoarele semnificații:

0 – Starea inițială
1 – Modul de programare
2 – Selectare programare a temperaturii AC
3 – Selectare programare a temperaturii I

Există posibilitatea ca în urma apăsării butonului PRG, utilizatorul să renunțe la programarea temperaturii și să revină în starea inițială prin apăsarea butonului PRG, iar graful asociat programării temperaturii devine:



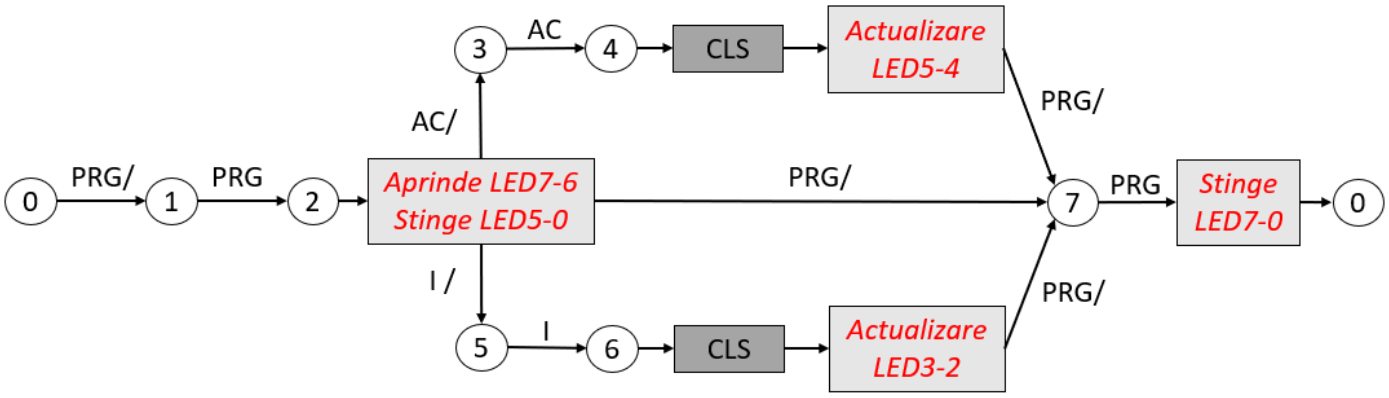

În plus, conform cerinței, butoanele sunt cu revenire (apăsat și eliberat), astfel că graful devine:




În momentul de față, graful ilustrează modalitatea de tranziție dintr-o stare în alta, dar nu și ieșirile corespunzătoare diferitelor stări. Astfel, se pot considera următoarele:

- Selecția modului de programare are ca efect aprinderea LED-urilor corespunzătoare AC și I, respectiv stingerea celorlalte
- Ieșirea CLS asociat AC are ca efect actualizarea LED-urilor corespunzătoare temperaturii AC
- Ieșirea CLS asociat I are ca efect actualizarea LED-urilor corespunzătoare temperaturii I
- Revenirea în starea inițială din modul de programare are ca efect stingerea tuturor LED-urilor.

Ținând cont de aceste aspecte, graful devine:




**Mod de operare AC (apă caldă) sau I (apă caldă și încălzire)**

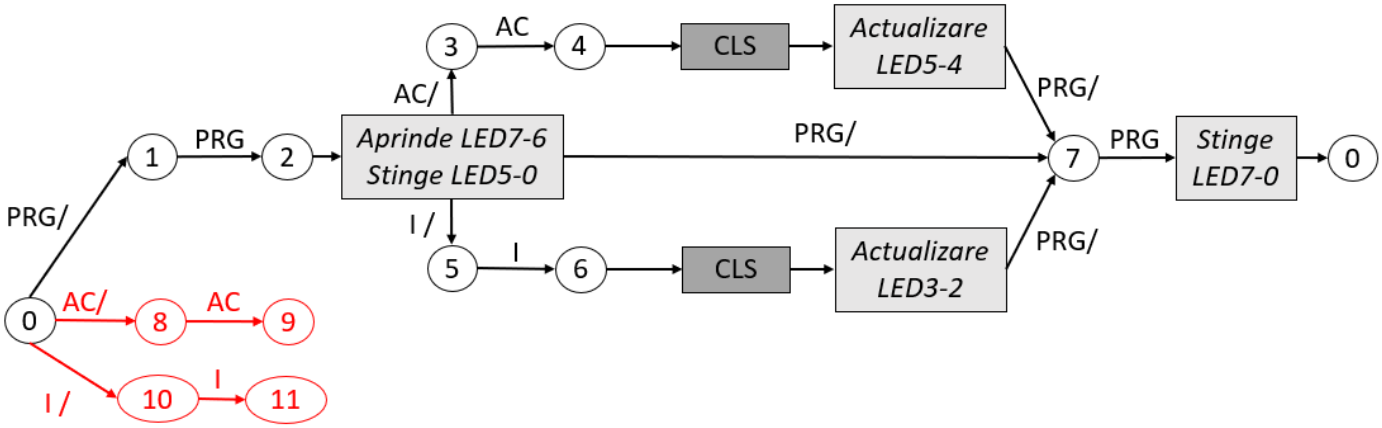
Conform cerinței, selecția modului de operare se realizează prin apăsarea butonului corespunzător modului de operare selectat (AC-SW5 sau I-SW4). Se va ține cont de următoarele aspecte:

- Din modul de operare se poate trece în modul de programare prin apăsarea butonului PRG(SW7)
- Modul de operare se poate schimba prin apăsarea butonului asociat modului de operare dorit
- La apariția unei erori sistemul se blochează, iar pentru revenirea la funcționarea normală, este necesară apăsarea butonului RST(SW6). Resetarea va fi semnalizată prin aprinderea, respectiv stingerea LED7-0.

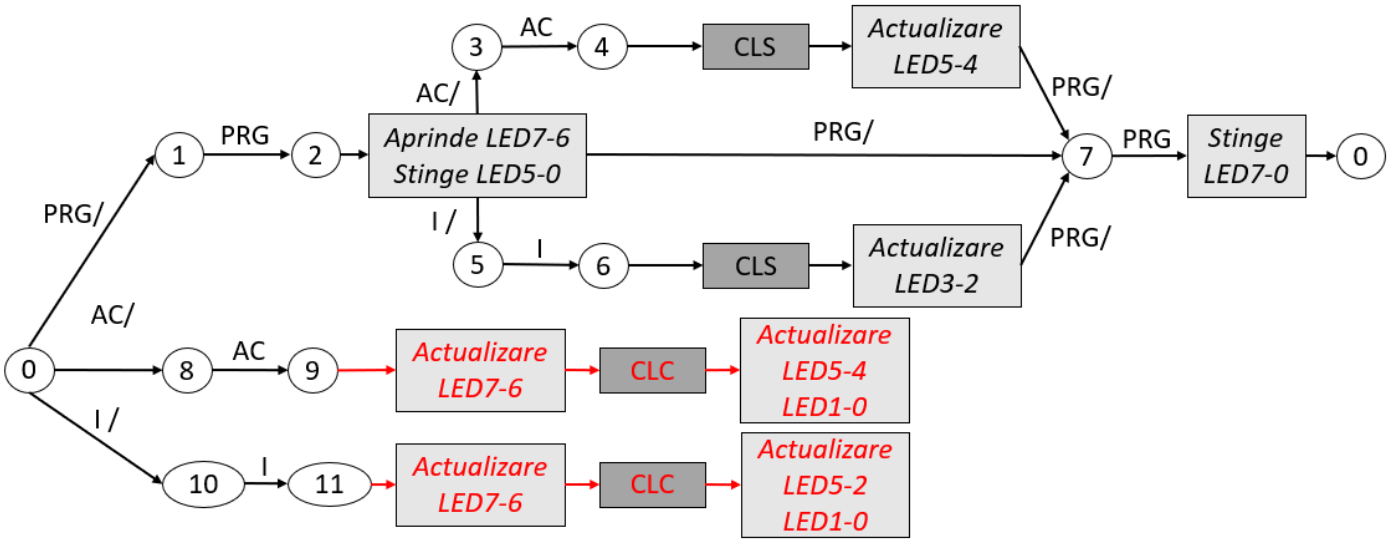
În plus, vor fi luate în considerare și următoarele indicații:

- Trecerea în modul de operare AC/I va avea ca efect actualizarea LED7-6 pentru afișarea modului de operare
- Modul de operare AC va avea ca efect actualizarea LED5-4 pentru afișarea temperaturii, respectiv LED1-0 ca ieșire a CLC-ului
- Modul de operare I va avea ca efect actualizarea LED5-2 pentru afișarea temperaturilor, respectiv LED1-0 ca ieșire a CLC-ului

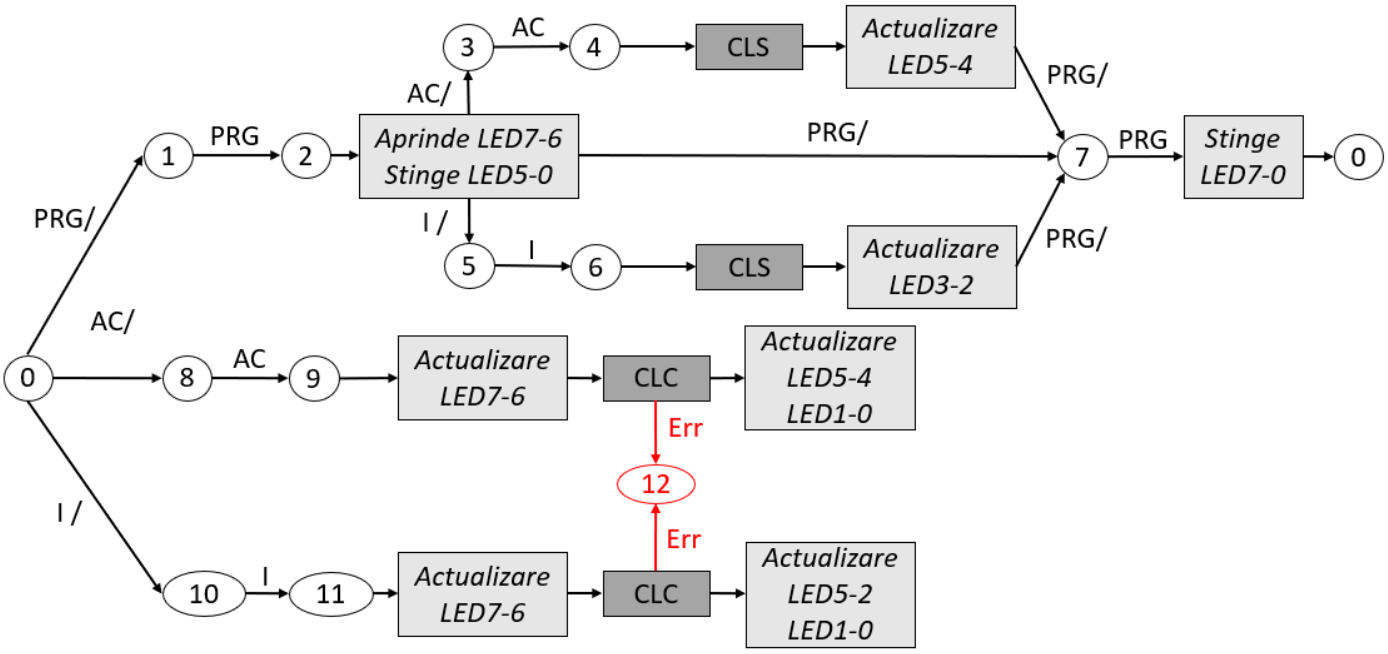
Inițial, graful va ilustra tranzițiile de bază din starea inițială în stările asociate celor două moduri de operare:



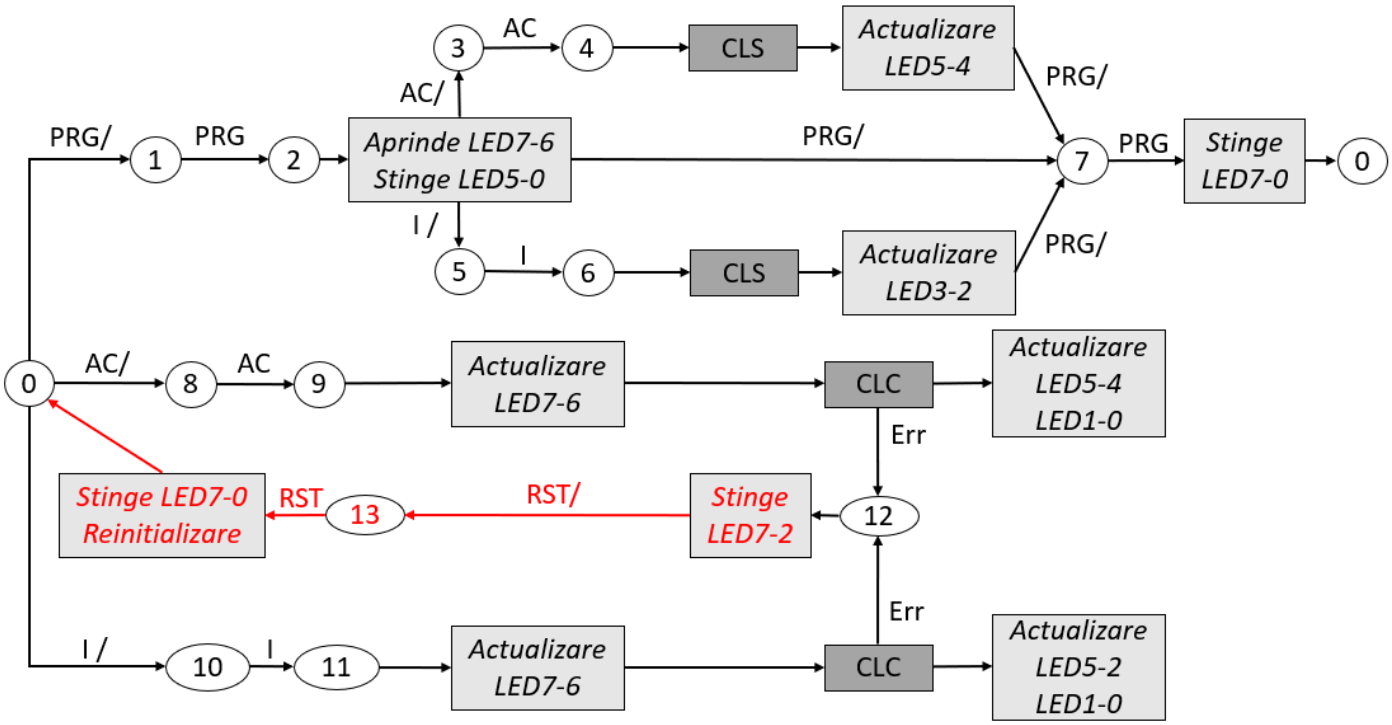

Consecința trecerii în modul de operare AC sau I are ca efect aprinderea LED7-6 pentru afișarea modului de operare selectat. În plus, conform cerinței, se afișează temperatura programată, respectiv erorile generate cu ajutorul CLC, iar graful devine:




În cazul apariției unei erori, sistemul se va bloca, astfel că este necesară tranziția într-o stare separată, specifică acestei situații. Ținând cont de acest aspect, graful devine:

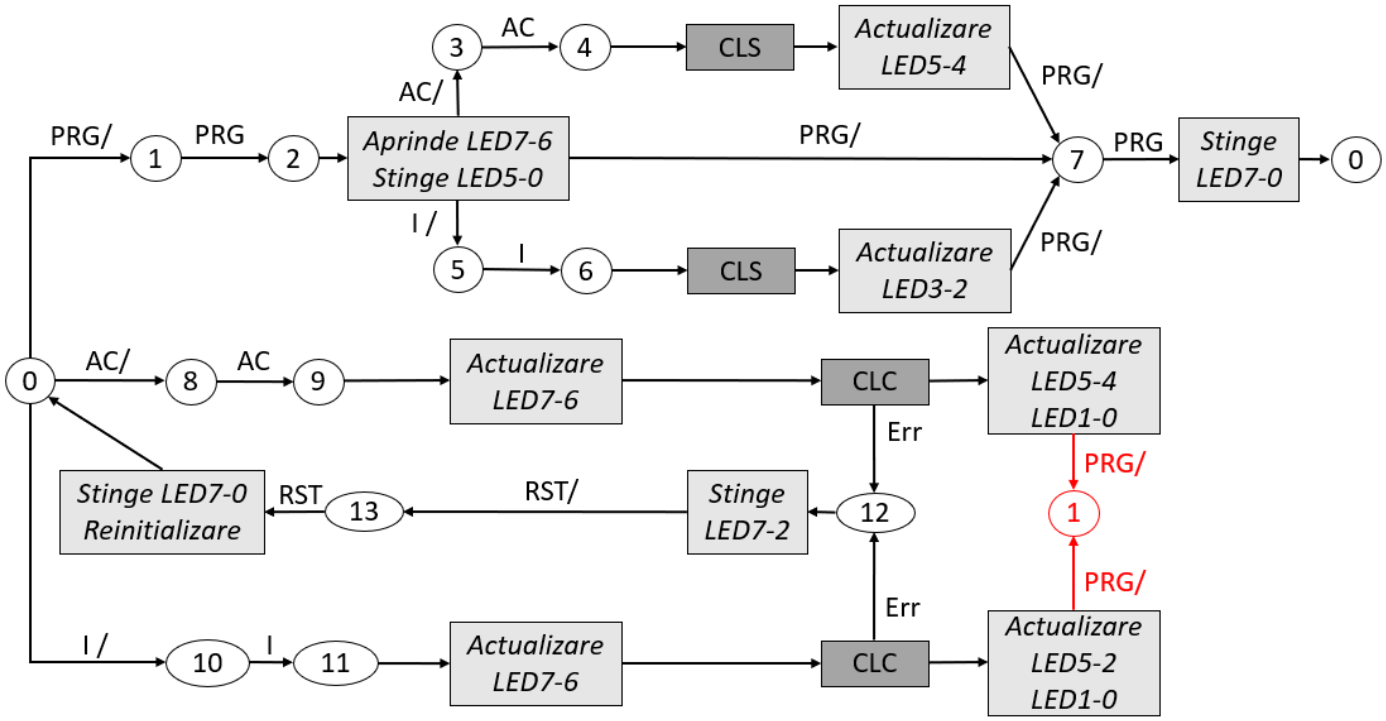



Pentru revenirea la funcționarea normală, este necesară apăsarea butonului RST, care are ca efect aprinderea, respectiv stingerea tuturor LED-urilor si reinițializarea prin revenirea în starea 0. Graful devine:

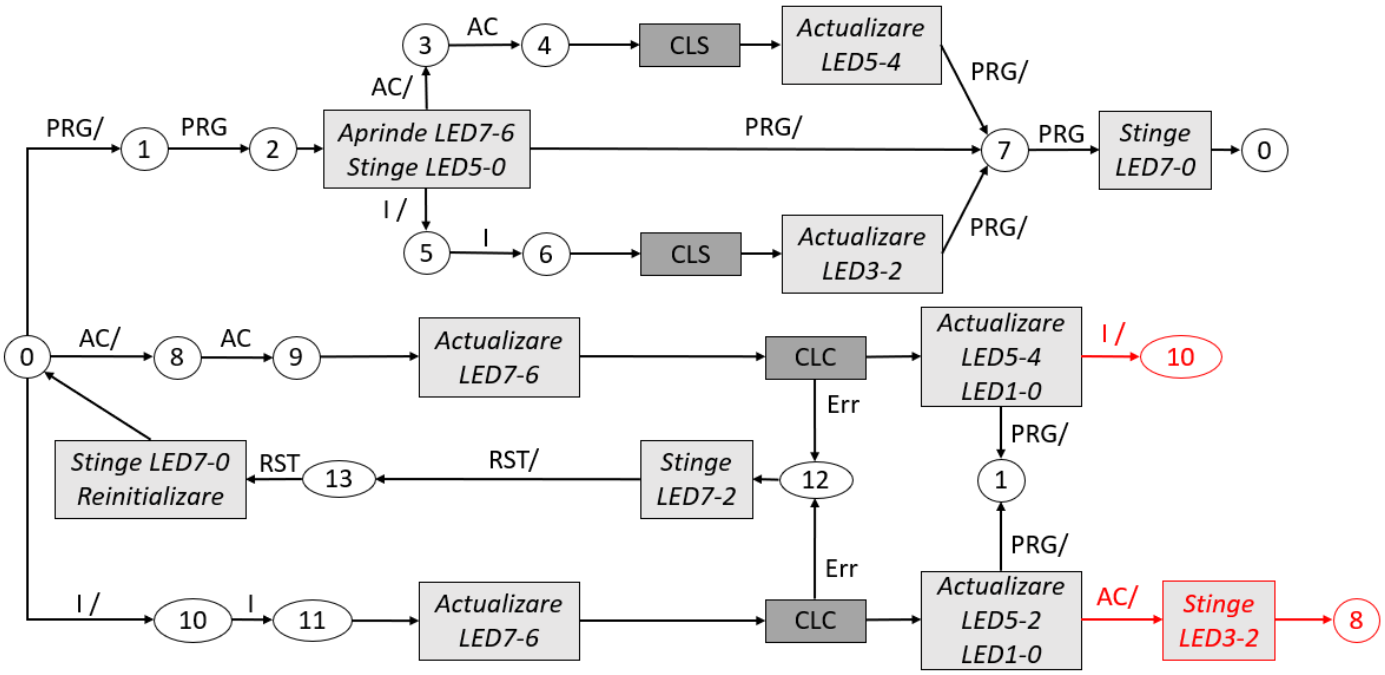



În cazul în care nu există erori, în urma actualizării LED-urilor corespunzătoare temperaturii, respectiv a celor corespunzătoare erorilor, se poate trece în modul de programare a temperaturii AC/I prin apăsarea butonului PRG. Această acțiune are ca efect trecerea în starea 1, stare corespunzătoare modului de programare.

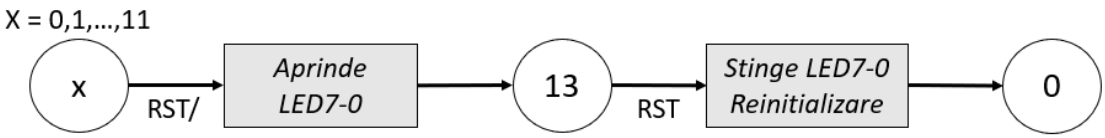
Graful devine:




În cazul în care se dorește schimbarea modului de operare, este necesară apăsarea butonului corespunzător modului dorit. Dacă se dorește trecerea în modul de operare I, apăsarea butonului I are ca efect tranziția în starea 10, corespunzătoare acestui mod de operare. Dacă se dorește tranziția în modul de operare AC, este necesară apăsarea butonului AC, respectiv stingerea LED-urilor LED3-2. Tranziția are loc în starea 8, corespunzătoare modului de operare Apă Caldă. Graful devine:




În plus, din orice stare se poate apăsa butonul RST, corespunzător reinițializării, astfel că graful poate fi completat cu următoarele tranziții:



# REZOLVARE

## Codul în limbaj C pentru întregul modul software (CLC, CLS şi PS)

```
#define T 0x10        // terminator CLS
#define NR 6          // numar stari CLS

char err;             // erori
char tempAC;          // temperatura AC
char tempI;           // temperatura I
char S;               // starea PS
char Q;               // starea CLS
char in;              // intrarea
char out;             // ieşirea

char TAB[]={ 0x03, 0x02, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00 };
// tabela CLC - senzori activi in 1

char *TABA[NR];    //tabela de adrese

// tabelele de semnale relevante CLS
char A0[]={0x00,1,T, 0};
char A1[]={0x08,2,T ,1};
char A2[]={0x00,3,T ,2};
char A3[]={0x08,4,T ,3};
char A4[]={0x00,5,T ,4};
char A5[]={0x08,0,T ,5};

char Tout[] = {0x02, 0x01, 0x01, 0x00, 0x00, 0x02};
//tabela iesirilor CLS

void main(void) {
   // initializare adrese tabele de semnale relevante
   TABA[0]=A0;
   TABA[1]=A1;
   TABA[2]=A2;
   TABA[3]=A3;
   TABA[4]=A4;
   TABA[5]=A5;
   S=0;
   Q=0;
   out=0xFF;
   while (1)
   {
   };
}
void clc(void)
{
   char sens;
   sens=in & 0x07; // senzorii SF - SW2, SG - SW1, SP - SW0
   err = TAB[sens]; // eroarea - bitii 0, 1
   if(err !=0x03) S=12; // blocare pe eroare
}

void cls(void)
{
   char i;
   char *adr;
   char ready;
   adr=TABA[Q];
   i=0;
   ready=0;
   while (!ready)
   {
      if ((in & 0x08) ==*(adr+i)) {Q=*(adr+i+1); ready=1;}
      else if (*(adr+i)==T) ready=1;
      else i=i+2;
   }// iesirea CLS este TempAC sau TempI }
```

```
void test_RST(void) // testeaza Reset (RST)
{
   if ((in & 0x40)==0)
   {
      out = out &0x00; // aprinde LED-0-7
      S=13;
   }
}

// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
TCNT0=0x4E;
in=PIND; // citeste intrarea

switch (S)
{
case 0:
if ((in & 0x80)==0) S=1; // testeaza PRG/
if ( (in & 0x20) == 0 ) S=8; // testeaza AC/
if ( (in & 0x10) == 0 ) S=10; // testeaza I/
test_RST();// testeaza RST/
break;

case 1:
if ((in & 0x80)!=0) S=2; // testeaza PRG
test_RST();// testeaza RST/
break;

case 2:
out = (out | 0xFF) & 0x3F; // aprinde LED 7, 6 , stinge LED 5-0
if ( (in & 0x20) == 0 ) S=3; // testeaza AC/
if ( (in & 0x10) == 0 ) S=5; // testeaza I/
if ( (in & 0x80) == 0 ) S=7; // testeaza PRG/
test_RST();// testeaza RST/
break;

case 3:
if ( (in & 0x20) != 0 ) S=4; // testeaza AC
test_RST();// testeaza RST/
break;

case 4:
cls();
tempAC = Tout[Q] << 4; // iesire CLS - bitii 4, 5
out = out & 0xCF; // sterge bitii 4 si 5 - masca 1100 1111
out = out | tempAC; // actualizeaza out
if ((in & 0x80)==0) S=7; // testeaza PRG/
test_RST();// testeaza RST/
break;

case 5:
if ( (in & 0x10) != 0 ) S=10; // testeaza I
test_RST();// testeaza RST/
break;

case 6:
cls();
tempI = Tout[Q] << 2; // iesire CLS - bitii 2, 3
out = out & 0xF3; // sterge bitii 2 si 3 - masca 1111 0011
out = out | tempI; // actualizeaza out
if ((in & 0x80)==0) S=7; // testeaza PRG/
test_RST();// testeaza RST/
break;
```

```
case 7:
if ( (in & 0x80) != 0 ) // testeaza PRG/
{
out = out | 0xFF; // stinge LED 0-7
S=0;
}
test_RST();// testeaza RST/
break;

case 8:
if ( (in & 0x20) != 0 ) S=9; // testeaza AC
test_RST();// testeaza RST/
break;

case 9:
out = out & 0x3F; // sterge bitii 7, 6
out = out | 0x40; // 0100 0000 AC=0, I =1
clc();
out = out & 0xCC; // sterge bitii 0,1, 4 si 5 - masca 1100 1100
out = out | err | tempAC; // actualizeaza out
if ((in & 0x80)==0) S=1; // testeaza PRG/
if ((in & 0x10)==0) S=10; // testeaza I/
test_RST();// testeaza RST/
break;

case 10:
if ( (in & 0x10) != 0 ) S=11; // testeaza I
test_RST();// testeaza RST/
break;

case 11:
out = out & 0x3F; // sterge bitii 7, 6
out = out | 0x80; // 1000 0000 AC=1, I =0
clc();
out = out & 0xC0; // sterge bitii 0,1, 2, 3, 4 si 5 - masca 11000000
out = out | err | tempAC | tempI; // actualizeaza out
if ((in & 0x80)==0) S=1; // testeaza PRG/
if ((in & 0x20)==0) // testeaza AC/
{
out = out | 0x0C; // stinge LED 3, 2
S=8;
}
test_RST();// testeaza RST/
break;

case 12:
out=out | 0xFC; // stinge LED 7-2
test_RST();
break;

case 13:
if ((in & 0x40)!=0)
{
out = out | 0xFF; // stinge LED 0 - 7
err=0x03;
out = out | err; // sterge erori
S=0;
}
break;
}
PORTB=out; // scrie iesirea
}
```