Universitatea Națională de Știință și Tehnologie POLITEHNICA București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Sistem Automat de Sortare a Monedelor

Nume**:**Ghilință Liviu-Gabriel
Grupa:422E

Mai 2025

Cuprins

Lista figurilor

Tabele

Figuri

Introducere

Într-o societate în care automatizarea joacă un rol tot mai important, sortarea eficientă a monedelor devine o necesitate practică în diverse domenii.

Astfel, scopul principal al acestui proiect este realizarea software a unui sistem automat de sortare a monedelor.

Pentru scrierea codului am utilizat programul CodeVisionAVR V4.03, iar pentru testarea lui am utilizat AVR Studio. Pentru configurarea proiectului am ales un microcontroler de tip ATmega164 si frecvenţa ceasului de 10MHz.

În implementarea codului mi-am ales un set de valori predefinite ale dimensiunilor monedelor din Romania și anume monedele de 5,10 și 50 de bani. Dimensiunile au fost preluate după un site dedicat monedelor românești.

M-am folosit de un **proces secvențial** cu stări finite pentru respectarea pașilor de funcționare ale sortatorului. Așadar, **procesul secvențial** are 5 stări :

- Starea 0 echivalentă stării de Așteptare
- Starea 1 echivalentă stării de Măsurare
- Starea 2 echivalentă stării de Comparare
- Starea 3 echivalentă stării de Direcționare
- Starea 4 echivalentă stării de Reset

Pentru măsurare folosesc 2 senzori , un senzor optic și un senzor de greutate pentru transmiterea informației către microcontroler despre diametrul si masa monedei masurate. Conexiunea dintre microcontroler si senzori se face prin pinul C, pentru senzorul optic si prin pinul D, pentru senzorul de greutate.

Compararea,respectiv identificarea monedelor se face la nivel intern printr-un **circuit logic combinațional** prin realizarea unor operații logice între valorilor predefinite cu cele receptate prin cei 2 senzori.

Direcționarea constă în trimiterea respectivei monede în zona corespunzatoare. La nivel software acest lucru se face printr-un **circuit logic secvenția**l care într-un final realizează incrementarea unui contor, dedicat fiecărei monede in parte. Totodată, în această etapă are loc și afișarea valorii monedei colectate, care se face prin aprinderea anumitor leduri si al unui afișaj cu 7 segmente.

Reset-ul se face prin revenirea în starea de așteptare.

Capitolul I Alegerea porturilor

Alegerea porturilor reprezintă primul pas din realizarea proiectului.

Pentru a realiza conexiunea dintre microcontroler și dispozitivele externe am ales să utilizez Port-urile astfel:

Port	Direcție	Funcție
Port A	Ieșire	Afișare pe Afisaj 7 seg
Port B	Ieșire	Afișare eroare
Port C	Intrare	Citire diametru monedă
Port D	Intrare	Citire masă monedă

Tabelul 1.1.Descrierea funcţionalităţii porturilor de intrare şi iesire

Port A							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Out	Out	Out	Out	Out	Out	Out	Out
1	1	1	1	1	1	1	1

Tabelul 1.2 Configurarea portului A

Port B							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Out	Out	Out	Out	Out	Out	Out	Out
1	1	1	1	1	1	1	1

Tabelul 1.3 Configurarea portului B

Port C							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
In	In	In	In	In	In	In	In
P	P	P	P	P	P	P	P

Tabelul 1.4 Configurarea portului C

Port D							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
In	In	In	In	In	In	In	In
P	P	P	P	P	P	P	P

Tabelul 1.5 Configurarea portului D

Pentru configurarea porturilor am folosit logica Pull-Up (active in 0).

- Prin Port-ul A se realizează conexiunea dintre microcontroler si afișajul cu 7 segmente. Afișarea valorii ultimei monede introduse pe un afișaj cu 7 segmente se realizează conform: *Tabelului 1.7*.
- Prin Port-ul B se realizează conexiunea dintre microcontroler si LED-uri, respectiv Buzzer. Afișarea valorii ultimei monede introduse prin aprinderea LED-urilor se realizează conform unei configurații binare specifice fiecărei monede.*(Tabelului 1.6)*

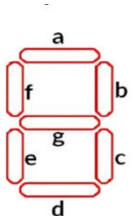
Monedă	**SW7**	**SW6**	**SW5**	**SW4**	**SW3**	**SW2**	**SW1**	**SW0**
5	1	1	1	1	1	0	1	0
10	1	1	1	1	0	1	0	1
50	1	1	0	0	1	1	0	1
Eroare	0	1	1	1	1	1	1	0

Tabelul 1.6. Configuratii binare

Alături de afișarea valorii prin led-urile am decis să utilizez și un afișaj cu 7 segmente pentru o identificare in zecimal a valorii monedei introduse. Afișajul cu 7 segmente a fost configurat astfel:

a	b	c	d	e	f	g	.
SW7	**SW6**	**SW5**	**SW4**	**SW3**	**SW2**	**SW1**	**SW0**

Tabelul 1.7. Configurare afișaj cu 7 segmente



- Prin Port-ul C se face conexiunea dintre microcontroler si senzorul optic, care transmite o tensiune specifică fiecărui diametru.
- Prin Port-ul D se face conexiunea dintre microcontroler si senzorul de greutate, care transmite o tensiune specifică fiecărei mase.

Capitolul II Schema de bloc

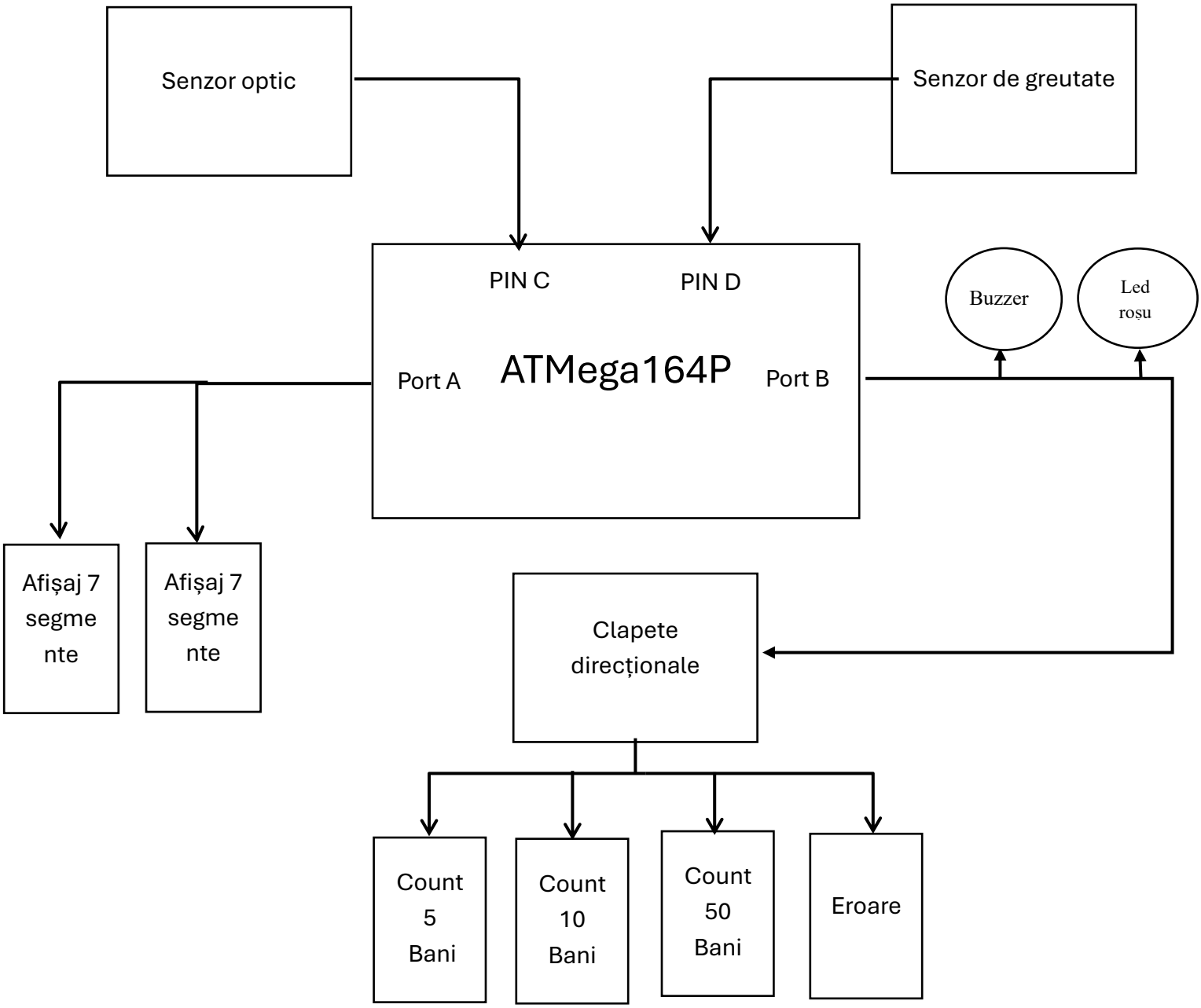


Fig. 2.1 Schema de bloc

Microcontrolerul Atmega164P are frecvenţa ceasului setată la 10MHz.

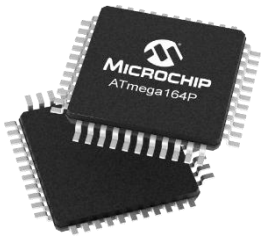


Fig.2.2 Microcontrolerul Atmega164P

Fig.2.3 Afişaj cu 7 segmente

Fig.2.4 Senzor optic

Fig.2.5 Senzor de greutate

Fig 2.6 Led roşu

Fig.2.7 Buzzer

Capitolul III. Descrierea cazurilor de utilizare a sistemului

Întregul process de sortare a monedelor cuprinde următoarele etape:

1. Așteptarea inserării unei monede.
2. Inserarea monedei.
3. Măsurarea monedei.
4. Identificarea monedei prin compararea cu valorile finite.
5. Direcționarea monedei spre compartimentul predestinat
6. Afișare pe 7 segmente + LED-uri în cazul în care moneda este validă

6.1. Aprindere LED roșu și Buzzer in cazul în care moneda este invalidă

7. Revenire în starea inițială de așteptare

În așteptarea inserării unei monede sistemul nu realizează nicio operație.

Prin măsurarea monedei sistemul primește informațiile despre moneda introdusă. Tot în această etapă la nivel software se face memorarea valorilor într-un set de variabile de care mă voi folosi pentru etapele următoare.

Identificarea monedei este realizată printr-un **circuit logic combinațional** care compară valorile receptate de senzorii aferenți și a valorilor standard aferente monedelor, definite la începutul programului și prin atribuirea unui id(id_moneda) fiecărei monede introduse.Diametrul si masa monedelor diferă de la o monedă la alta , astfel :

- Moneda de 5 bani are diametrul de 18mm (aproximat prin adaos) și masa de 3g.
- Moneda de 10 bani are diametrul de 21mm (aproximat prin adaos) și masa de 4g.
- Moneda de 50 bani are diametrul de 24mm (aproximat prin adaos) și masa de 6g.

Direcționarea monedei spre compartimentul predestinat este rezultată printr-un **circuit logic secvențial** prin care are loc incrementarea unor contoare(counter5,counter10,counter50 și eroare).

Afișarea pe 7 segmente se face prin intermediul portului A. Iar, aprinderea LED-urilor ,respectiv LED-ului roșu și a buzzer în care moneda este invalidă, se face prin intermediul portului B.

Revenirea in starea inițială se face prin resetarea tuturor variabilelor utilizate de-a lungul **FSM**-ului.

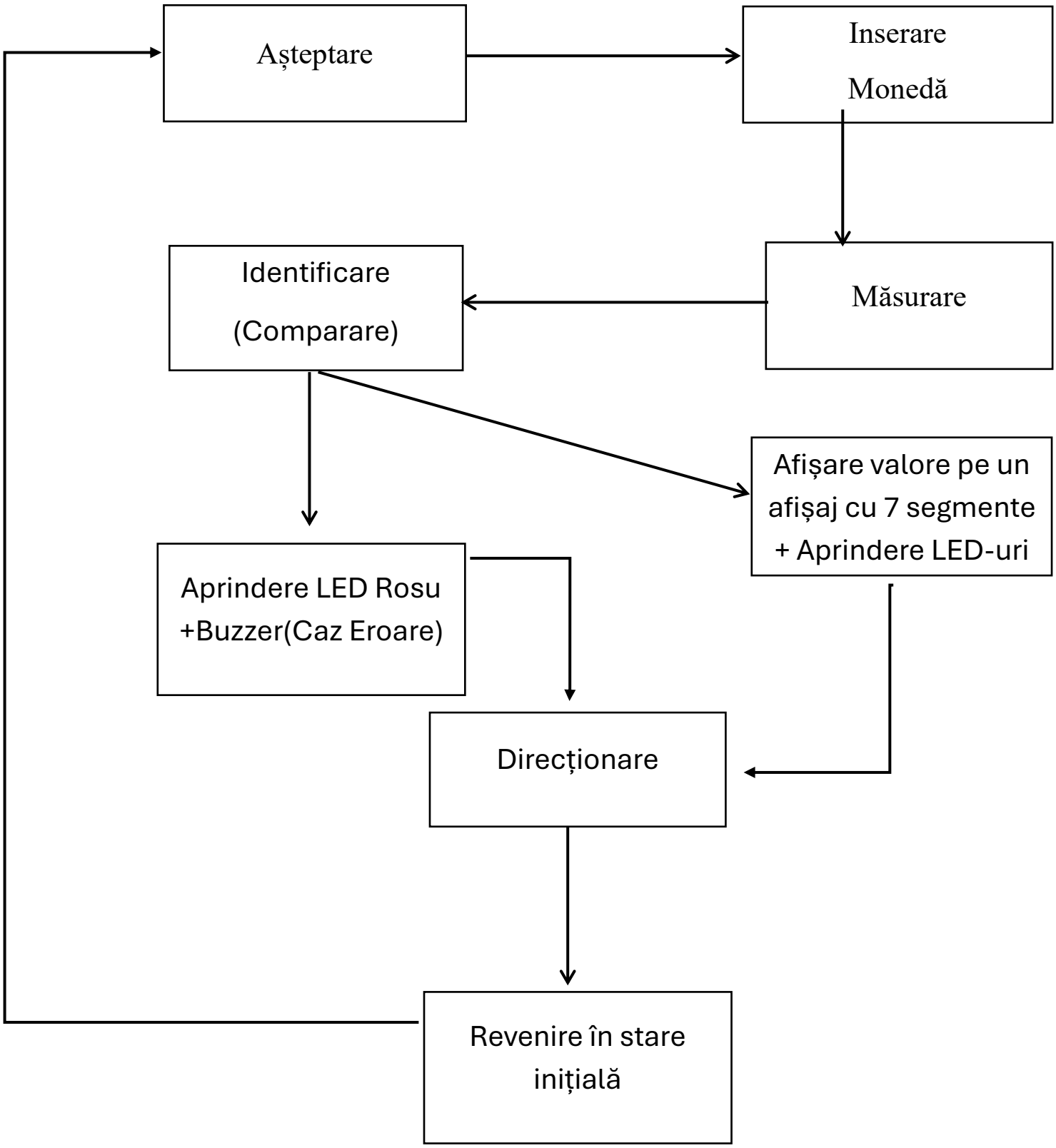


Fig.3.1 Diagrama cazurilor de utilizare

Capitolul IV Implementarea codului sursă

Pentru proiectarea codului sursă am parcurs următorii pași:

Pasul 1. Definirea unor măștilor. În acest pas mi-am definit un set de măști(valori) corespunzătoare fiecărei dimensiuni în parte , ca de exemplu : pentru diametrul de 24 de mm mi-am definit M24.

Pasul 2. Declararea variabilelor. În acest pas mi-am ales un set de variabile cu care am reușit să reproduc funcțional ce face un sortator de monede. Variabilele alese au fost de tip **CHAR** pentru ce primește / transmite microcontrolerul, respectiv pentru stările **FSM**-ului și de tip **INT** pentru simularea unor zone de stocare ale memoriilor.Totodată, am declarat un set de pointeri cu care am implementat **circuitul secvențial**.

Pasul 3. Am folosit un **SWITCH-CASE** pentru realizarea **procesului secvențial** (**FSM**-ului), unde fiecare **CASE** corespunde unei stări. Parcurgerea FSM-ului am realizat-o printr-o variabila de tip **CHAR**(*stare_moneda*).

Pasul 4. Am realizat o serie de **IF-ELSE**-uri pentru fiecare caz(stare) în parte pentru a determina valoarea monedei ,atribuirea unui id specific(*id_moneda*) și direcționarea în compartimentul destinație. În această etapă am inclus și un **circuit logic combinațional**, respectiv un **circuit logic secvențial**. **Circuitul logic combinațional** a fost realizat prin verificarea egalității simultane între valorile măsurate si valorile prestabilite. Iar, **circuitul logic secvențial** a fost realizat prin realizarea unor stări specifice fiecărui contor(compartiment).

Pasul 5. Afișarea am realizat-o în momentul în caret este cunoscut id-ul si dimensiunile monedei.Pentru aceasta am utilizat un vector de tip **CHAR** pentru afișarea pe LED-uri a valorii , iar pentru afișajul cu 7 segmente am utilizat un set de varbile tot de tip **CHAR** , ale căror valori au fost alese conform *Tabelului 1.7*.

Port C								
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
D18	1	1	1	0	1	1	0	1
D21	1	1	1	0	1	0	1	0
D24	1	1	1	0	0	1	1	1

Tabelul 4.1 Măştile pentru diametrele fiecărei monede stas

Port D								
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
M3	1	1	1	1	1	1	0	0
M4	1	1	1	1	1	0	1	1
M6	1	1	1	1	1	0	0	1

Tabelul 4.2 Măştile pentru masele fiecărei monede stas

Tabel de adevăr pentru CLC				
Intrări		Ieșiri		Semnificație
SG	SO	Id-moneda (valoare binară)		
D24	M6	1	1	*Valoare Monedă 5 bani*
D21	M4	1	0	*Valoare Monedă 10 bani*
D18	M3	0	1	*Valoare Monedă 50 bani*
Orice alta combinație		0	0	*Eroare*

Tab 4.3 Tabelul de adevăr pentru CLC

Tabel de semnale relevante		
Stare	*Adresa*	*Semnificație*
Q_0	A_0	*Clapeta Valoare Monedă 5 bani*
Q_1	A_1	*Clapeta Valoare Monedă 10 bani*
Q_2	A_2	*Clapeta Valoare Monedă 50 bani*
Q_3	A_3	*Eroare*

Tab 4.4 Tabelul de stări relevante

Capitolul V Proiectarea codului sursă corespunzător sistemului.

Procesul secvențial a fost realizat printr-un SWITCH-CASE cu 6 cazuri(cele 5 corespunzătoare cerinței și cazul default) astfel :

```
switch(stare_moneda)
    {
    case 0 : if(x!= 0 && y != 0)
            {
             stare_moneda += 1;
            }
            break;
    case 1 : diam_moneda=x;
           masa_moneda=y;
           stare_moneda +=1;
           break;
    case 2 :  if(diam_moneda == D18 && masa_moneda == M3)
            {
              id_moneda = 1;
            }
            else if (diam_moneda == D21 && masa_moneda ==
M4)
            {
               id_moneda = 2;
            }
            else if (diam_moneda == D24 && masa_moneda ==
M6)
            {
               id_moneda = 3;
            }
            else {id_moneda = 0;}

           stare_moneda++;

            break;
      case 3 : while(!ready)
            {   index=TAB[Q];
               if(*(index+i) == w)
               {
                  if(*(index+i+1) == 1 && id_moneda == 1)
                  {counter5++;
                  ready = 1;
                  }
                  if(*(index+i+1) == 2 && id_moneda == 2)
                  {counter10++;
                  ready = 1;
                  }
                  if(*(index+i+1) == 3 && id_moneda == 3)
                  {counter50++;
                  ready = 1;
                  }
                  if(*(index+i+1) == 0 && id_moneda == 0)
                  {eroare++;
                  ready = 1;
                  }
                  Q=*(index+i+1);
               }
               else if (*(index+i) == 'T')
               {
                  ready = 1;
               }
            }
           stare_moneda ++;
            break;
case 4 : stare_moneda=0;
           id_moneda=0;
           z=0;
           diam_moneda = 0;
           masa_moneda = 0;
           Q=0;
           index=0;
            x=0;
            y=0;
            ready=0;
           break;

default : stare_moneda = 0;
           id_moneda=0;
           diam_moneda = 0;
           masa_moneda = 0;
           Q=0;
           index=0;
           x=0;
           y=0;
           ready=0;
break;
}
```

Cirucuitul logic secvențial a fost realizat prin declararea tuturor vectorilor de stări, respectiv pointerului de vectori:

```
char A0[]={0x00,1,'T', 0};
char A1[]={0x00,2,'T' ,1};
char A2[]={0x00,3,'T' ,2};
char A3[]={0x00,0,'T' ,3};
char *TAB[4];
```

În procesul secvențial acesta a fost utilizat in cazul 3 :

```
case 3 : while(!ready)
        {   index=TAB[Q];
          if(*(index+i) == w)
          {
            if(*(index+i+1) == 1 && id_moneda == 1)
            {counter5++;
            ready = 1;
            }
            if(*(index+i+1) == 2 && id_moneda == 2)
            {counter10++;
            ready = 1;
            }
            if(*(index+i+1) == 3 && id_moneda == 3)
            {counter50++;
            ready = 1;
            }
            if(*(index+i+1) == 0 && id_moneda == 0)
            {eroare++;
            ready = 1;
            }
            Q=*(index+i+1);
          }
          else if (*(index+i) == 'T')
          {
            ready = 1;
          }
        }
        stare_moneda ++;
         break;
```

Circuitul combinațional a fost realizat printr-o serie de **if**-uri:

```
if(diam_moneda == D18 && masa_moneda == M3)
{
id_moneda = 1;
}
else if (diam_moneda == D21 && masa_moneda == M4)
{
id_moneda = 2;
}
else if (diam_moneda == D24 && masa_moneda == M6)
{
id_moneda = 3;
}
else {id_moneda = 0;}
```

Cap VI.Diagrama secvenţiala corespunzătoare codului sursă.

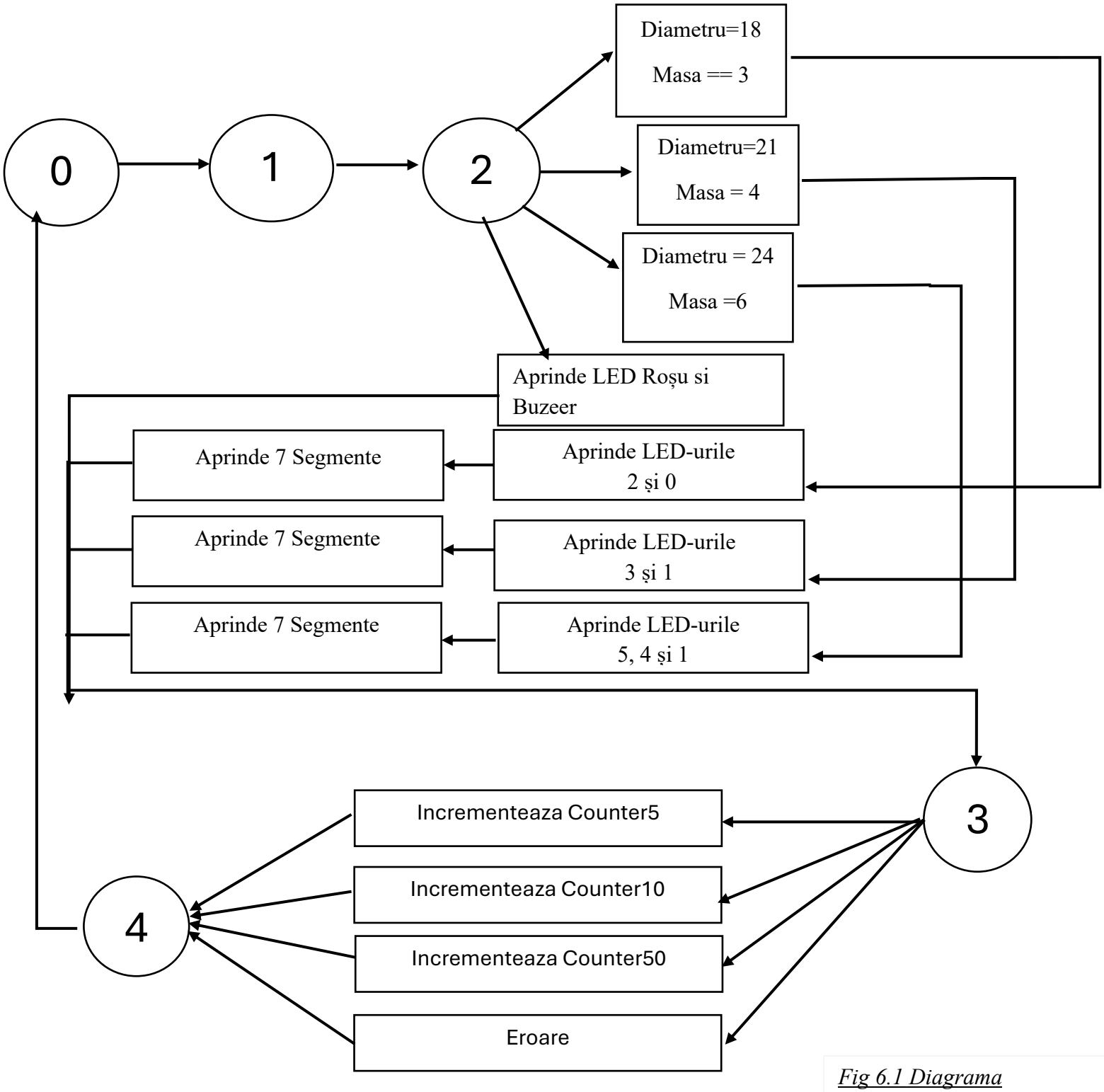


Fig 6.1 Diagrama secvenţiala corespunzătoare codului sursă.sursă

Capitolul VII. Testarea funcţionalităţii sistemului

Testarea funcţionalităţii sistemului am realizat-o cu AVR STUDIO.

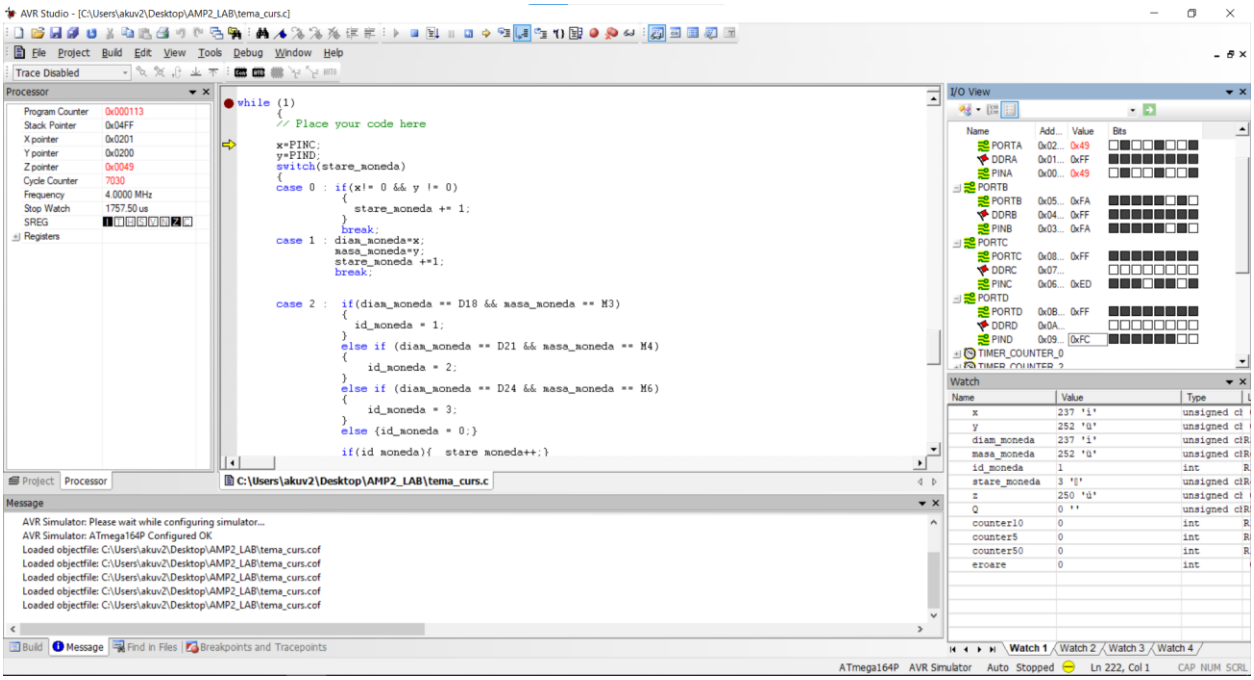


Fig.7.1 Testarea monedei de 5 bani

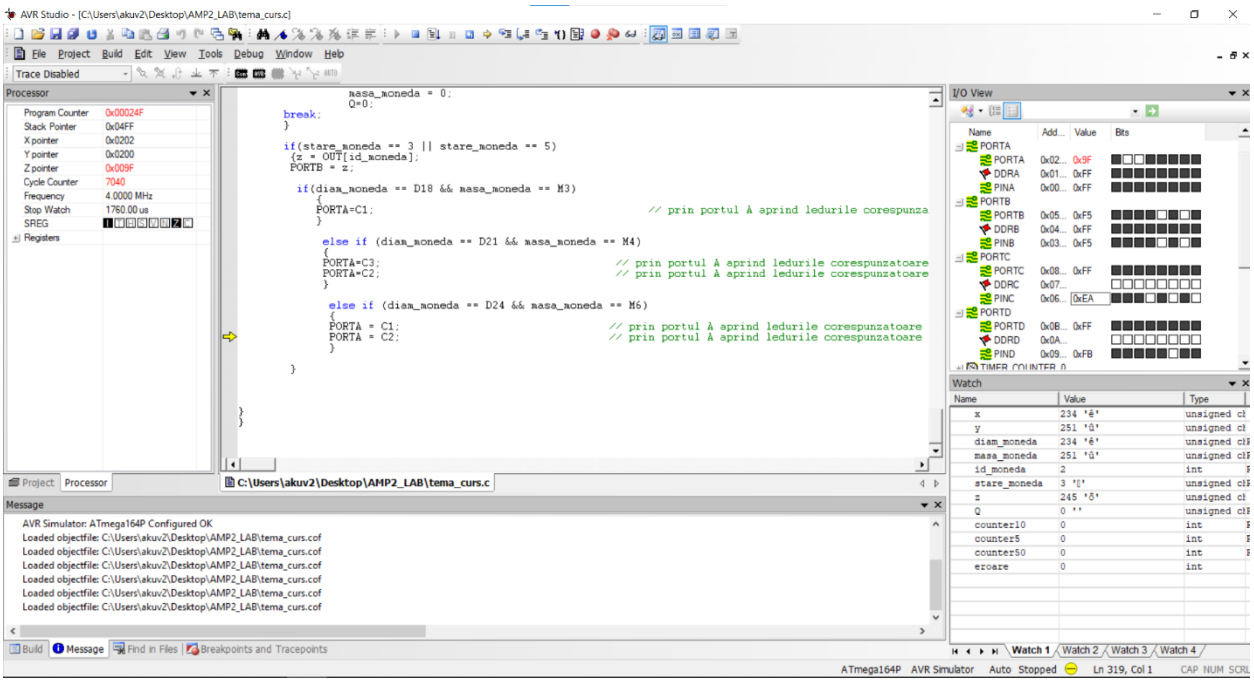


Fig.7.2.a Testarea monedei de 10 bani prin afişarea cifrei 1

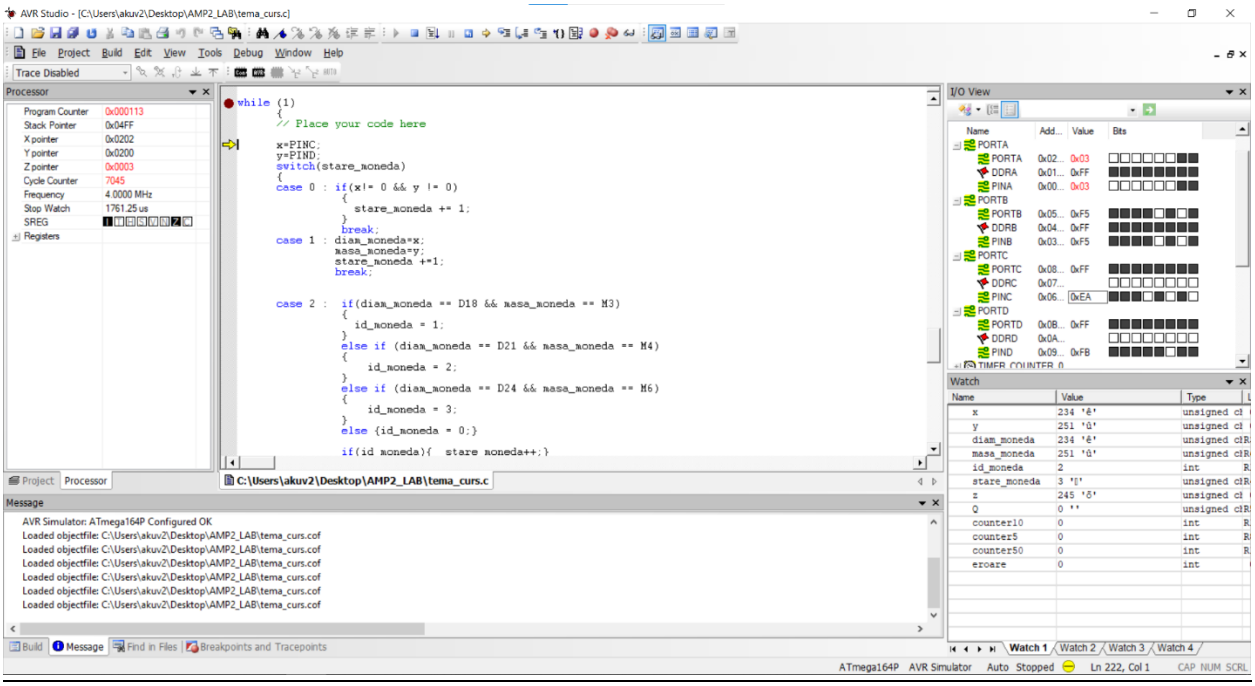


Fig.7.2.b Testarea monedei de 10 bani prin afişarea cifrei 0

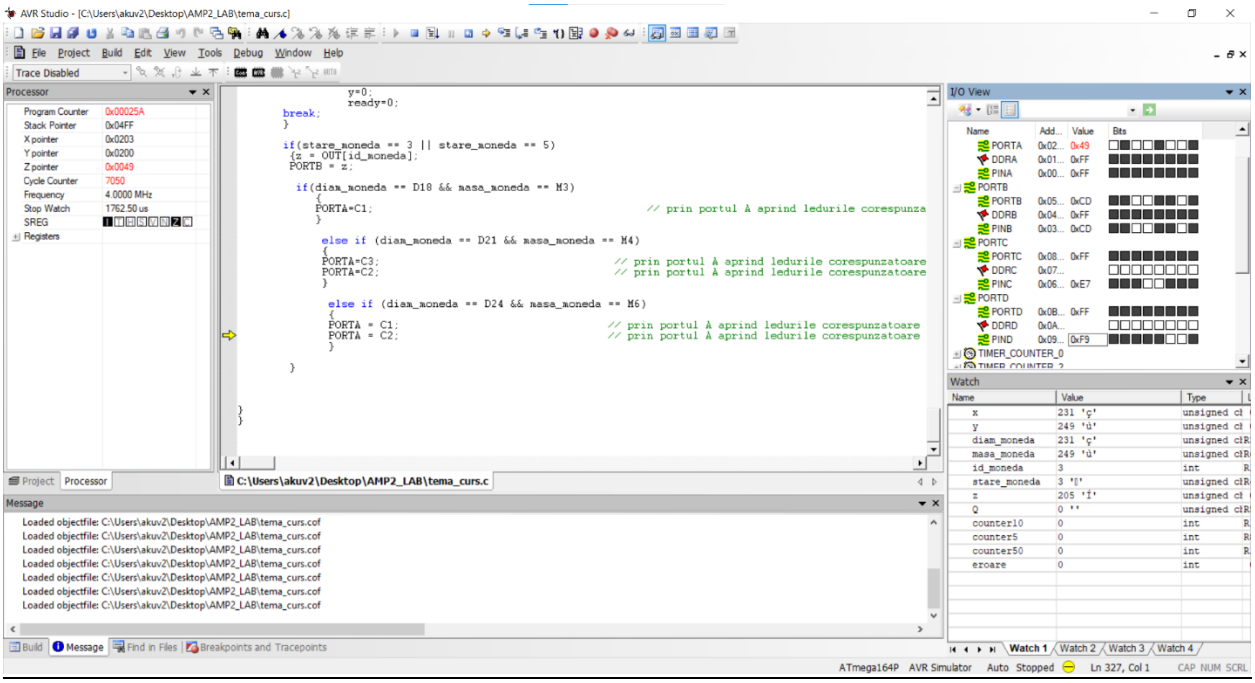


Fig 7.3.a Testarea monedei de 50 bani prin afișarea cifrei 5

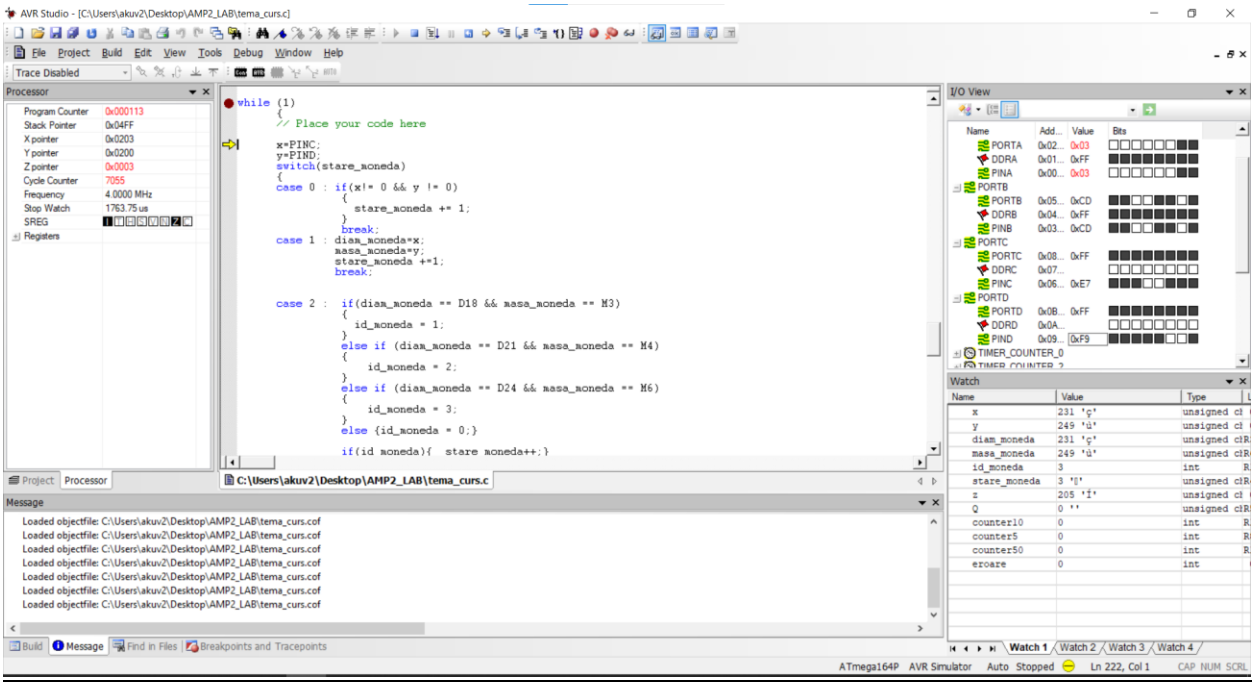


Fig 7.3.b Testarea monedei de 50 bani prin afișarea cifrei 0

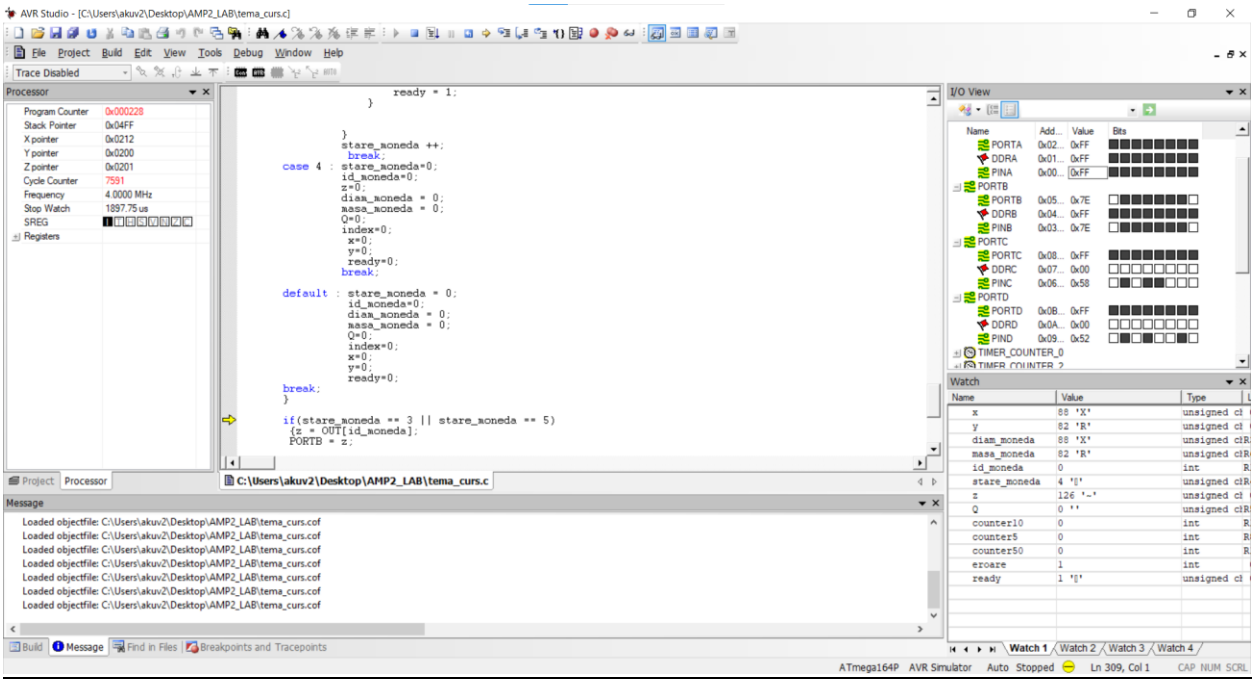


Fig 7.4 Testarea cazului de eroare.

Concluzii.

Prin acest proiect am învățat cat de important este să-mi organizez ideile structural și să respect anumiți pași pentru o realizare eficientă.

Pentru proiectarea unui ”Sistem Automat de Sortare a Monedelor” este necesară respectarea pașilor de realizare, încă de la ideea de implementare până în momentul testării acestuia.Totodată, indiferent de tipul proiectului realizat, software sau hardware, este necesară tranpsunerea ideilor pe hârtie. Această metodă m-a ajutat să schițez realizarea întregului proiect prin transpunerea ideilor principale ale acestuia, fapt ce m-a ajutat sa realizez mai ușor codul sursă.

Totodată, prin acest proiect, am reușit să-mi dezvolt modul de gândire și să înțeleg modul de funcționare al automatelor de sortare a monedelor,dar și ideile de bază ce stau la implementarea oricărui automat.

Astfel, prin intermediul acestui proiect am reușit să utilizez toate cunoștințele dobândite până în prezent.

Bibliografie

https://romaniancoins.org/ro1ban2005.html

Anexă.

Codul sursă al proiectului

```
/*****************************************
*****************

This program was created by the
CodeWizardAVR V4.03

Automatic Program Generator

© Copyright 1998-2024 Pavel Haiduc, HP
InfoTech S.R.L.

http://www.hpinfotech.ro

Project :

Version :

Date    :  4/17/2025

Author  :

Company :

Comments:



Chip type               : ATmega164

Program type            : Application

AVR Core Clock frequency: 10.000000 MHz

Memory model            : Small

External RAM size       : 0

Data Stack size         : 256

*****************************************
****************/


// I/O Registers definitions

#include <mega164.h>


#define D18 0xED  // diametru unei monezi de 5
bani aproximat prin adaos

#define D21 0xEA  // diametru unei monezi de
10 bani aproximat prin adaos.

#define D24 0xE7  // diametru unei monezi de
50 bani aproximat prin adaos.


#define M3 0xFC // masa unei monezi de 5 bani
aproximat prin adaos.

#define M4 0xFB // masa unei monezi de 10
bani aproximat prin adaos.

#define M6 0XF9 // masa unei monezi de 50
bani aproximat prin reducere.


// Declare your global variables here


char stare_moneda = 0;

char diam_moneda = 0;

char masa_moneda = 0;


char OUT[]={0x7E,0xFA,0xF5,0xCD};   // cu
0x81 aprind ledu rosu si buzzeru

int counter5 = 0;

int counter10 = 0;

int counter50 = 0;

int id_moneda = 0;

int eroare = 0;
```

```
char A0[]={0x00,1,'T', 0};
char A1[]={0x00,2,'T' ,1};
char A2[]={0x00,3,'T' ,2};
char A3[]={0x00,0,'T' ,3};
char *TAB[4];
char Q = 0;
char w = 0;
char x = 0;
char y = 0;
char z = 0;
char C1 = 0x49; // cifra 5 afisata pe 7 segmente , unde a este MSB si . este LSB.
char C2 = 0x03; // cifra 0 afisata pe 7 segmente , unde a este MSB si . este LSB.
char C3 = 0x9F;  // cifra 1 afisata pe 7 segmente , unde a este MSB si . este LSB.

char ready = 0;
char *index;
int i=0;
// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
// Reinitialize Timer 0 value
TCNT0=0x3C;
// Place your code here

}

void main(void)
{
// Declare your local variables here
TAB[0]=A0;
TAB[1]=A1;
TAB[2]=A2;
TAB[3]=A3;
// Clock Oscillator division factor: 1
#pragma optsize-
CLKPR=(1<<CLKPCE);
CLKPR=(0<<CLKPCE) | (0<<CLKPS3) | (0<<CLKPS2) | (0<<CLKPS1) | (0<<CLKPS0);
#ifdef OPTIMIZE_SIZE
#pragma optsize+
#endif

// Input/Output Ports initialization
// Port A initialization
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=Out
DDRA=(1<<DDA7) | (1<<DDA6) | (1<<DDA5) | (1<<DDA4) | (1<<DDA3) | (1<<DDA2) | (1<<DDA1) | (1<<DDA0);
// State: Bit7=1 Bit6=1 Bit5=1 Bit4=1 Bit3=1 Bit2=1 Bit1=1 Bit0=1
PORTA=(1<<PORTA7) | (1<<PORTA6) | (1<<PORTA5) | (1<<PORTA4) | (1<<PORTA3) | (1<<PORTA2) | (1<<PORTA1) | (1<<PORTA0);

// Port B initialization
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=Out
DDRB=(1<<DDB7) | (1<<DDB6) | (1<<DDB5) | (1<<DDB4) | (1<<DDB3) | (1<<DDB2) | (1<<DDB1) | (1<<DDB0);
```

```
// State: Bit7=1 Bit6=1 Bit5=1 Bit4=1 Bit3=1
Bit2=1 Bit1=1 Bit0=1

PORTB=(1<<PORTB7) | (1<<PORTB6) |
(1<<PORTB5) | (1<<PORTB4) | (1<<PORTB3)
| (1<<PORTB2) | (1<<PORTB1) |
(1<<PORTB0);


// Port C initialization

// Function: Bit7=In Bit6=In Bit5=In Bit4=In
Bit3=In Bit2=In Bit1=In Bit0=In

DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5)
| (0<<DDC4) | (0<<DDC3) | (0<<DDC2) |
(0<<DDC1) | (0<<DDC0);

// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T
Bit2=T Bit1=T Bit0=T

PORTC=(1<<PORTC7) | (1<<PORTC6) |
(1<<PORTC5) | (1<<PORTC4) | (1<<PORTC3)
| (1<<PORTC2) | (1<<PORTC1) |
(1<<PORTC0);


// Port D initialization

// Function: Bit7=In Bit6=In Bit5=In Bit4=In
Bit3=In Bit2=In Bit1=In Bit0=In

DDRD=(0<<DDD7) | (0<<DDD6) |
(0<<DDD5) | (0<<DDD4) | (0<<DDD3) |
(0<<DDD2) | (0<<DDD1) | (0<<DDD0);

// State: Bit7=P Bit6=P Bit5=P Bit4=P Bit3=P
Bit2=P Bit1=P Bit0=P

PORTD=(1<<PORTD7) | (1<<PORTD6) |
(1<<PORTD5) | (1<<PORTD4) | (1<<PORTD3)
| (1<<PORTD2) | (1<<PORTD1) |
(1<<PORTD0);


// Timer/Counter 0 initialization

// Clock source: System Clock

// Clock value: 9.766 kHz

// Mode: Normal top=0xFF

// OC0A output: Disconnected

// OC0B output: Disconnected

// Timer Period: 20.07 ms

TCCR0A=(0<<COM0A1) | (0<<COM0A0) |
(0<<COM0B1) | (0<<COM0B0) |
(0<<WGM01) | (0<<WGM00);

TCCR0B=(0<<WGM02) | (1<<CS02) |
(0<<CS01) | (1<<CS00);

TCNT0=0x3C;

OCR0A=0x00;

OCR0B=0x00;


// Timer/Counter 1 initialization

// Clock source: System Clock

// Clock value: Timer1 Stopped

// Mode: Normal top=0xFFFF

// OC1A output: Disconnected

// OC1B output: Disconnected

// Noise Canceler: Off

// Input Capture on Falling Edge

// Timer1 Overflow Interrupt: Off

// Input Capture Interrupt: Off

// Compare A Match Interrupt: Off

// Compare B Match Interrupt: Off

TCCR1A=(0<<COM1A1) | (0<<COM1A0) |
(0<<COM1B1) | (0<<COM1B0) |
(0<<WGM11) | (0<<WGM10);

TCCR1B=(0<<ICNC1) | (0<<ICES1) |
(0<<WGM13) | (0<<WGM12) | (0<<CS12) |
(0<<CS11) | (0<<CS10);

TCNT1H=0x00;

TCNT1L=0x00;

ICR1H=0x00;

ICR1L=0x00;

OCR1AH=0x00;
```

```
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2A output: Disconnected
// OC2B output: Disconnected
ASSR=(0<<EXCLK) | (0<<AS2);
TCCR2A=(0<<COM2A1) | (0<<COM2A0) |
(0<<COM2B1) | (0<<COM2B0) |
(0<<WGM21) | (0<<WGM20);
TCCR2B=(0<<WGM22) | (0<<CS22) |
(0<<CS21) | (0<<CS20);
TCNT2=0x00;
OCR2A=0x00;
OCR2B=0x00;

// Timer/Counter 0 Interrupt(s) initialization
TIMSK0=(0<<OCIE0B) | (0<<OCIE0A) |
(1<<TOIE0);

// Timer/Counter 1 Interrupt(s) initialization
TIMSK1=(0<<ICIE1) | (0<<OCIE1B) |
(0<<OCIE1A) | (0<<TOIE1);

// Timer/Counter 2 Interrupt(s) initialization
TIMSK2=(0<<OCIE2B) | (0<<OCIE2A) |
(0<<TOIE2);

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
// Interrupt on any change on pins PCINT0-7:
Off
// Interrupt on any change on pins PCINT8-15:
Off
// Interrupt on any change on pins PCINT16-23:
Off
// Interrupt on any change on pins PCINT24-31:
Off
EICRA=(0<<ISC21) | (0<<ISC20) | (0<<ISC11)
| (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
EIMSK=(0<<INT2) | (0<<INT1) | (0<<INT0);
PCICR=(0<<PCIE3) | (0<<PCIE2) |
(0<<PCIE1) | (0<<PCIE0);

// USART0 initialization
// USART0 disabled
UCSR0B=(0<<RXCIE0) | (0<<TXCIE0) |
(0<<UDRIE0) | (0<<RXEN0) | (0<<TXEN0) |
(0<<UCSZ02) | (0<<RXB80) | (0<<TXB80);

// USART1 initialization
// USART1 disabled
UCSR1B=(0<<RXCIE1) | (0<<TXCIE1) |
(0<<UDRIE1) | (0<<RXEN1) | (0<<TXEN1) |
(0<<UCSZ12) | (0<<RXB81) | (0<<TXB81);

// Analog Comparator initialization
// Analog Comparator: Off
// The Analog Comparator's positive input is
// connected to the AIN0 pin
// The Analog Comparator's negative input is
// connected to the AIN1 pin
```

```
ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) |
(0<<ACI) | (0<<ACIE) | (0<<ACIC) |
(0<<ACIS1) | (0<<ACIS0);

ADCSRB=(0<<ACME);

// Digital input buffer on AIN0: On

// Digital input buffer on AIN1: On

DIDR1=(0<<AIN0D) | (0<<AIN1D);


// ADC initialization

// ADC disabled

ADCSRA=(0<<ADEN) | (0<<ADSC) |
(0<<ADATE) | (0<<ADIF) | (0<<ADIE) |
(0<<ADPS2) | (0<<ADPS1) | (0<<ADPS0);


// SPI initialization

// SPI disabled

SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) |
(0<<MSTR) | (0<<CPOL) | (0<<CPHA) |
(0<<SPR1) | (0<<SPR0);


// TWI initialization

// TWI disabled

TWCR=(0<<TWEA) | (0<<TWSTA) |
(0<<TWSTO) | (0<<TWEN) | (0<<TWIE);


// Globally enable interrupts

#asm("sei")


while (1)

    {

    // Place your code here


    x=PINC;

    y=PIND;

    switch(stare_moneda)

    {

    case 0 : if(x!= 0 && y != 0)

            {

             stare_moneda += 1;

            }

            break;

    case 1 : diam_moneda=x;

            masa_moneda=y;

            stare_moneda +=1;

            break;



    case 2 :  if(diam_moneda == D18 &&
masa_moneda == M3)

            {

             id_moneda = 1;

            }

            else if (diam_moneda == D21 &&
masa_moneda == M4)

            {

               id_moneda = 2;

            }

            else if (diam_moneda == D24 &&
masa_moneda == M6)

            {

               id_moneda = 3;

            }

            else {id_moneda = 0;}


            stare_moneda++;
```

```
            break;
        case 3 : while(!ready)
            {   index=TAB[Q];
               if(*(index+i) == w)
               {
                  if(*(index+i+1) == 1 &&
id_moneda == 1)
                  {counter5++;
                  ready = 1;
                  }
                  if(*(index+i+1) == 2 &&
id_moneda == 2)
                  {counter10++;
                  ready = 1;
                  }
                  if(*(index+i+1) == 3 &&
id_moneda == 3)
                  {counter50++;
                  ready = 1;
                  }
                  if(*(index+i+1) == 0 &&
id_moneda == 0)
                  {eroare++;
                  ready = 1;
                  }
                  Q=*(index+i+1);
               }
               else if (*(index+i) == 'T')
               {
                  ready = 1;
               }
            }
            stare_moneda ++;
             break;
case 4 : stare_moneda=0;
            id_moneda=0;
            z=0;
            diam_moneda = 0;
            masa_moneda = 0;
            Q=0;
            index=0;
             x=0;
             y=0;
             ready=0;
            break;

default : stare_moneda = 0;
             id_moneda=0;
             diam_moneda = 0;
             masa_moneda = 0;
             Q=0;
             index=0;
             x=0;
             y=0;
             ready=0;
break;
}

if(stare_moneda == 3 || stare_moneda == 5)
 {z = OUT[id_moneda];
 PORTB = z;
```

```
        if(diam_moneda == D18 &&
masa_moneda == M3)
          {
          PORTA=C1;                                  //
prin portul A aprind ledurile corespunzatoare                }
cifrei 5.                                                    }
          }


           else if (diam_moneda == D21 &&
masa_moneda == M4)
           {
           PORTA=C3;                                 //
prin portul A aprind ledurile corespunzatoare
cifrei 1.
           PORTA=C2;                                 //
prin portul A aprind ledurile corespunzatoare
cifrei 0 si formez cifra 10.
           }


            else if (diam_moneda == D24 &&
masa_moneda == M6)
            {
            PORTA = C1;                              // prin
portul A aprind ledurile corespunzatoare cifrei 5.
            PORTA = C2;                              // prin
portul A aprind ledurile corespunzatoare cifrei 0
si formez cifra 10.
            }


       }
```