UNIVERSITATEA POLITEHNICĂ DIN BUCUREȘTI

# TEMĂ SUPLIMENTARĂ

Arhitectura microprocesoarelor - Microcontrolere

SAUCĂ ALEXANDRU ANDREI

CUPRINS

LISTA FIGURILOR ȘI LISTA TABELELOR

# INTRODUCERE

Obiectivul principal al acestui proiect este implementarea software a unui bancomat folosind microprocesorul Atmega164P. Bancomatul trebuie să reușească să efectueze operații precum introducere PIN, verificare sold, retragere și să se specifice opțiunea de chitanță. În cazul în care codul PIN este incorect se va lua în calcul aprinderea/stingerea intermitentă a unui LED la alegere timp de 3 secunde. Emiterea chitanței, retragerea cardului din bancomat și ridicarea bancnotelor vor fi marcate vizual folosind LED-uri specifice. Se va considera un scenariu cu maxim 20 de utilizatori.

În realizarea acestui sistem s-a utilizat logica programată, folosind un Circuit Logic Combinațional(CLC), un Circuit Logic Secvențial(CLS) și Procese Secvențiale(PS).

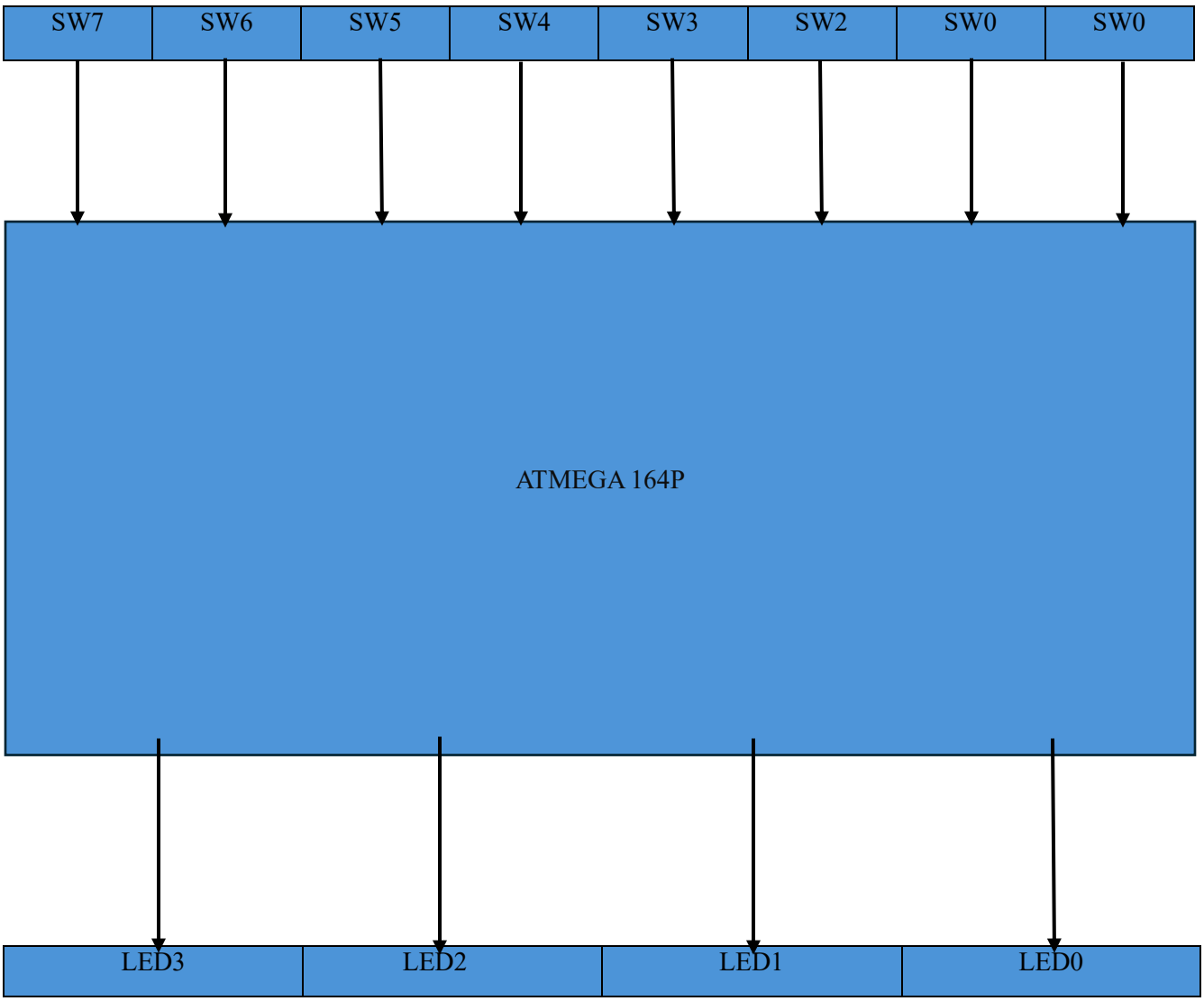
Marcarea vizuală a LED-urilor  pentru emiterea chitanței, retragerea cardului din bancomat și ridicarea bancnotelor a fost realizată cu ajutorul unui CLC, având ca Input-uri comenzile verificare sold, retragere și specificarea opțiunii de chitanță .

Introducerea codului PIN a fost realizată folosind un CLS cu 81 de stări care reușește să verifice fiecare cifră a PIN-ului. Au fost alocate câte 4 stări pentru parola fiecărui utilizator.

Procesele Secvențiale au fost utilizate atât pentru determinarea fiecărei stări în care se află sistemul, cât și pentru revenirea sa în starea inițială.

# CAPITOLUL 1

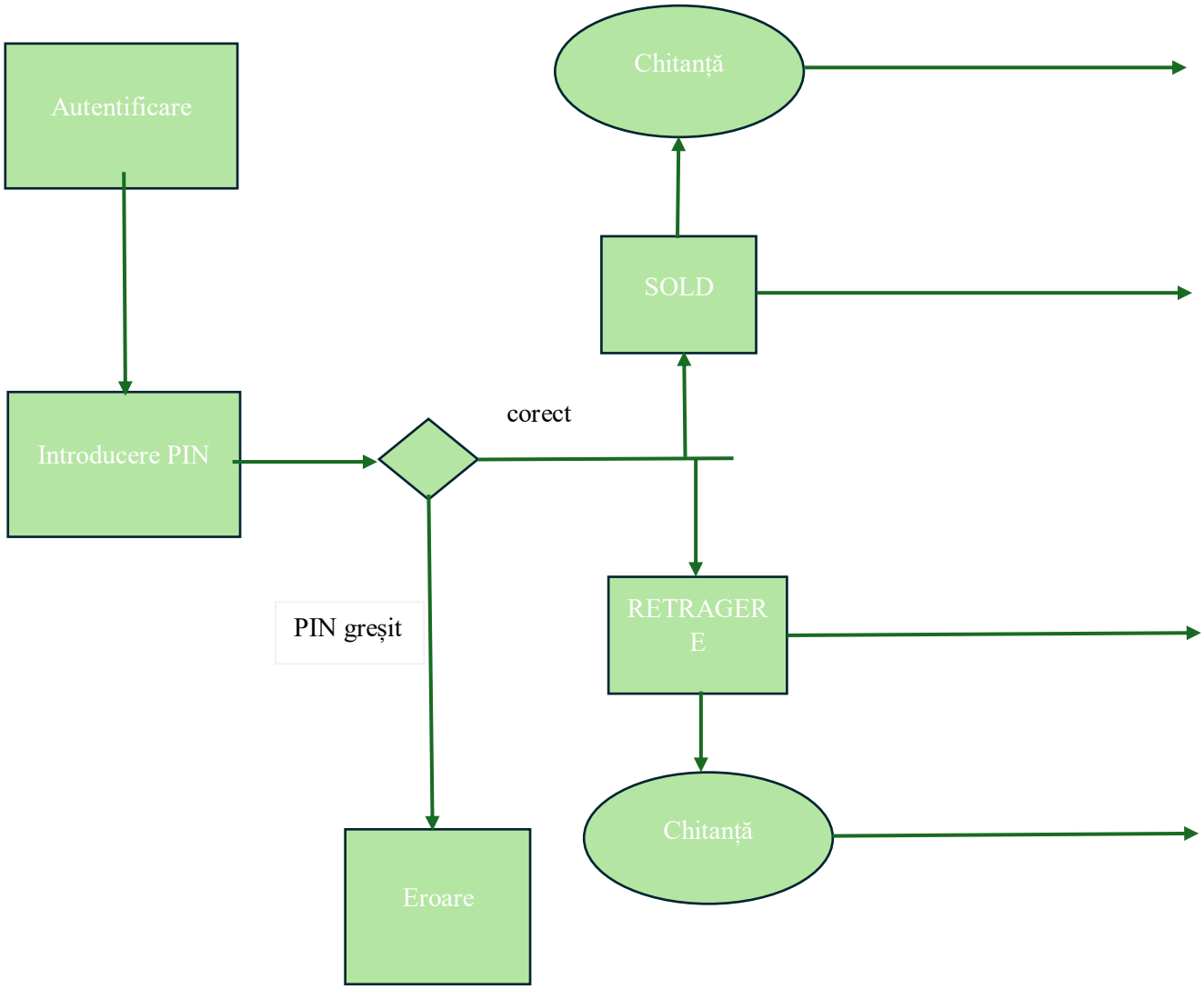
## Schema bloc a sistemului




1.1. Schema bloc a bancomatului

CAPITOLUL 2

Diagrama cazurilor de utilizare a sistemului




2.1. Diagrama cazurilor de utilizare a bancomatului

# CAPITOLUL 3

## Descrierea funcţionalităţii porturilor de intrare şi ieşire

INTRARE – PORT D

| SW7 | SW6 | SW5 | SW4 | SW3 | SW2 | SW1 | SW0 |
|---|---|---|---|---|---|---|---|
| INTRODUCERE PIN | | NUMĂR CARD | | | | | |
| | CONFIRMARE PIN | | | PIN CARD | | | |
| - | - | - | - | STOP CLC | INTRARE CLC(S=2) | | |

3.1. Intrări SW

Numărul cardului este format din următoarea concatenare:

(ID_rezidență | ID_bancă | ID_client)

Pentru simplicarea scenariului se va considera un număr maxim de 20 de utilizatori ținând cont de următoarele:

| Rezidenta | Banca | Numar clienti |
|---|---|---|
| National | ING | 4 |
| | BCR | 6 |
| | BT | 3 |
| International | Revolut | 7 |

3.2.Tabel scenariu

| Rezidența | Bancă | Număr_clienți |
|---|---|---|
| 0 | 00 | 001,010,011,100 |
| | 01 | 001,010,011,100,101,110 |
| | 10 | 001,010,011 |
| 1 | 11 | 001,010,011,100,101,110,111 |

3.3. Valori ID_card

IEȘIRE – PORT B

| LED7 | LED6 | LED5 | LED4 | LED3 | LED2 | LED1 | LED0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | EROARE PIN GREȘIT | CHITANȚĂ | RIDICARE BANCNOTE | RETRAGERE CARD |

3.4. Ieșire LED

SW7, SW6, SW3 reprezintă butoane active in 1 fără revenire:

INTRODUCERE PIN – buton specific pentru încercare de autentificare

CONFIRMARE PIN – buton specific pentru încheierea introducerii codului PIN

STOP CLC – buton specific pentru încheierea acțiunilor SOLD,RETRAGERE,CHITANȚĂ

SW4-SW0 sunt folosite pentru introducerea numărului de la card și verificarea acestuia pentru a se putea introduce ulterior numărul PIN al cardului.

SW3-SW0 – sunt utilizate pentru introducerea numărului PIN al cardului după introducerea ID_card.

În starea 2 a procesului secvențial SW2-SW0 sunt folosite pentru acțiunile SOLD(SW0) , RETRAGERE(SW1) , CHITANȚĂ(SW2).

LED2-LED1 sunt o consecință în urma utilizării CLC-ului pentru a afișa comenzile utilizate , acestea fiind active in 0.

LED3 afișează eroarea introducerii unui PIN greșit prin aprinderea/stingerea sa intermitentă timp de 3 secunde.

| SW2 | SW1 | SW0 | LED2 | LED1 | LED0 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| x | x | x | 1 | 1 | 1 |

3.5. Descrierea funcționalității CLC

# CAPITOLUL 4

## Proiectarea codului sursă corespunzător sistemului

În proiectarea codului sursă am folosit 3 stări de tip Proces Secvențial pentru testarea bancomatului cu 20 de utilizatori. Inițial am pornit de la o stare 0 în care bancomatul așteaptă comenzi. La apăsarea SW7 sistemul trece în starea 1 în care este implementat un CLS pentru aflarea PIN-ului corect corespunzător numărului de la card. Înaintea apelării funcției cls() este apelată o funcție card_number_goes_to_pin() în care se primește ca intrare un număr de card. După introducerea ID_card se apelează funcția card_number_goes_to_pin() în care se testează ce număr de card este , modificând starea Q a CLS-ului pentru verificarea codului PIN. Se introduce pe rând fiecare cifră verificându-se dacă face parte din tabelul de semnale. În cazul în care cifra este corectă, starea CLS se modifică, incrementându-se cu 1, în caz contrar , se rămâne in aceeași stare CLS. Prin apăsarea SW6 se încheie CLS-ul cu ajutorul terminatorului, trecându-se în starea 2.

În această stare se verifică dacă PIN-ul este corect.

În cazul în care starea finală Q este egala cu starea inițială a CLS-ului adunată cu 4 (deoarece acesta este numărul de cifre pe care un cod PIN le are) se pot efectua operațiile sold, retragere, chitanță cu ajutorul unui CLC. Aceste comenzi au un afișaj pe 3 LED-uri exemplificate în tabelul 3.5.Descrierea funcționalității CLC. După efectuarea comenzilor se poate apăsa SW4 care încheie operațiile CLC-ului și se trece în starea 3 . Starea 3 este utilizată pentru resetarea și stingerea SW și a LED-urilor, trecându-se în starea inițială.

Pentru cazul introducerii unui PIN incorect se apelează o funcție PIN_incorect() în care LED3 se aprinde/stinge intermitent timp de 3 secunde cu ajutorul unui contor . După ieșirea din funcție se actualizează toate SW și se trece în starea inițială.

# CAPITOLUL 5
## Implementarea codului sursă

```
// I/O Registers definitions
#include <mega164.h>

char in; //intrare
char out;  //ieşire
char Q; //stare CLS
char eroare; //eroare=0 eroare introducere pin
char S; //stare PS
char Q1;//copie a lui Q

char *TAB1[81]; //tabela de adrese

//tabele de semnale relevante CLS
//pin 1
char A0[] = {5, 1, 16 , 0};
char A1[] = {4, 2 , 16 , 1};
char A2[] = {2, 3 , 16 , 2};
char A3[] = {7, 4 , 16 , 3};
//pin 2
char A4[] = {2, 5 , 16 , 4};
char A5[] = {7, 6, 16 , 5};
char A6[] = {4, 7 , 16 , 6};
char A7[] = {0, 8 , 16 , 7};
//pin 3
char A8[] = {7, 9 , 16 , 8};
char A9[] = {1, 10 , 16 , 9};
char A10[] = {0, 11, 16 , 10};
char A11[] = {1, 12, 16 , 11};
//pin 4
char A12[] = {4, 13, 16 , 12};
char A13[] = {2, 14, 16 , 13};
char A14[] = {1, 15, 16 , 14};
char A15[] = {5, 16, 16 , 15};
//pin 5
char A16[] = {8, 17 , 16 , 16};
char A17[] = {2, 18 , 16 , 17};
char A18[] = {2, 19 , 16 , 18};
char A19[] = {6, 20 , 16 , 19};
//pin 6
char A20[] = {0, 21, 16 , 20};
char A21[] = {4, 22 , 16 , 21};
char A22[] = {2, 23 , 16 , 22};
char A23[] = {0, 24 , 16 , 23};
//pin 7
char A24[] = {1, 25 , 16 , 24};
char A25[] = {1, 26 , 16 , 25};
char A26[] = {8, 27 , 16 , 26};
char A27[] = {4, 28 , 16 , 27};
//pin 8
char A28[] = {0, 29 , 16 , 28};
```

```
char A29[] = {7, 30 , 16 , 29};
char A30[] = {7, 31 , 16 , 30};
char A31[] = {9, 32 , 16 , 31};
 //pin 9
char A32[] = {2, 33 , 16 , 32};
char A33[] = {0, 34 , 16 , 33};
char A34[] = {8, 35 , 16 , 34};
char A35[] = {3, 36 , 16 , 35};
//pin 10
char A36[] = {1, 37 , 16 , 36};
char A37[] = {7, 38 , 16 , 37};
char A38[] = {8, 39 , 16 , 38};
char A39[] = {4, 40 , 16 , 39};
//pin 11
char A40[] = {6, 41 , 16 , 40};
char A41[] = {4, 42 , 16 , 41};
char A42[] = {2, 43 , 16 , 42};
char A43[] = {5, 44 , 16 , 43};
//pin 12
char A44[] = {5, 45 , 16 , 44};
char A45[] = {3, 46 , 16 , 45};
char A46[] = {0, 47 , 16 , 46};
char A47[] = {1, 48 , 16 , 47};
 //pin 13
char A48[] = {0, 49 , 16 , 48};
char A49[] = {6, 50 , 16 , 49};
char A50[] = {3, 51 , 16 , 50};
char A51[] = {7, 52 , 16 , 51};
//pin 14
char A52[] = {9, 53 , 16 , 52};
char A53[] = {1, 54 , 16 , 53};
char A54[] = {0, 55 , 16 , 54};
char A55[] = {7, 56 , 16 , 55};
//pin 15
char A56[] = {5, 57 , 16 , 56};
char A57[] = {7, 58 , 16 , 57};
char A58[] = {6, 59 , 16 , 58};
char A59[] = {6, 60 , 16 , 59};
//pin 16
char A60[] = {3, 61 , 16 , 60};
char A61[] = {9, 62 , 16 , 61};
char A62[] = {1, 63 , 16 , 62};
char A63[] = {1, 64 , 16 , 63};
//pin 17
char A64[] = {0, 65 , 16 , 64};
char A65[] = {0, 66 , 16 , 65};
char A66[] = {1, 67 , 16 , 66};
char A67[] = {2, 68 , 16 , 67};
//pin 18
char A68[] = {3, 69 , 16 , 68};
char A69[] = {3, 70 , 16 , 69};
char A70[] = {2, 71 , 16 , 70};
char A71[] = {9, 72 , 16 , 71};
 //pin 19
char A72[] = {0, 73 , 16 , 72};
char A73[] = {2, 74 , 16 , 73};
```

```
char A74[] = {2, 75 , 16 , 74};
char A75[] = {7, 76 , 16 , 75};
//pin 20
char A76[] = {3, 77 , 16 , 76};
char A77[] = {9, 78 , 16 , 77};
char A78[] = {2, 79 , 16 , 78};
char A79[] = {0, 80 , 16 , 79};
char A80[]={16,80};  //terminator

void card_number_goes_to_pin(void)   //am ales valori ale nr_card=[03,17] deoarece cifra 0x00(nr_card) nu era luată in calcul
{
   switch(PIND)   //switch pentru a afla pinul corespunzător fiecarui cont , in funcție de nr_card introdus se trece la starea Q
corespunzătoare
          {
          case 0x01: Q=0;
          break;
          case 0x02: Q=4;
          break;
          case 0x03: Q=8;
          break;
          case 0x04: Q=12;
          break;
          case 0x09: Q=16;
           break;
          case 0x0A: Q=20;
          break;
          case 0x0B: Q=24;
          break;
          case 0x0C: Q=28;
          break;
          case 0x0D: Q=32;
          break;
          case 0x0E: Q=36;
          break;
          case 0x11: Q=40;
          break;
          case 0x12: Q=44;
          break;
          case 0x13: Q=48;
          break;
          case 0x39: Q=52;
          break;
          case 0x3A: Q=56;
          break;
          case 0x3B: Q=60;
          break;
          case 0x3C: Q=64;
          break;
          case 0x3D: Q=68;
          break;
          case 0x3E: Q=72;
          break;
          case 0x3F: Q=76;
          break;
          }
}
```

```
void PIN_incorect(void)
{
   char x = 0xFF;  //inițial led oprit
   char cnt =0;
   while (cnt <30)  //30 intreruperi = 3s -->1s = 10 intreruperi
   {

      if ( in == 0x40 )   // dacă SW6 pentru incheierea pinului este aprins se execută
      {
         switch(x)
         {
         case 0xFF: x=0xF7; PORTB = x;   //LED3 aprins
         break;
         case 0xF7: x=0xFF; PORTB = x;  //LED3 stins
         break;
         }
      }
      cnt ++;
   }
}

//tabela de adevar CLC
char TAB[8] = {0xFF, 0xFE, 0xFC, 0xFF, 0xFF, 0xFA, 0xF8, 0xFF};  //SOLD,RETRAGERE,CHITANȚĂ sunt active in 0


//fct CLC
void clc(void)
{
 char tmp;
 tmp = in & 0x07; //masca 0x07, SOLD-SW0,RETRAGERE-SW1,CHITANȚĂ-SW2
 out = TAB[tmp];  //acțiunile sunt reprezentate prin LED-urile 0,1,2
   PORTB = out;
}


//fct 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;
          }
}


// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
```

```
{
 // Reinitialize Timer 0 value
 TCNT0 = 0x3C;

 // Place your code here

 //initializari variabile

 in = PIND;
 switch (S)
            {
 case 0:
            if(in == 0x80) {S = 1;}  //SW7  introducere pin
   break;
            case 1:
       eroare=1;    //resetare eroare
       if( Q == 0 )
       {
          card_number_goes_to_pin();
          Q1=Q;
       }
            cls(); //se testeaza pin
            if(in == 0x40) {S = 2;} //SW6   incheiere pin
 break;
 case 2:
            if((Q1+4)==Q)   //comanda Q%4==0 nu era bună deoarece la introducerea greșită a tuturor cifrelor intră in CLC
                        {
                        PORTB = PORTB | 0x0F;
                        clc();
                        if(in == 0x08) {S = 3;} //stop clc nu se mai solicită nicio comandă si se trece in următoarea stare
                        }
            else
                        {
                        PIN_incorect();   //se trece in funcția pentru pin incorect
          Q = 0;
                        S = 0;      //se reseteaza stare S și starea Q pentru CLS
                        eroare = 0;   //eroarea devine 0-eroare introducere pin
          PIND = 0x00;   //se alctualizează toate SW
                        }
 break;
 case 3:
            PORTB = PORTB | 0xFF; //stinge LED-uri
       PIND = 0x00;   //se actualizează toate SW
            S = 0;
       Q = 0;     //se modifică starea S si Q pentru a realua procesul daca se doreste folosirea altui cont
 break;
 PORTB = out;  //scrie ieșirea
 }


}

// Declare your global variables here

void main(void)
{
```

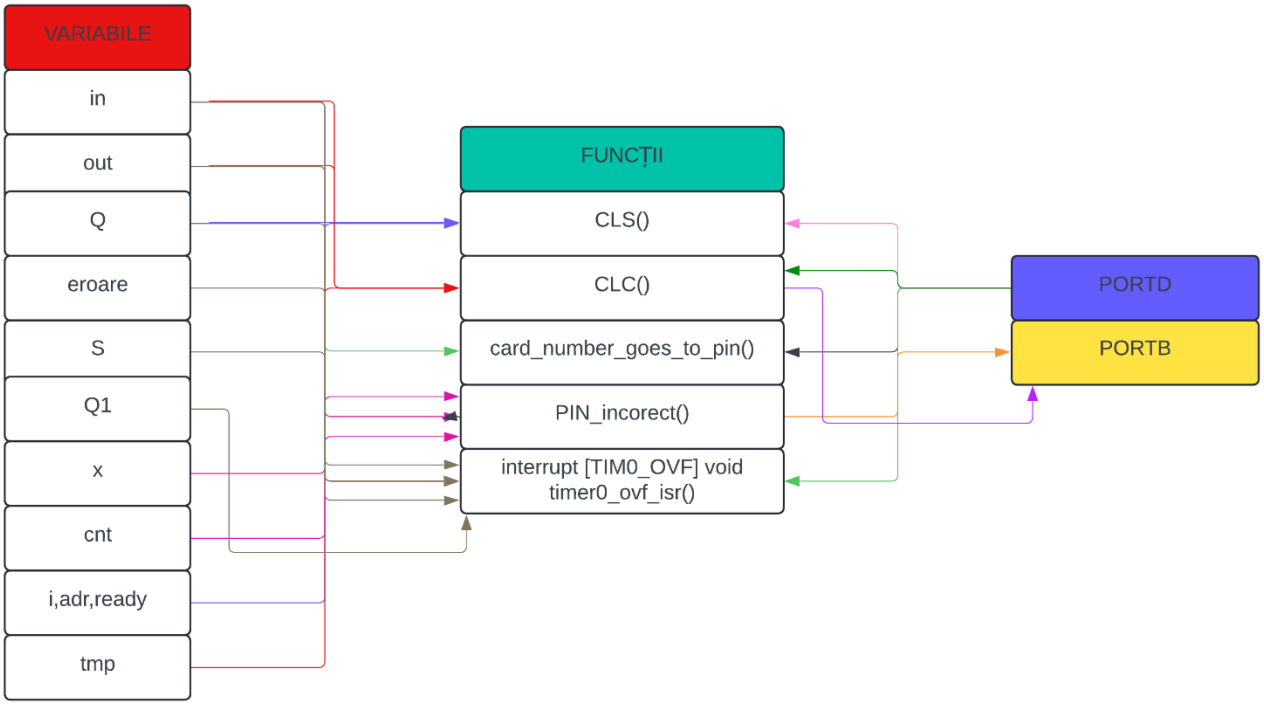
```
// Declare your local variables here
  eroare = 1;
PORTB = (eroare << 3) | 0xF7;   // LED3 pentru introducere incorectă PIN
  //inițializare adrese tabele de semnale relevante
TAB1[0] = A0;
TAB1[1] = A1;
TAB1[2] = A2;
TAB1[3] = A3;
TAB1[4] = A4;
 TAB1[5] = A5;
 TAB1[6] = A6;
 TAB1[7] = A7;
 TAB1[8] = A8;
 TAB1[9] = A9;
 TAB1[10] = A10;
 TAB1[11] = A11;
 TAB1[12] = A12;
 TAB1[13] = A13;
 TAB1[14] = A14;
 TAB1[15] = A15;
 TAB1[16] = A16;
 TAB1[17] = A17;
 TAB1[18] = A18;
 TAB1[19] = A19;
 TAB1[20] = A20;
 TAB1[21] = A21;
 TAB1[22] = A22;
 TAB1[23] = A23;
 TAB1[24] = A24;
 TAB1[25] = A25;
 TAB1[26] = A26;
 TAB1[27] = A27;
 TAB1[28] = A28;
 TAB1[29] = A29;
TAB1[30] = A30;
TAB1[31] = A31;
TAB1[32] = A32;
TAB1[33] = A33;
 TAB1[34] = A34;
 TAB1[35] = A35;
 TAB1[36] = A36;
 TAB1[37] = A37;
 TAB1[38] = A38;
 TAB1[39] = A39;
TAB1[40] = A40;
TAB1[41] = A41;
TAB1[42] = A42;
TAB1[43] = A43;
 TAB1[44] = A44;
 TAB1[45] = A45;
 TAB1[46] = A46;
 TAB1[47] = A47;
 TAB1[48] = A48;
 TAB1[49] = A49;
TAB1[50] = A50;
TAB1[51] = A51;
```

```
 TAB1[52] = A52;
 TAB1[53] = A53;
  TAB1[54] = A54;
  TAB1[55] = A55;
  TAB1[56] = A56;
  TAB1[57] = A57;
  TAB1[58] = A58;
  TAB1[59] = A59;
 TAB1[60] = A60;
 TAB1[61] = A61;
 TAB1[62] = A62;
 TAB1[63] = A63;
  TAB1[64] = A64;
  TAB1[65] = A65;
  TAB1[66] = A66;
  TAB1[67] = A67;
  TAB1[68] = A68;
  TAB1[69] = A69;
 TAB1[70] = A70;
 TAB1[70] = A70;
 TAB1[71] = A71;
 TAB1[72] = A72;
  TAB1[73] = A73;
  TAB1[74] = A74;
  TAB1[75] = A75;
  TAB1[76] = A76;
  TAB1[77] = A77;
  TAB1[78] = A78;
  TAB1[79] = A79;
  TAB1[80] = A80;


 S = 0;
 Q = 0;
 out = 0xFF;
}
```

# CAPITOLUL 6

Diagrame structurale corespunzătoare codului sursă




6.1. Diagramă structurală corespunzătoare codului sursă

Funcția CLS() este folosită pentru introducerea PIN-ului primind valorile fiecărei cifre din PORTD și conține variabilele {I,adr,ready,Q}.

Funcția CLC() este utilizată doar în starea S=2 pentru comenzile de tip sold, retragere, chitanță și primește valori din PORTD și afișează în PORTB cu ajutorul variabilelor {in,out,tmp}.

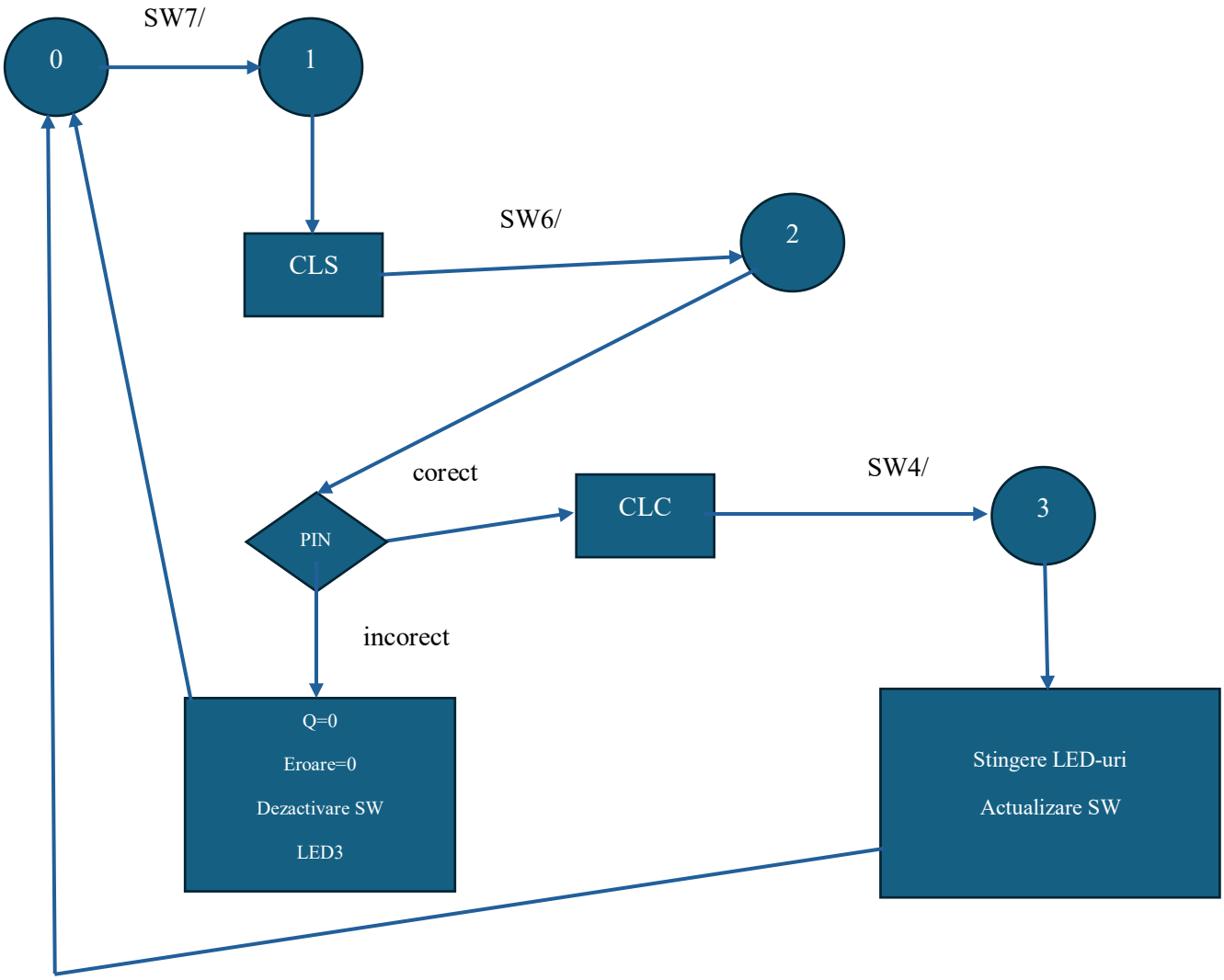
card_number_goes_to_pin() este o funcție în care se verifică numărul cardului și se trece la starea următoare Q pentru introducerea PIN-ului corespunzător.

Pentru aprindere/stingerea LED3 specific erorii se folosește funcția PIN_incorect() și are ca variabile {cnt,in,x}. Această funcție poate fi testată cu ajutorul opțiunii AUTOSTEP din AVRStudio.

Funcția interrupt[TIM0_OVF] void timer0_ovf_isr() este utilizată pentru trecerea sistemului dintr-o stare in alta în care se primesc si se afișează valori.

# CAPITOLUL 7

Diagrame secvențiale corespunzătoare codului sursă




7.1. Graful PS

Stările au următoarele semnificații :

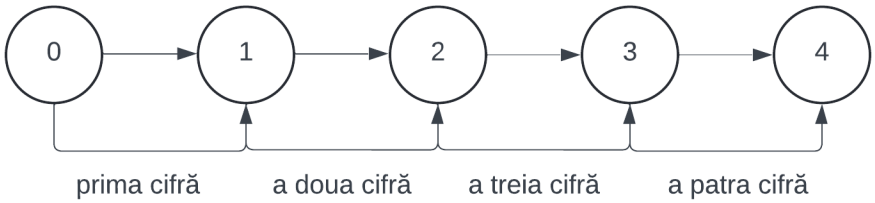
0 – stare inițială

1 – autentificare

2 – verificare PIN introdus

3 – stare finală (resetare sistem)

Inițial bancomatul se află în starea 0 în care așteaptă comenzi. La apăsarea SW7 se va trece într-o stare de autentificare în care se citește numărul cardului. La baza autentificării stă o funcție CLS() în care se testează introducerea fiecărei cifre în funcție de numărul cardului introdus.




7.2. Graful asociat CLS

Trecerea dintr-o stare în alta este realizată cu ajutorul SW3-SW0 pentru introducerea fiecărei cifre. În cazul în care cifra este corectă se modifică starea CLS , în caz contrar, starea rămâne aceeași .

După introducerea PIN-ului dorit, SW6 este apăsat pentru confirmarea acestuia și se testează introducerea corectă a codului PIN prin verificarea valorii stării Q a circuitului logic combinațional .În cazul în care s-a introdus o parolă eronată se aprinde/stinge intermitent LED3 timp de 3 secunde și se revine înapoi în starea inițială .

În cazul în care PIN-ul este corect se intră într-o funcție CLC() care efectuează operațiile de tip sold, retragere, chitanță. Combinațiile alese pentru CLC() sunt următoarele :

- SOLD ➔ RETRAGERE CARD

- RETRAGERE ➔ RETRAGERE CARD + RIDICARE BANCNOTE

- SOLD + CHITANȚĂ ➔ RETRAGERE CARD + CHITANȚĂ

- RETRAGERE + CHITANȚĂ ➔ RETRAGERE CARD + RIDICARE BANCNOTE + CHITANȚĂ

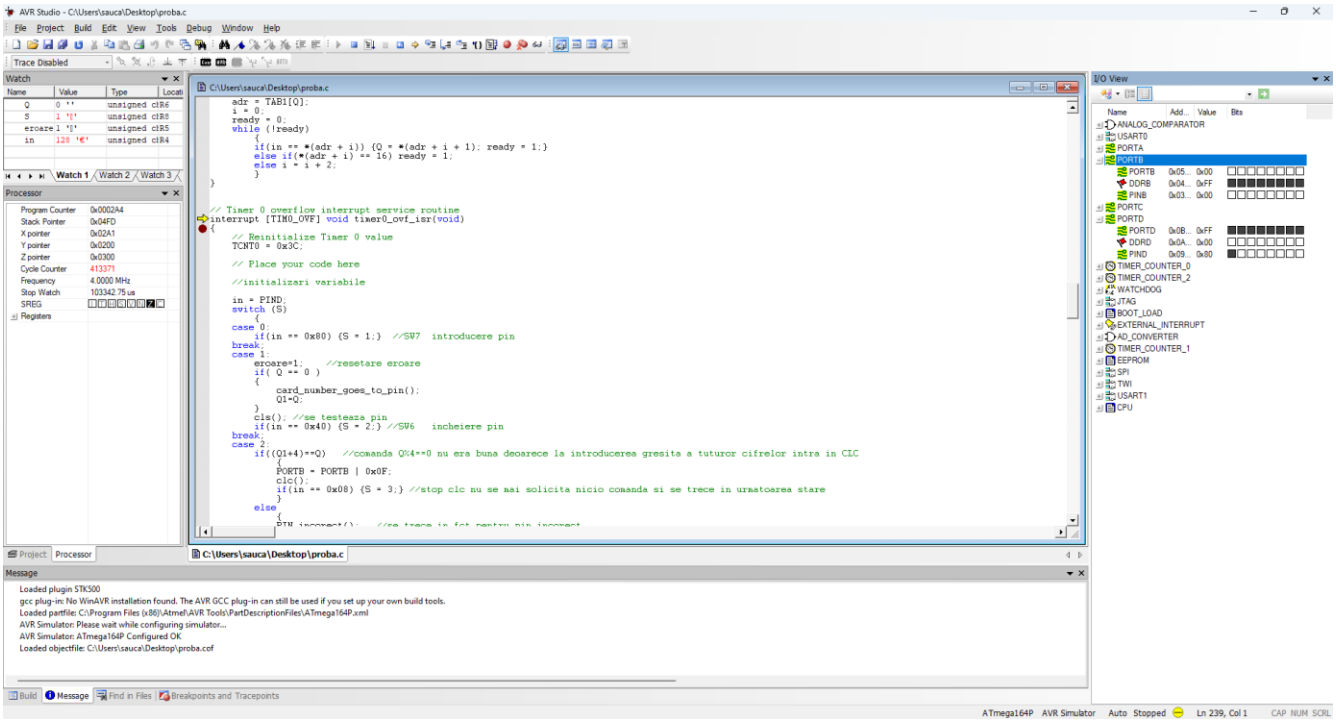
Orice altă combinație este ilustrată prin stingerea tuturor LED-urilor.

Se presupune ca utilizatorul nu mai dorește nici o altă tranzacție. Pentru confirmare se necesită apăsarea SW4 care modifică starea S. În această stare toate LED-urile se sting și se actualizează toate SW, urmând să se treacă înapoi la starea inițială

# CAPITOLUL 8

## Testarea funcționalității sistemului

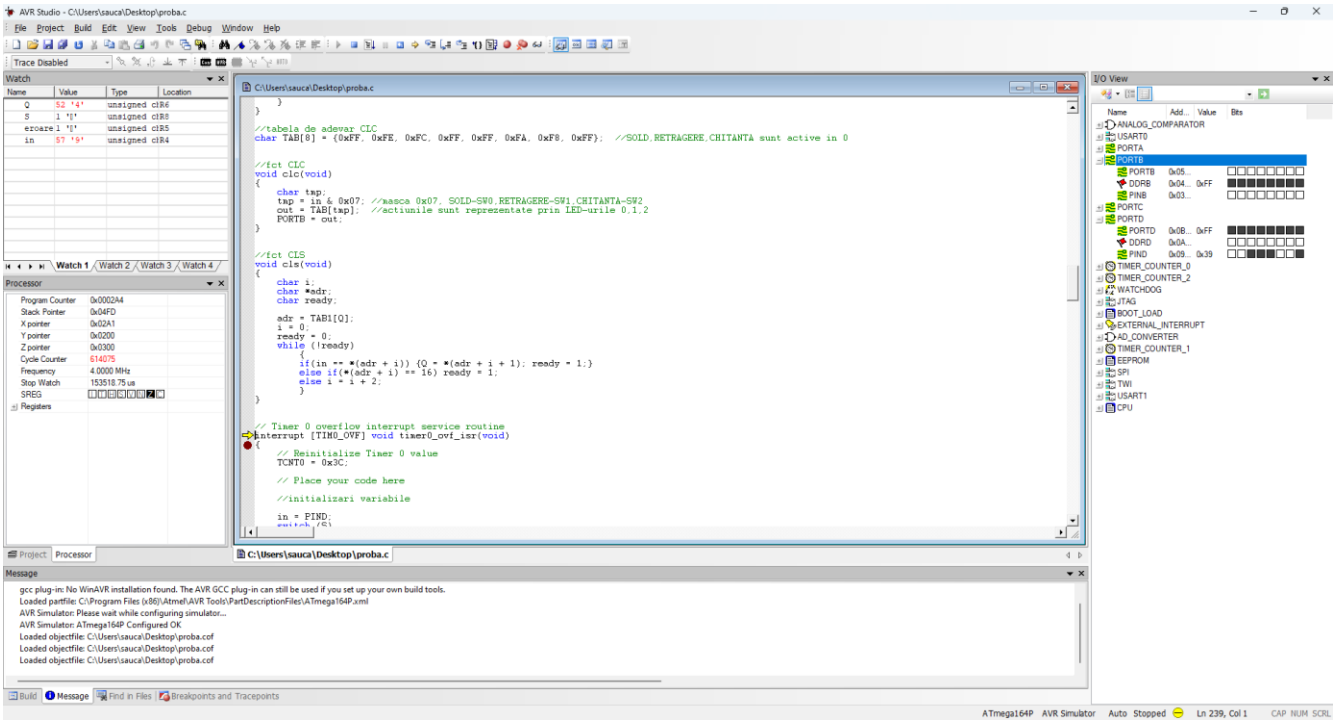
Inițial sistemul se află in starea 0 și așteaptă o comandă.




8.1.figura1

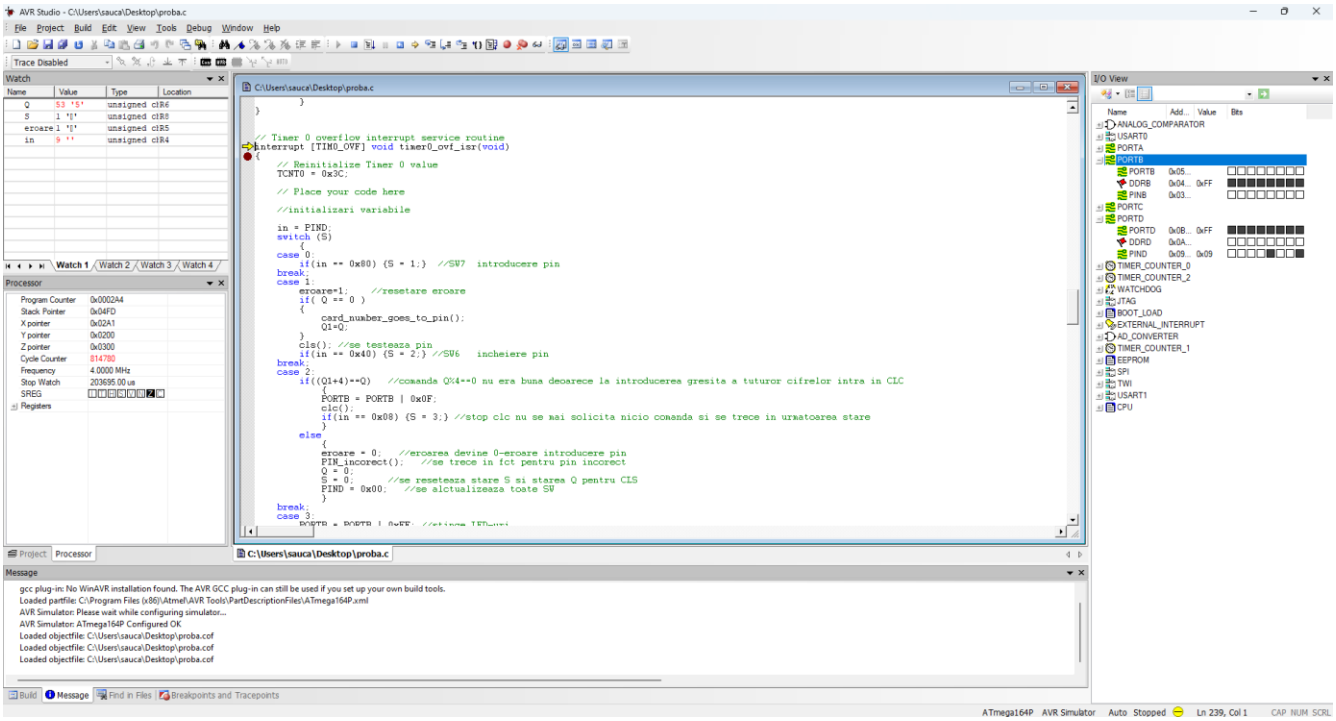
Se poate observa că prin apăsarea SW7(INTRODUCERE PIN) starea S a PS se modifică in 1.

Vom alege un număr de card din funcția card_number_goes_to_pin().



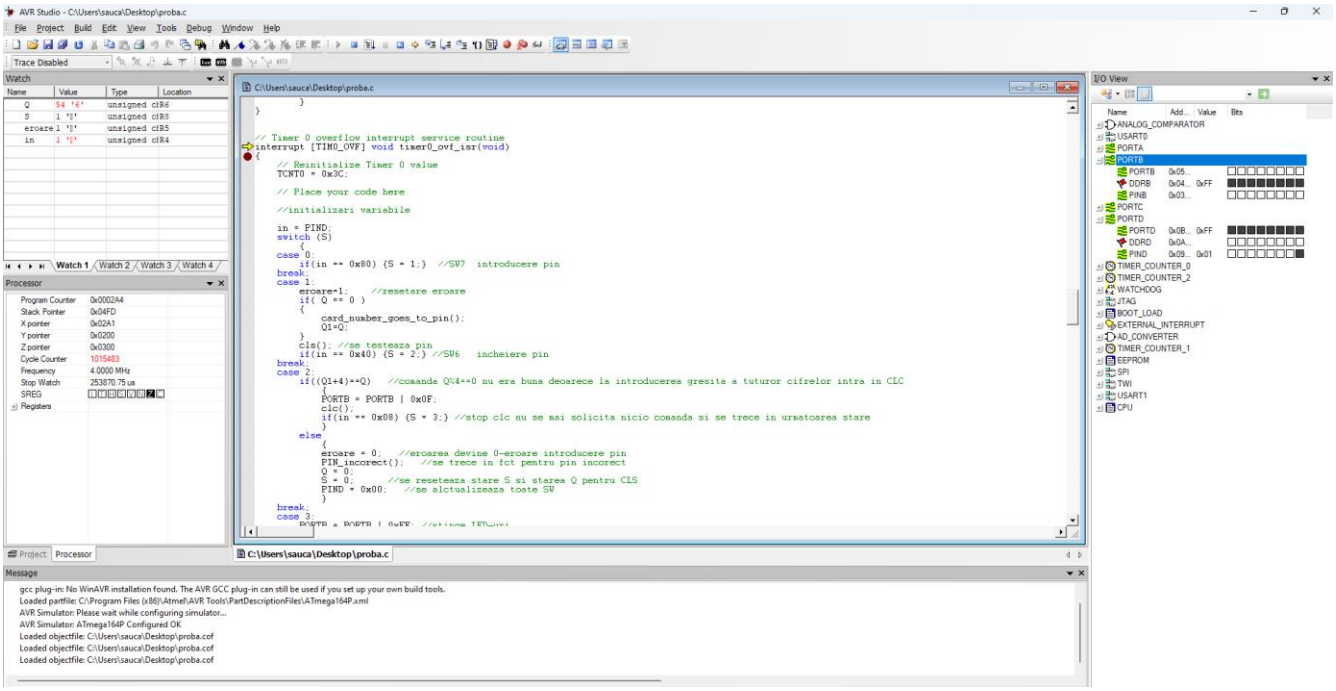
8.2.figura2

Am ales valoarea 0x39 pentru numărul cardului iar stare Q a CLS-ului s-a schimbat și a trecut la 52 pentru a se putea introduce PIN-ul corespunzător. Prima dată vom testa metoda introducerii PIN-ului corect.



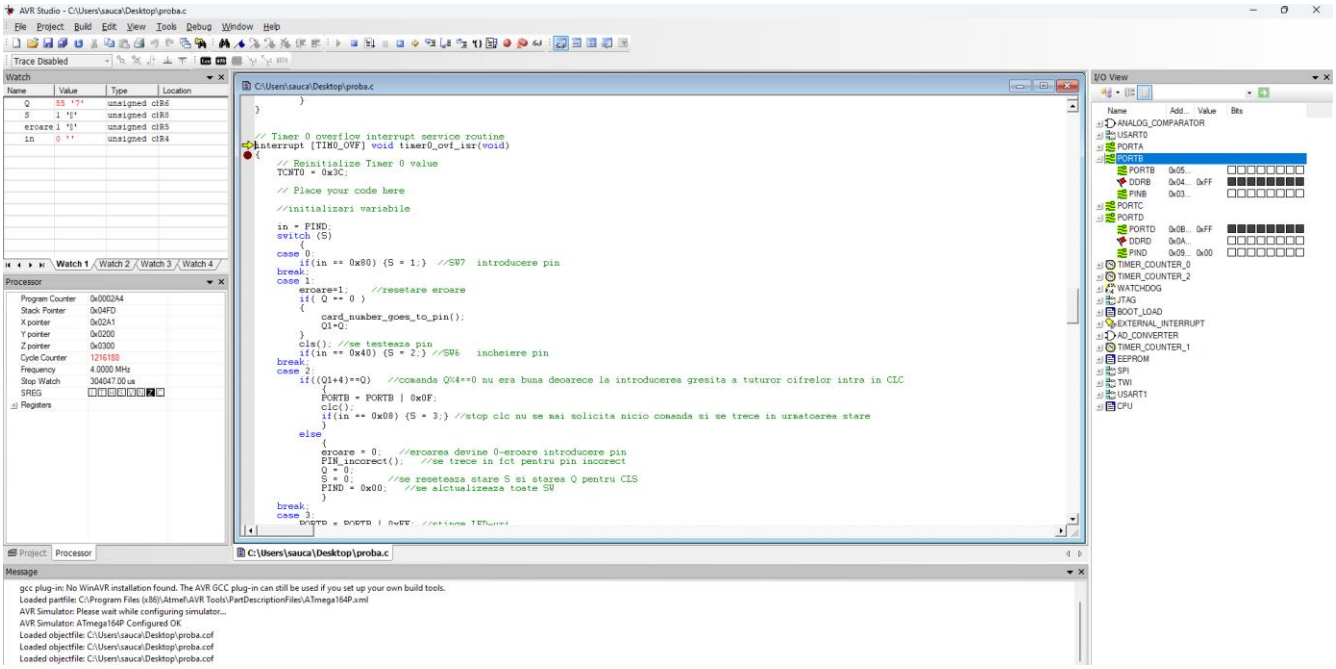
8.3.figura3

Prima cifră din PIN este 9, starea Q se modifică în 53.



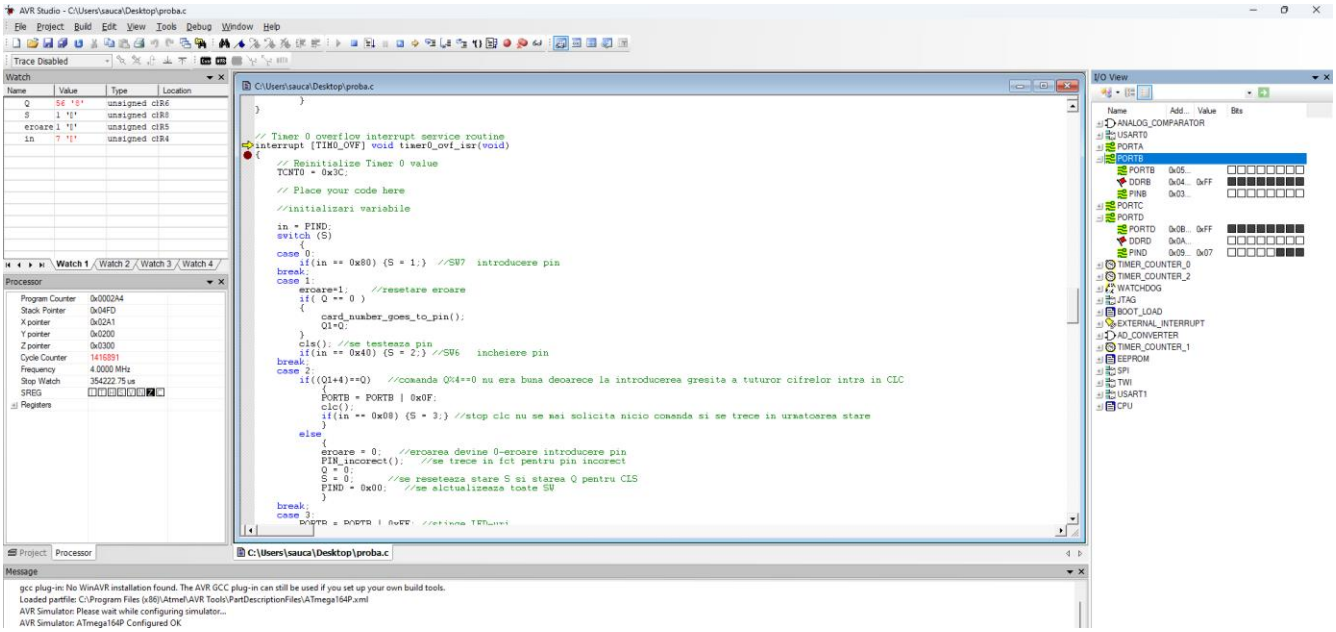
8.4.figura4

A doua cifră este 1, iar Q este 54.



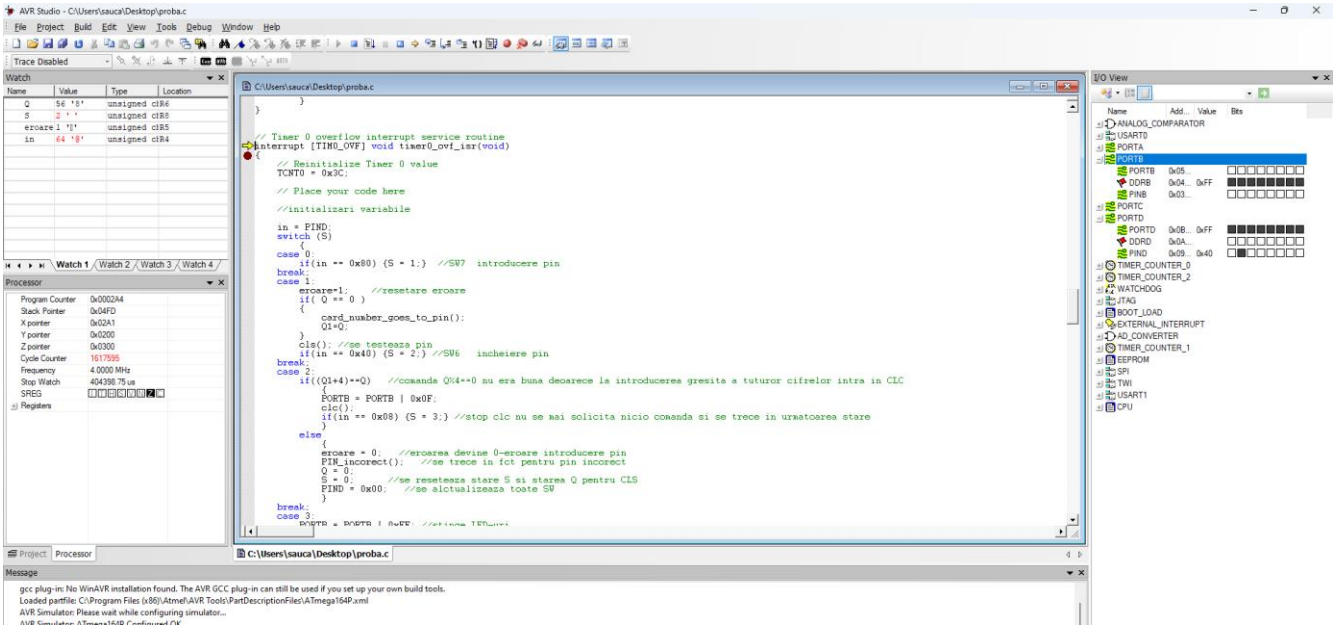
8.5.figura5

A treia cifră este 0 ,iar Q s-a modificat în 55.



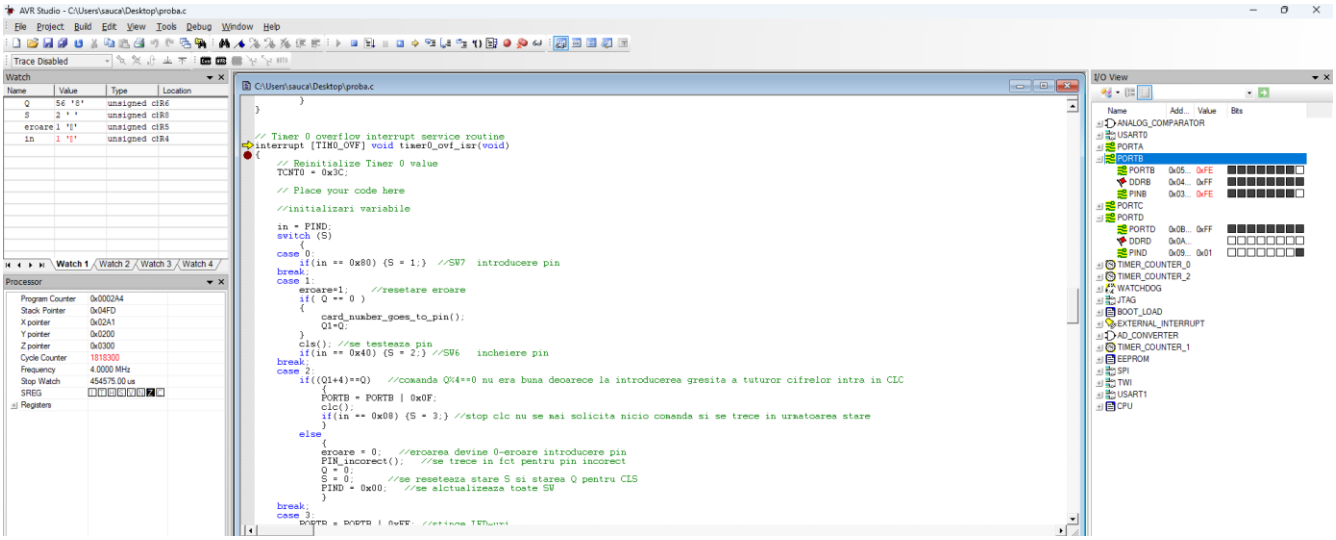
8.6.figura6

A patra cifră este 7 iar Q este 56.



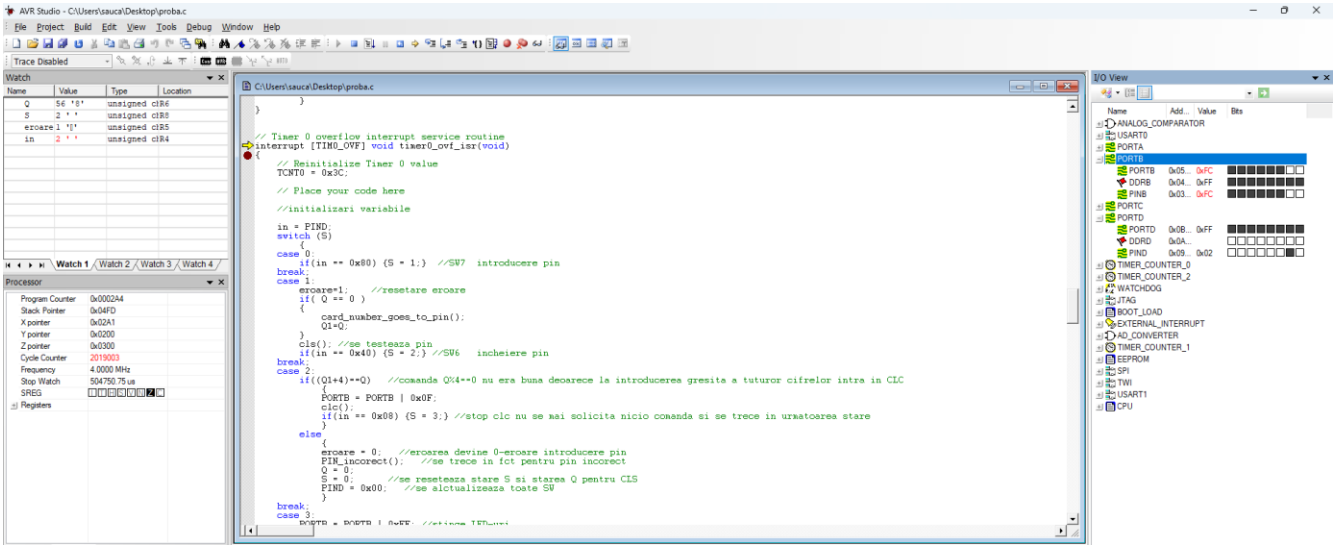
8.7.figura7

Prin apăsarea SW6 pentru confirmarea PIN-ului Q nu se modifica deoarece CLS-ul sare la terminator .Se modifică starea S în 2.



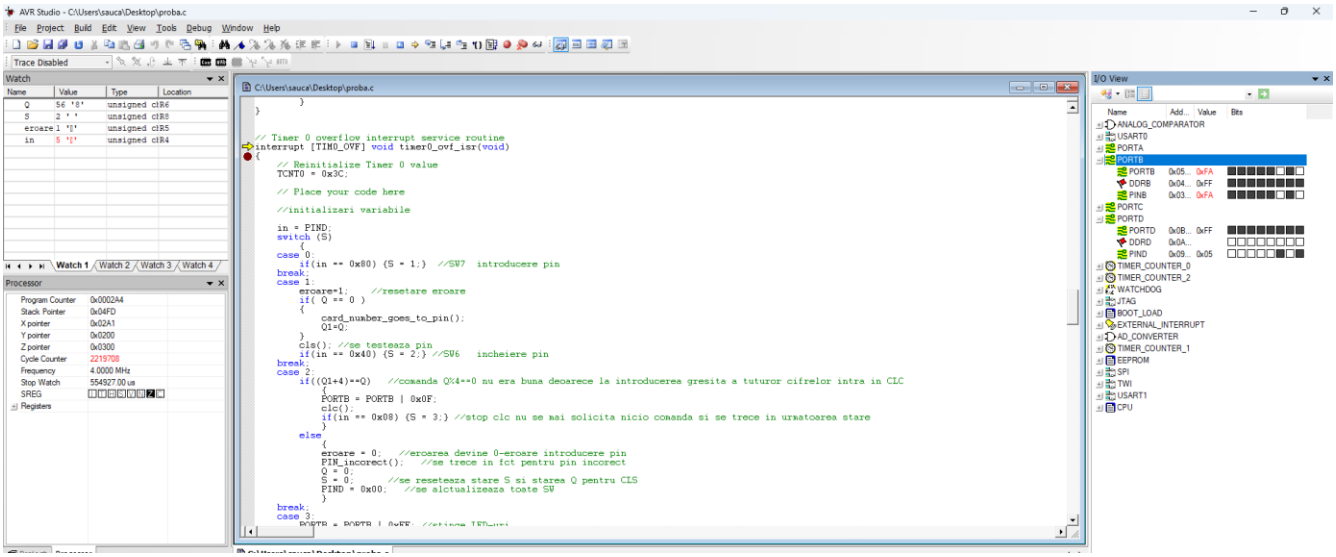

8.8.figura8

S-a selectat opțiunea pentru interogare sold, iar LED0 corespunzător s-a aprins.



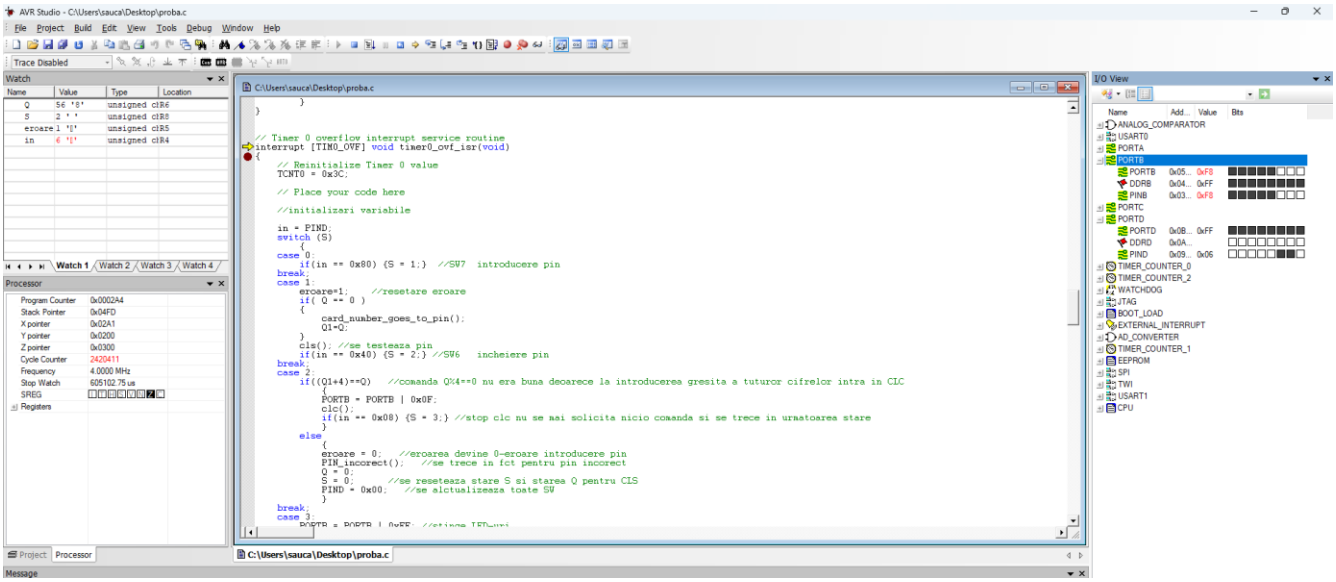

8.9.figura9

S-a selectat opțiunea pentru retragere și s-au aprins LED-urile corespunzătoare pentru retragere card și ridicare bancnote.



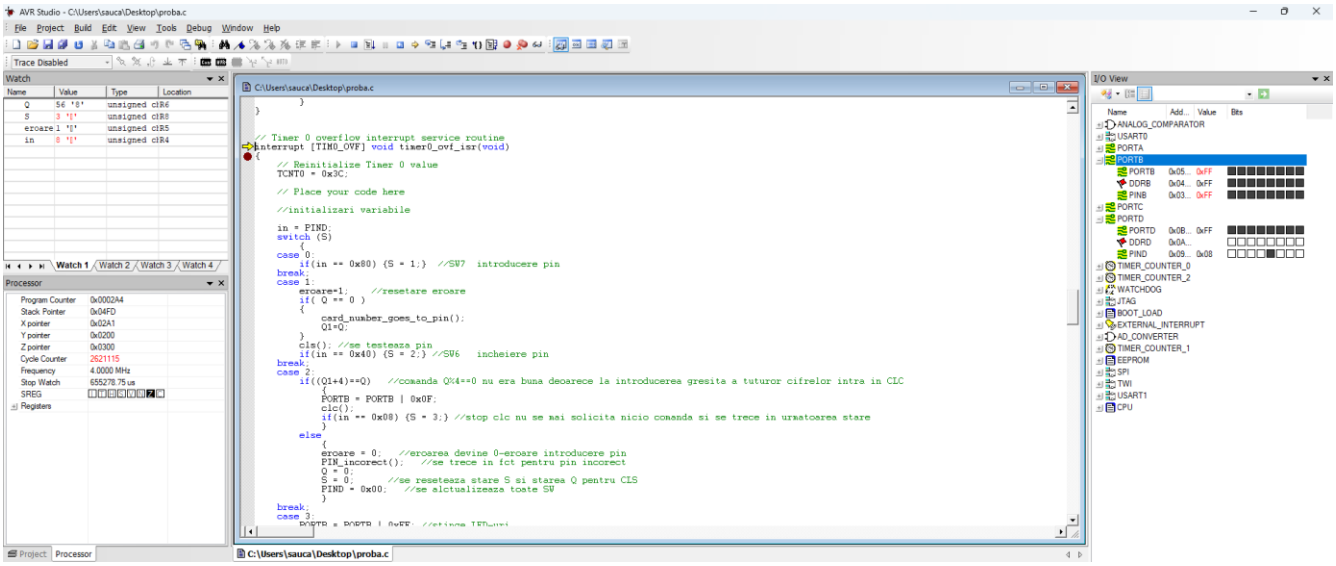
8.10.figura10

S-a selectat opțiunea pentru interogare sold cu chitanță și s-au aprins LED-urile 0 si 2.



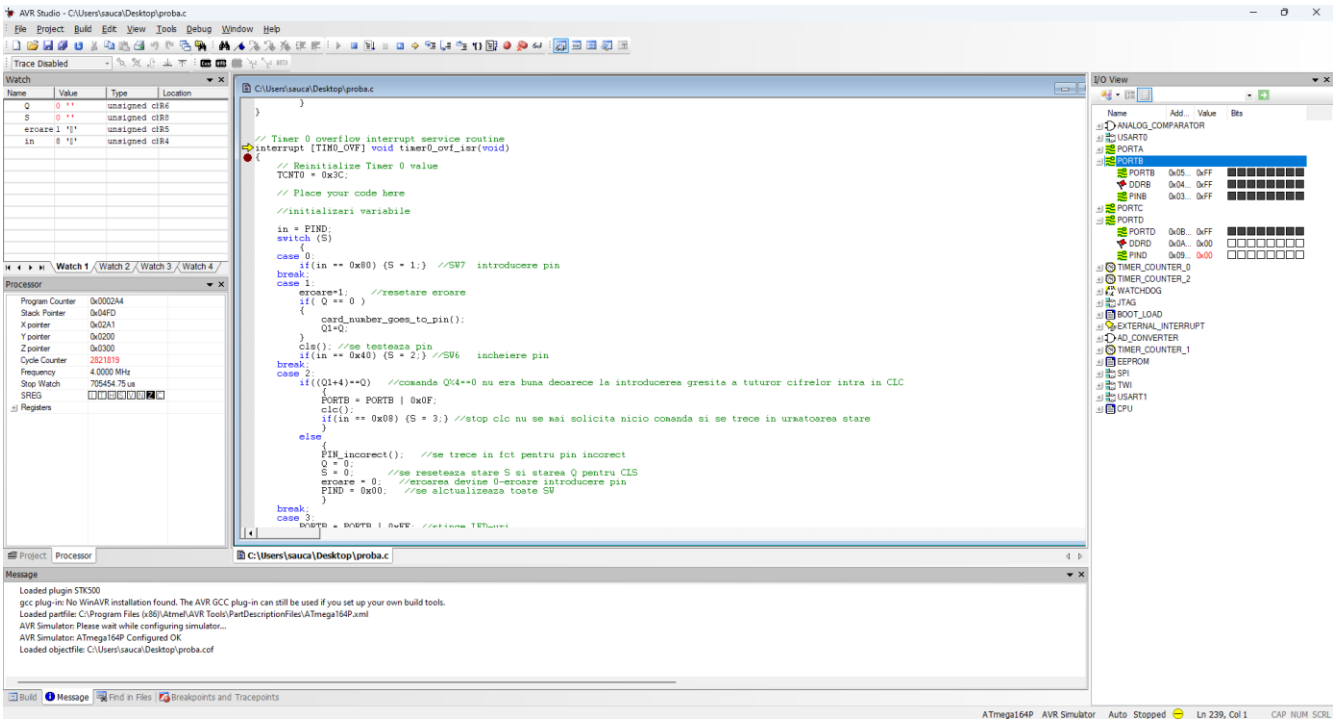
8.11.figura11

Se dorește retragere cu chitanță și s-au aprins LED2-LED0.



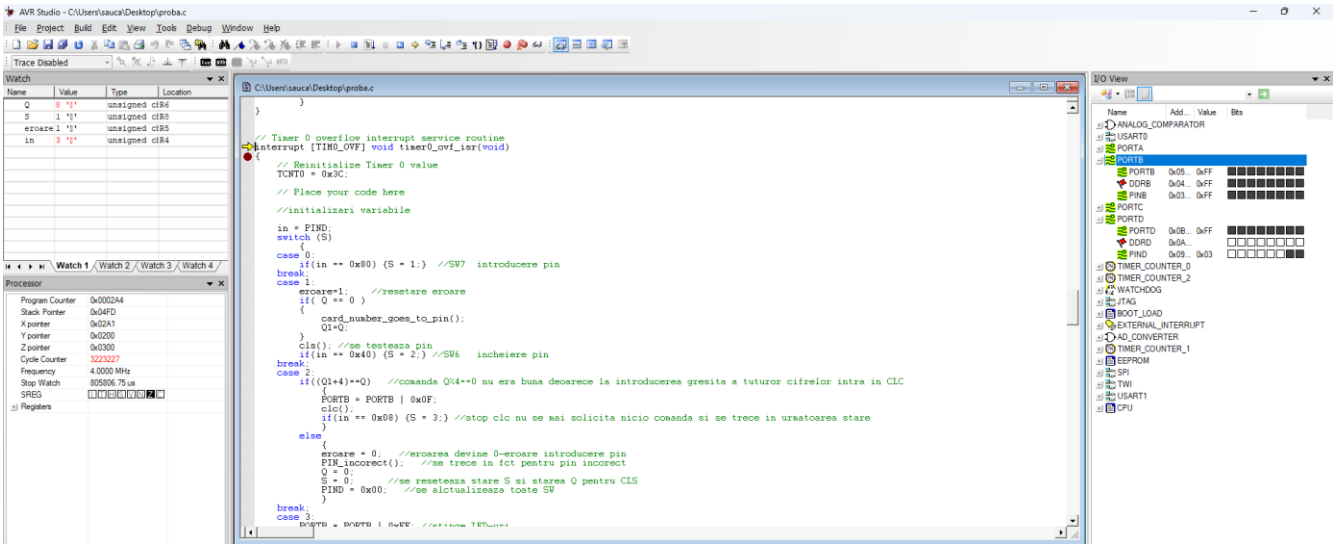
8.12.figura12

Prin apăsarea SW4 se confirmă ca nu se mai dorește nicio comandă, iar toate LED-urile se opresc, urmând ca apoi să se reseteze toate SW și toate valorile S și Q. Se trece înapoi în starea inițială.



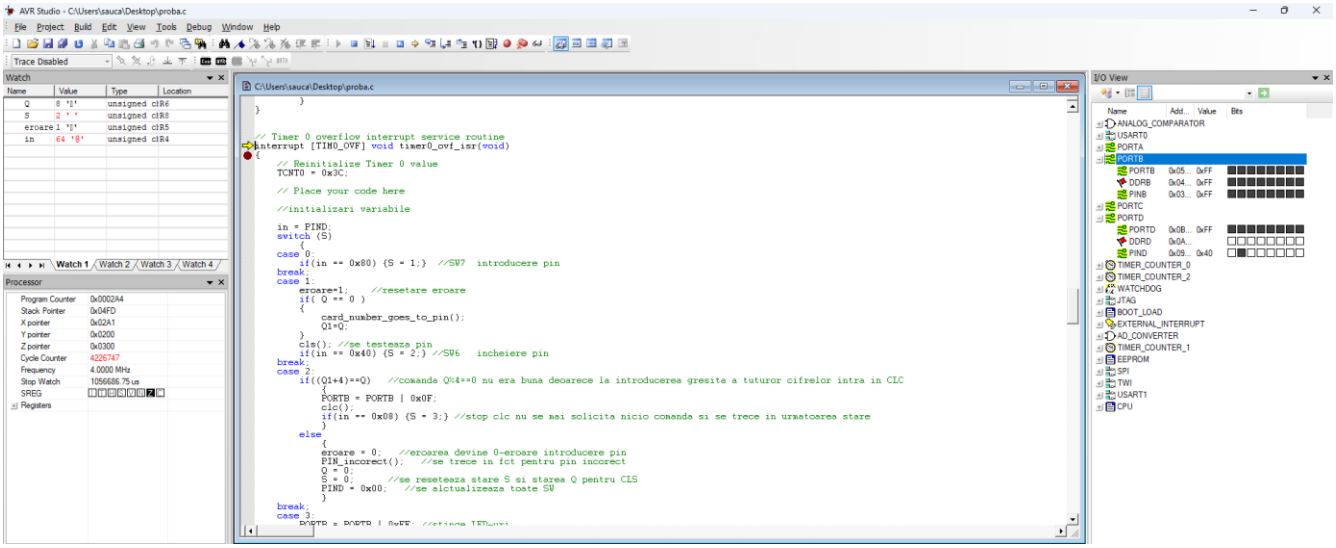
8.13.figura13

Acum se poate introducere un nou PIN pentru un nou utilizator.



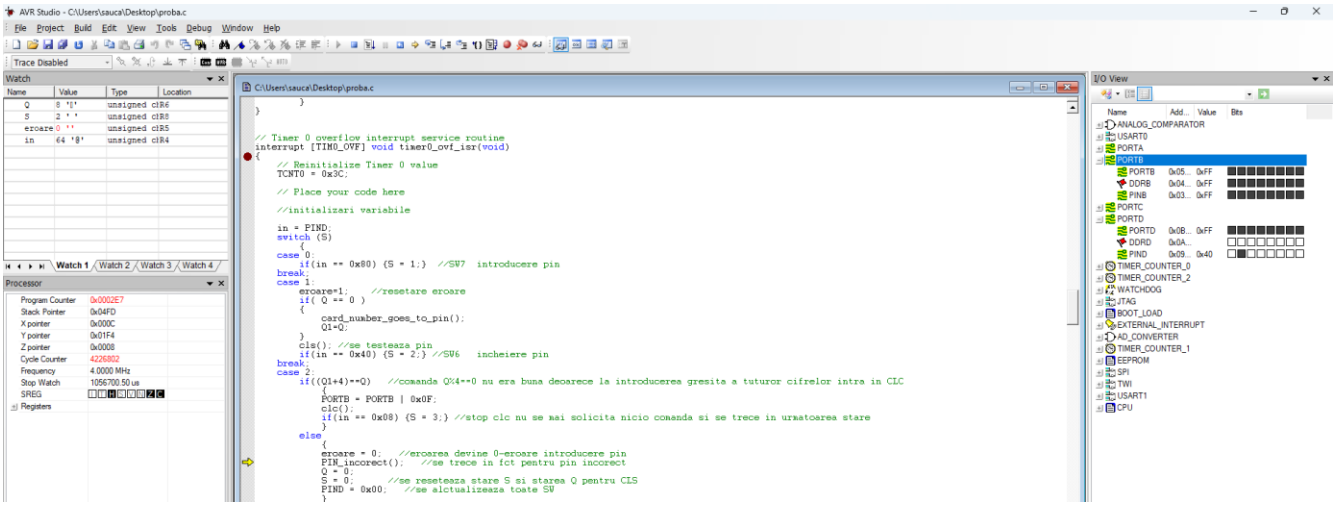
8.14.figura14

Acum se va testa funcționalitatea introducerii unui PIN greșit. PIN-ul corect pentru valoarea 0x03 a numărului de card este 7101.Se va introduce un PIN eronat(1234).



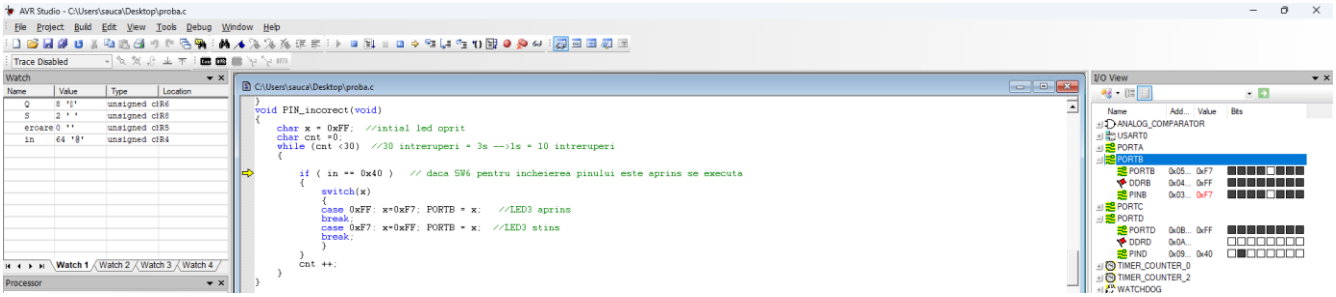
8.15.figura15

În urma introducerii PIN-ului eronat se poate observa că starea S s-a modificat în 2.



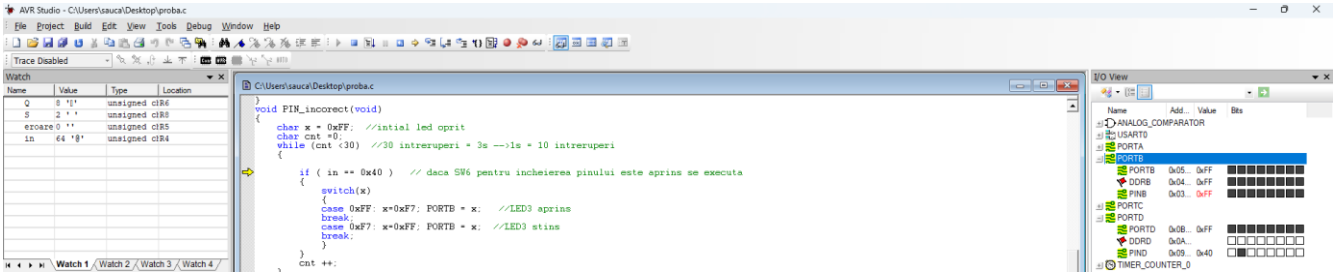
8.16.figura16

Eroarea s-a modificat în 0 , indicând introducerea unui PIN eronat, urmând să se intre în funcția care aprinde/stinge intermitent LED4 timp de 3s.



8.17.figura17

LED4 aprins.



8.18.figura18

LED4 stins.

Acest proces are loc timp de 3 secunde ,urmând să se oprească toate LED-urile și să se reseteze atât SW, cât și starea Q și S.

# CONCLUZIILE PROIECTULUI

În acest proiect s-a realizat implementarea software a unui bancomat utilizând microprocesorul Atmega164P cu un maxim de 20 de utilizatori. Programele cu care s-a reușit implementarea și testarea sistemului sunt CodeVisionAVR și AVR Studio.

Componentele cheie ale proiectului includ descrierea și utilizarea porturilor de ieșire și de intrare, proiectarea și utilizarea unei modalități de lucru adecvate, implementarea codului sursă și testarea întreg sistemului.

De asemenea, a fost foarte importantă partea de testare a codului sursă , deoarece acesta oferă sistemului o funcționalitate corectă,  fără erori.