Sa se realizeze o aplicatie cu microcontroler, pe placa de evaluare STK 500, cu urmatoarele specificatii:

- operatia de baza: la apasarea butonului SW_k se va aprinde LED-ul LED_{3-k} , k = 0,1,2,3
- operatia de baza este conditionata de introducerea unei parole de activare: 2481
- initial sistemul este neactivat
- dupa activare, incheierea operatiei de baza se va face prin apasarea butonului SW_4 , apoi sistemul poate fi blocat cu parola de blocare: 8421
- parolele sint introduse cu ajutorul butoanelor $SW_3 – SW_0$ (cifra in parola=$2^k \rightarrow$ SW_k – apasat)
- introducerea parolelor va fi facuta astfel: se apasa butonul SW_7 , se introduce parola si se apasa butonul SW_6
- introducerea unei parole gresite va fi resetata prin apasarea butonului SW_5
- se va afisa pe LED_7 starea de blocare a sistemului (LED aprins) si pe LED_6 aparitia unei erori (parola gresita – LED aprins)
- operatia de baza se va implementa cu ajutorul unui CLC
- introducerea parolelor se va implementa cu ajutorul unui CLS
- intreaga aplicatie va fi descrisa printr-un process secvential

Solutia propusa:

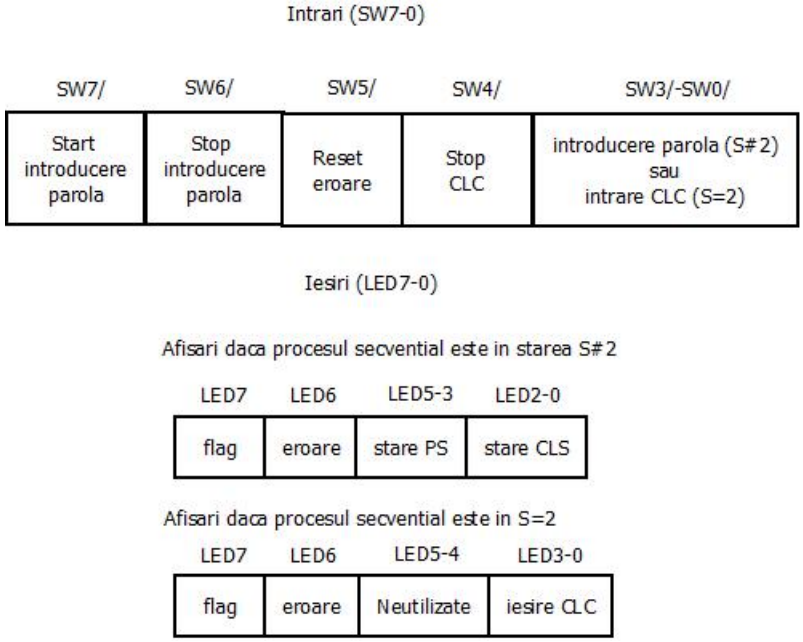

Figura 1. Alocarea resurselor I/O

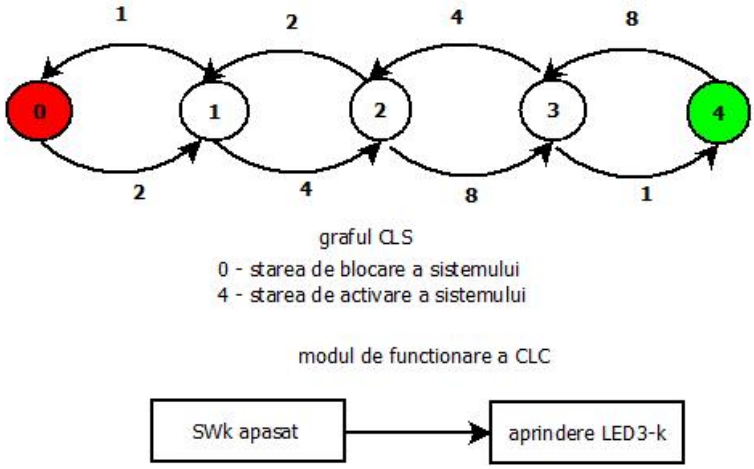

Figura. 2 Descrierea functionarii CLS si CLC

SW 3	SW 2	SW 1	SW 0	LED 7	LED 6	LED 5	LED 4	LED 3	LED 2	LED 1	LED 0
0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	1	1	1	1	1	1	1	0	1
1	1	0	1	1	1	1	1	1	0	1	1
1	1	1	0	1	1	1	1	0	1	1	1
x	x	x	x	1	1	1	1	1	1	1	1

Figura 3. Tabela de adevar a CLC

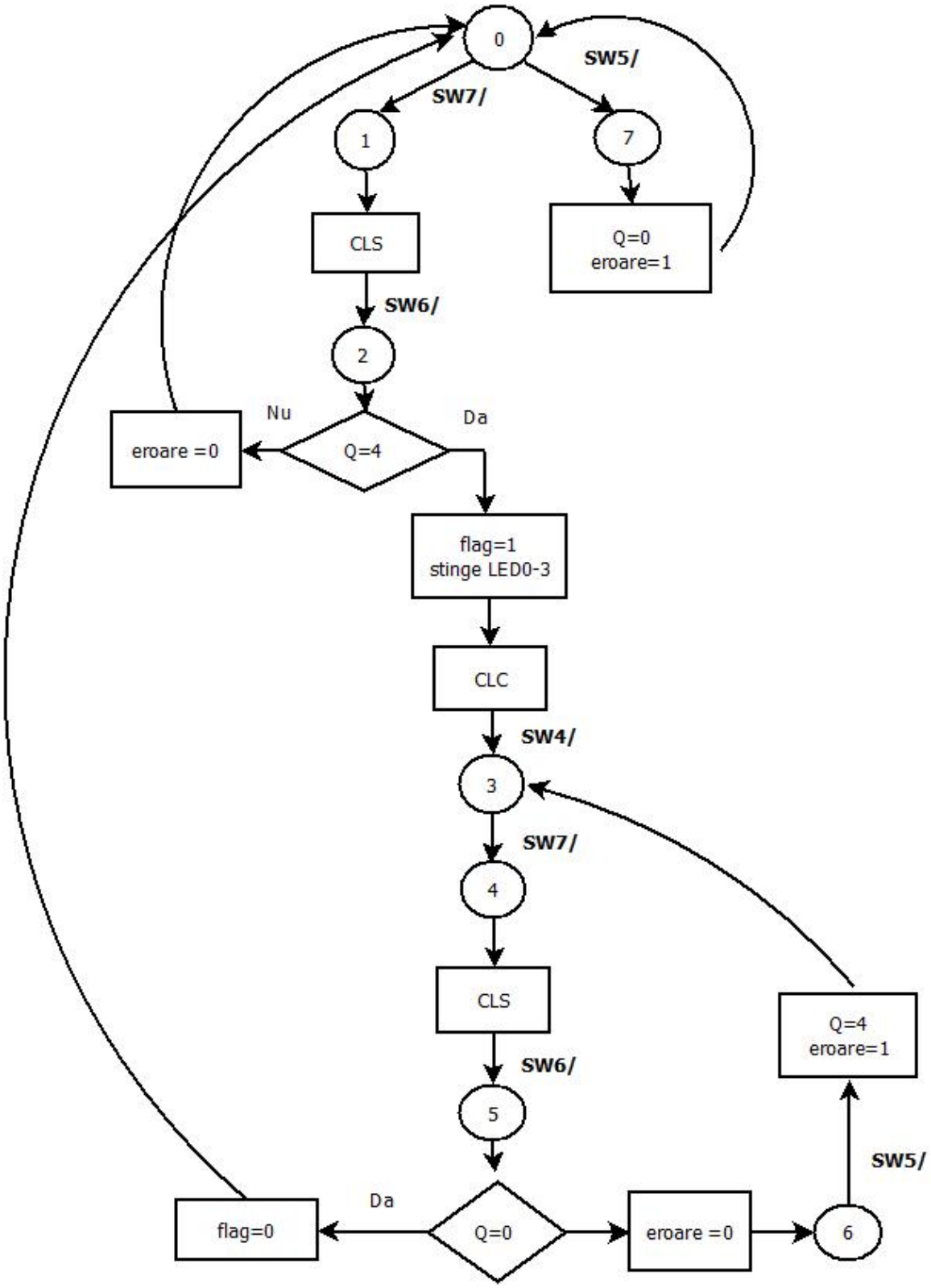

Figura 4. Graful hybrid de tranzitii al PS

Programul in limbaj C

```
// Variabilele globale

char S;    // starea PS
char Q;    // starea CLS
char flag; // flag = 0 - sistem inactiv, flag=1 - sistem activ
char eroare; // eroare = 0 - eroare introducere parola
char in;   // intrarea SW7-SW0
char out;  // iesirea CLC


// tabela de adevar a CLC

char TAB[16]={
0xFF,0xFF,0xFF,0xFF,
0xFF,0xFF,0xFF,0xFE,
0xFF,0xFF,0xFF,0xFD,
0xFF,0xFB,0xF7,0xFF
};

char *TAB1[5];  //tabela de adrese

// tabelele de semnale relevante
char A0[]={2,1,16,0};
char A1[]={4,2,1,0,16,1};
char A2[]={8,3,2,1,16,2};
char A3[]={1,4,4,2,16,3};
char A4[]={8,3,16,4};


// initializari variabile

flag=0;
eroare=1;
PORTB=(flag << 7) | (eroare << 6) | 0x3F;

TAB1[0]=A0;
TAB1[1]=A1;
TAB1[2]=A2;
TAB1[3]=A3;
TAB1[4]=A4;

S=0;
Q=0;
out=0xFF;

// functia CLC

void clc(void)
{
```

```
char tmp;
tmp=in & 0x0F;
out=TAB[tmp];

}

// functia CLS

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

adr=TAB1[Q];
i=0;
ready=0;

while (!ready)
{
if (~in==*(adr+i)) {Q=*(adr+i+1); ready=1;}
else if (*(adr+i)==16) ready=1;
      else i=i+2;
}

}


// Rutina de servire a intreruperii

interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
// Reinitialize Timer 0 value
TCNT0=0x4E;
// Place your code here
in =PIND;
switch (S)
{
case 0:
if (in==0x7F) {S=1;} //SW7
if (in==0xDF) {S=7;} //SW5
break;
case 1:
cls();
if (in==0xBF) {S=2;} //SW6
break;
case 2:
if (Q==4) {
flag=1;
PORTB=PORTB | 0x0F;
clc();
if (in==0xEF) {S=3;} //SW4
```

```
}
else
{
eroare=0;
S=0;
}
break;
case 3:
if (in==0x7F) { S=4;} //SW7
break;
case 4:
cls();
if (in==0xBF) {S=5;} //SW6
break;
case 5:
if (Q==0) {
flag=0;
S=0;
}
else
{
eroare=0;
S=6;
}
break;
case 6:
if (in==0xDF) {Q=4; eroare=1; S=3;} //SW5
break;
case 7:
Q=0;
eroare=1;
S=0;
break;
}
if (S!=2) PORTB = (((((flag << 7) | (eroare << 6) ) | 0x3F ) &
out) &  (~Q) & (~(S<<3)));
else PORTB = (((((flag << 7) | (eroare << 6) ) | 0x3F ) & out) );

}
```