

Universitatea Politehnica Bucuresti
Facultatea de Electronica,Telecomunicatii si Tehnologia Informatiei

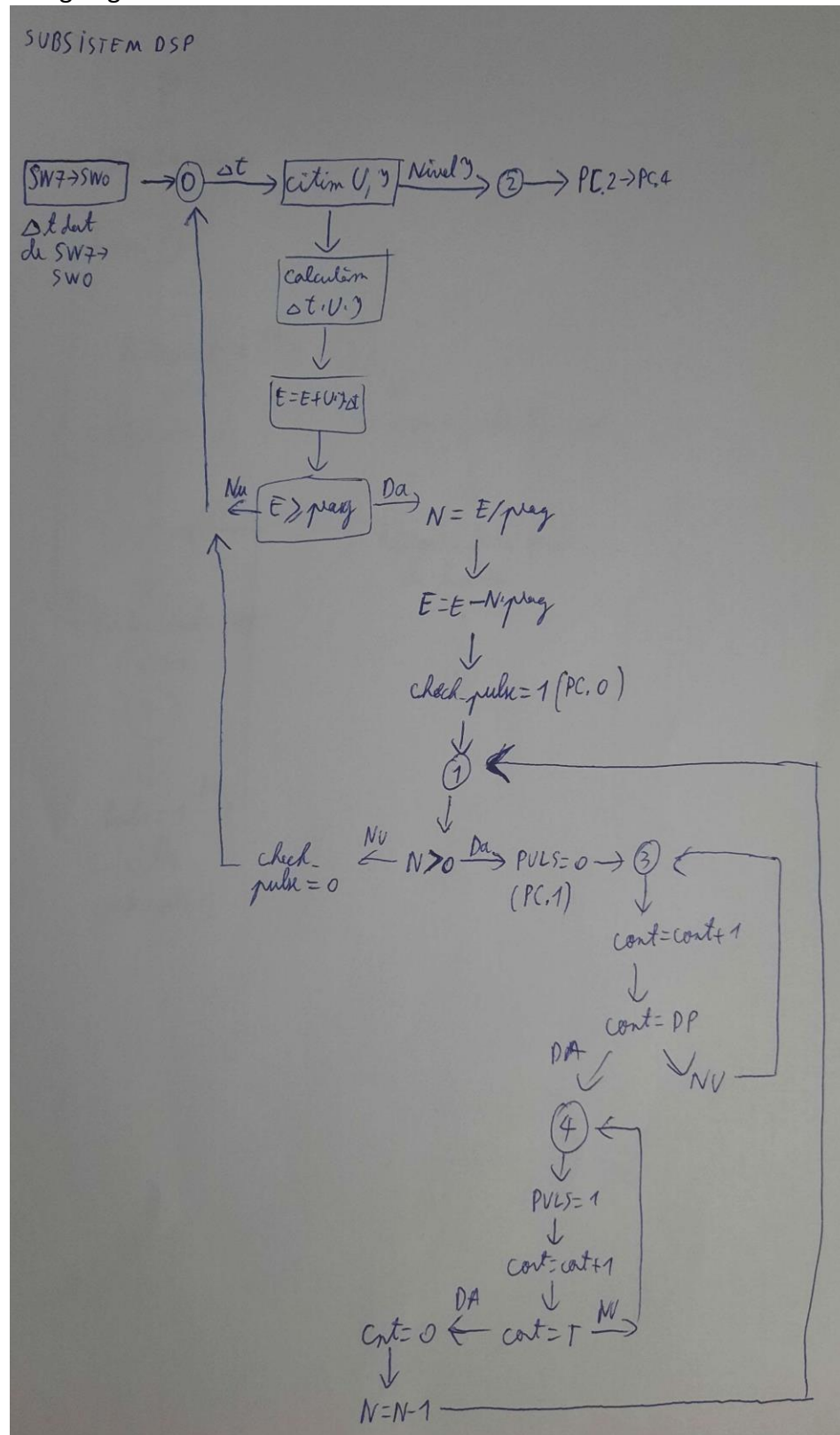
PROIECT 2

Profesor coordonator:Zoican Sorin

Realizat de:
Burlacel George
Busicescu Mihai
Nica Bogdan Claudiu
Tudoran Daniel

Grupa: 433Db

1.Organigrama:



SUBSISTEM AVR

Măsurarea timpului:

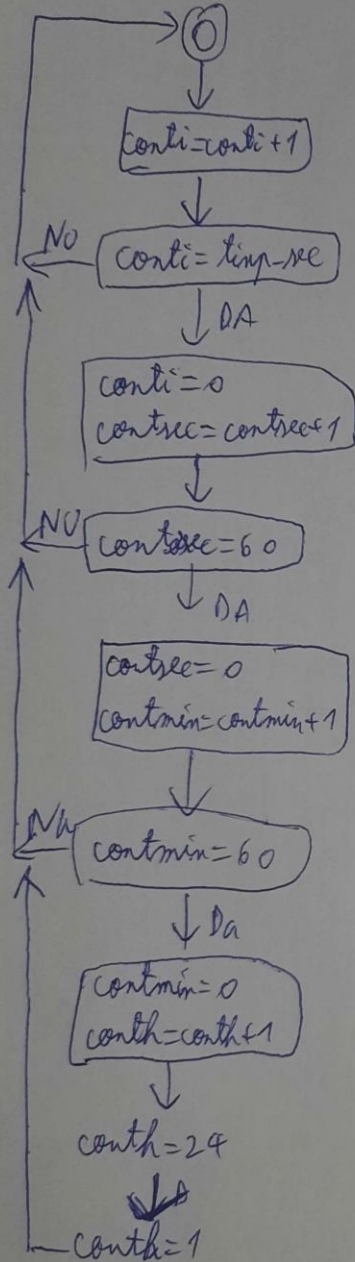
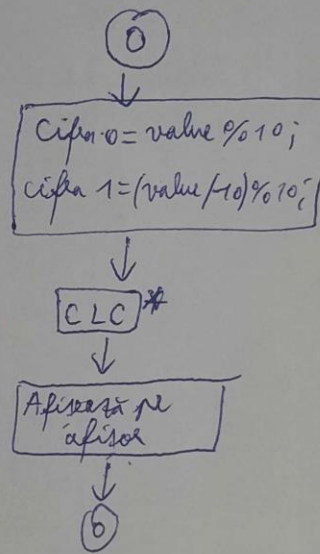


Tabela adresă LED-uri stare curent:

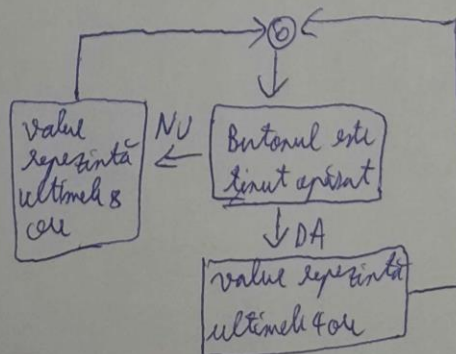
PC.2	PC.3	PC.4	LED 0	LED 1	LED 2
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0

Afișor 2 cifre:

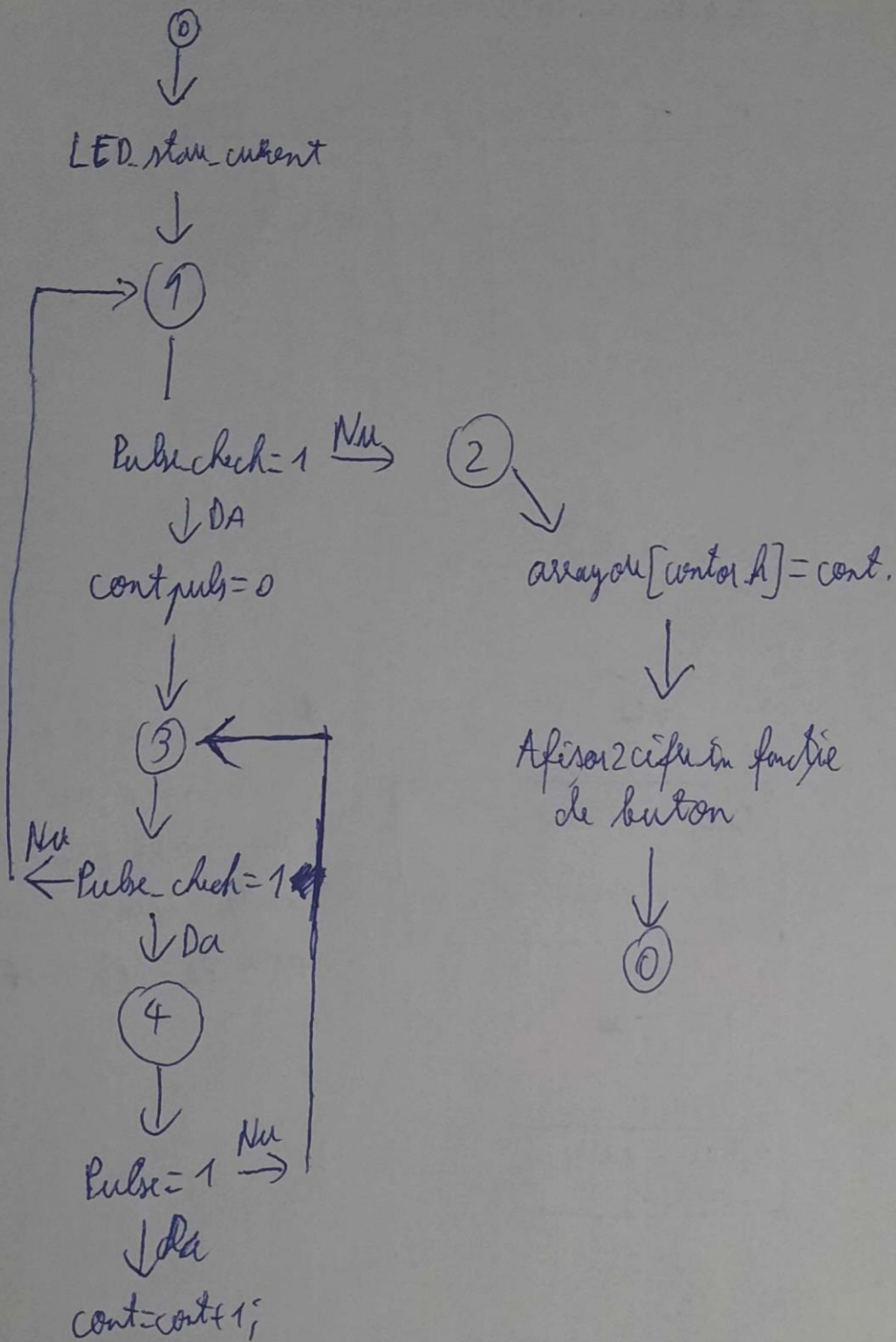


* Pentru cifra 0 s-au alocat pinii PA. 0 → PA. 6 iar pentru cifra 1 pinii PA. 7; PD. 7; PB. 0 → PB. 4

Butonul:

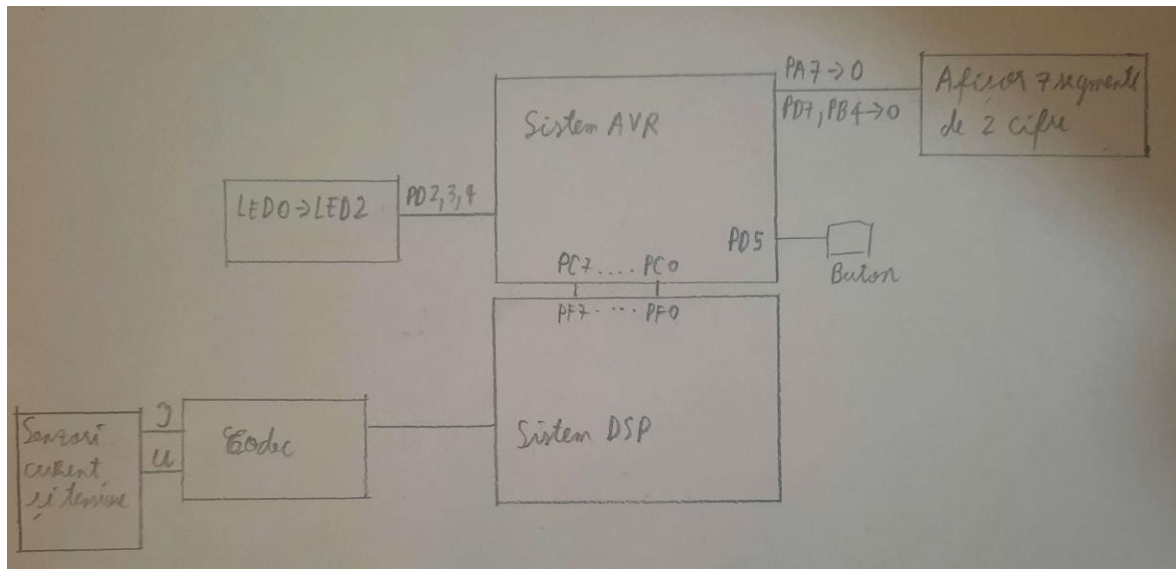


Controlarea și interpretare pulsuri:

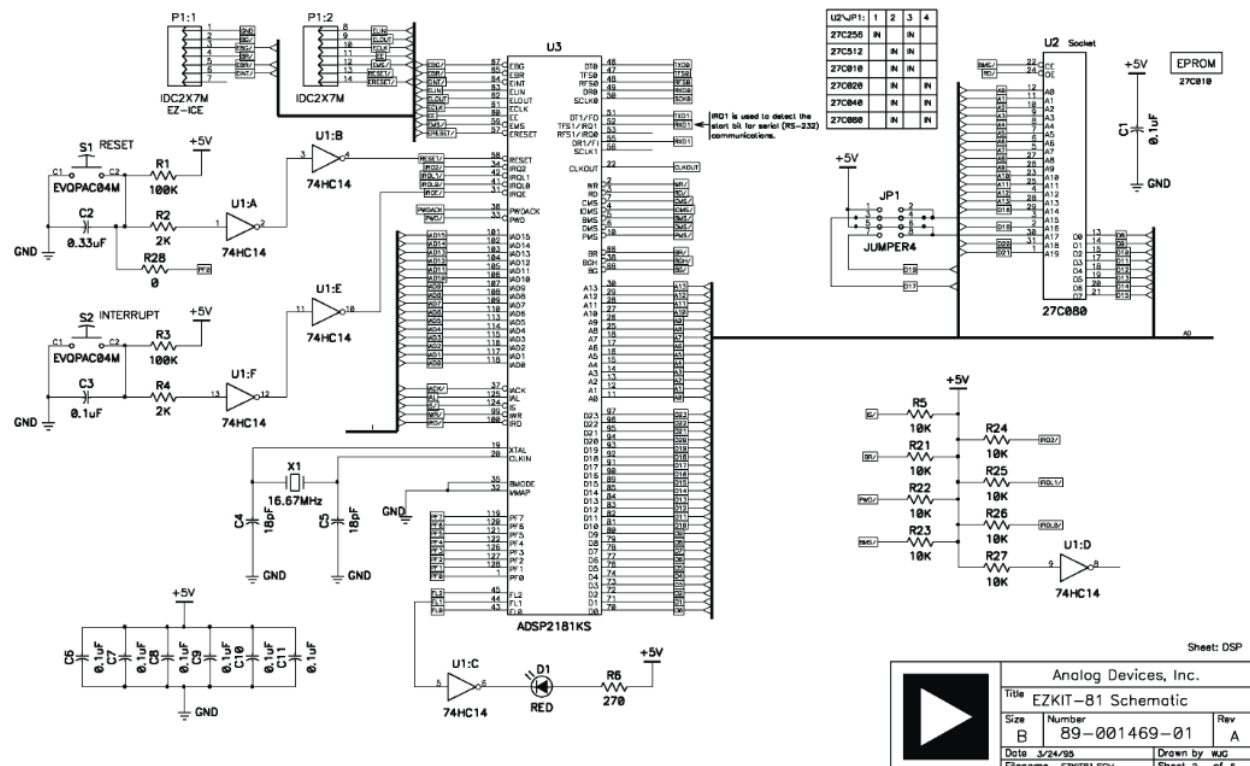


2.Hardware

Schema bloc



Sistemul DSP



SW1 este folosit pentru a reseta microcontrolerul acesta muta punctul de masa in fata rezistentei R2 astfel trimite un semnal de "0" logic.

Cristalul de cuarț X1 și condensatoarele C4, C5 formează un oscilator de cuarț.

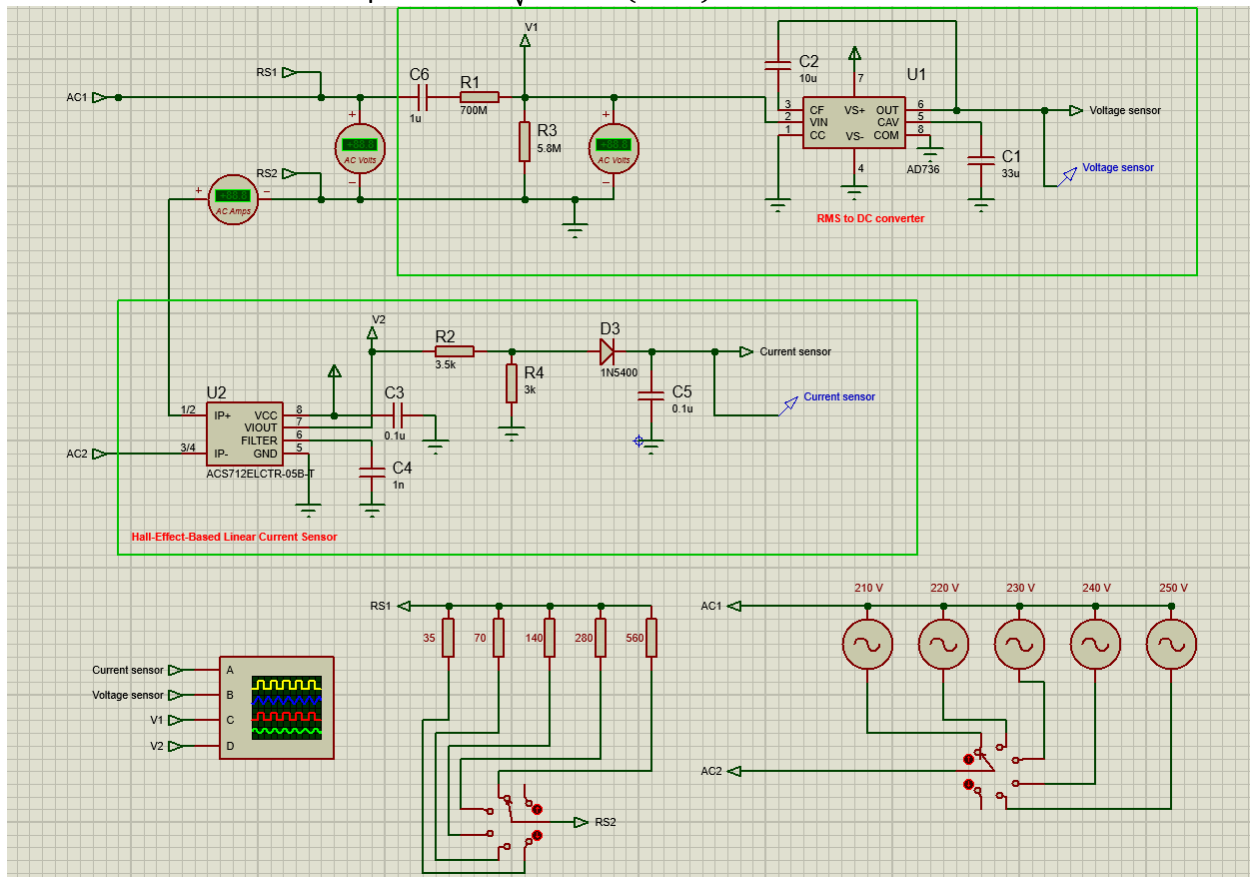
6

Senzori curent si tensiune

Senzorul de Curent conform documentatiei si schemei electrice ar avea in total la iesire 4.48 V pentru 1 A, dioda 1N5400 ne mentine un nivel de curent de maxim 3A cu o influenta mica asupra tensiunii la iesire(maxim 1 V conforma documentatiei).

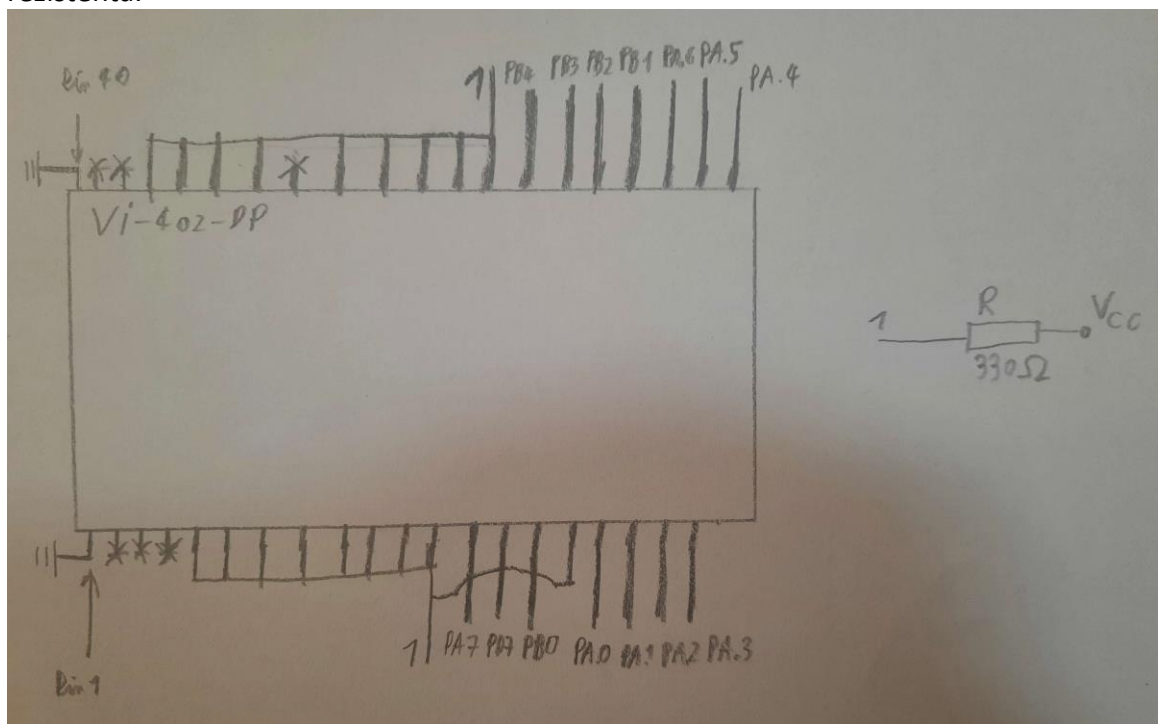
Senzorul de tensiune conform documentatiei acesta face conversia de la un semnal de amplitudine

“Vin” la o tensiune DC de tip $V_{out} = \sqrt{\text{media}(V_{in}^2)}$

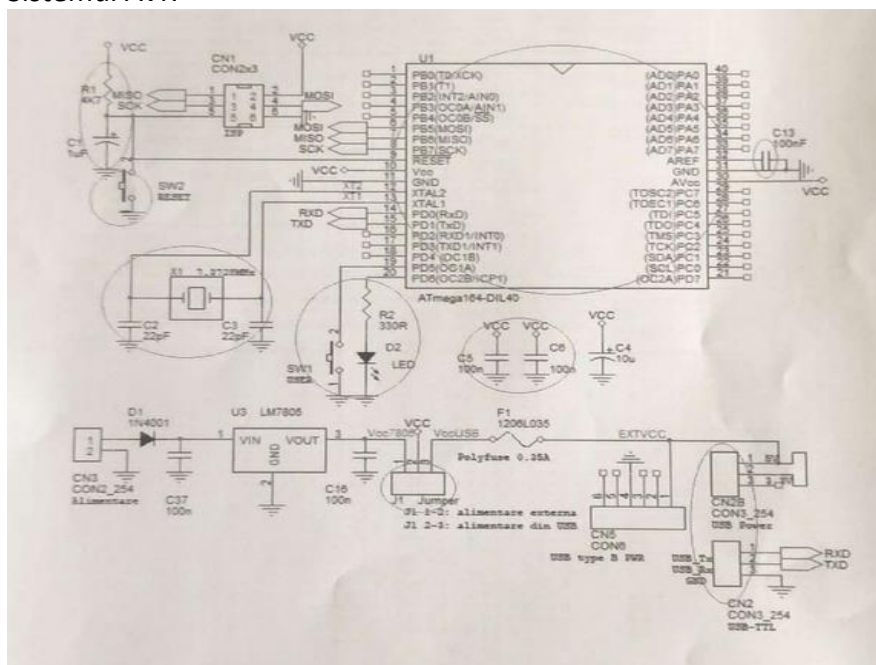


Afisorul pe 7 segmente cu 2 cifre

Cum acesta functioneaza pe logica inversa pini pentru celelalte cifre si simboluri trebuie mentinuti pe "1" logic pentru a se aprinde astfel conectam pinii respectivi la Vcc printr-o rezistenta.



Sistemul AVR



- R1,C1 formeaza un circuit de power-on/reset. Prin aplicarea tensiunii de alimentare Vcc se va aplica "0" logic resetand microcontrolerul pana cand condensatorul C1 se incarca, transmitandu-se apoi "1" logic. SW2 functioneaza ca un buton de reset.
- LEDul D2 se aprinde pe "1" logic si acesta poate functiona ca un indicator power-on/power-off.
- SW1 este folosit pentru subsistemul AVR , acesta daca este tinut apasat ne va afisa ultimele 4 ore de consum iar daca nu e apasat ne va afisa ultimele 8.
- Cristalul de cuarț X1 alaturi de condensatoarele C2,C3 formeaza un oscilator de cuarț.
- Prin CN3 se poate realiza alimentare externa, dioda D1 este folosita pentru a proteja la alimentare inversa.
- LM7805 este un stabilizator de tensiune pentru a ne aduce tensiunea de alimentare la 5V
- Jumperul J1 poate fi pus de 2 pozitii: 1-2 pentru alimentare externa prin CN3 iar 2-3 pentru alimentare externa prin USB.
- F1 este o rezistenta fuzibila folosita pentru a nu cauza daune sistemului la o supra alimentare de curent.
- Pini PC7->PC0 sunt conectati la porti PF7->0 de la DSP, pini PD2,3 si 4 sunt conectati la LED0->2, Pini PB0->4, PD7 si PA 7->0 sunt conectati la afisorul de 4 cifre.

3.Subsistemul AVR

Sub sistemul AVR proceseaza informatiile primite de la subsistemul DSP pentru: a afisa pe LEDurile LED0 → LED2 nivelul de curent consumat avand 4 stari: niciun LED aprins semnifica faptul ca nu s-a primit nimic de la DSP, LED0 aprins ne semnifica un nivel scazut de curent, LED1 un nivel mediu de consum si LED3 un nivel mare; a ne afisa pe un afisor cu sapte segmente ce ne arata doua cifre valoarea in kWh consumata in ultimele 8 ore daca nu e apasat butonul sau valoarea consumata in ultimele 4 ore daca este tinut apasat butonul.

Subsistemul AVR primeste pe PC0 un semnal de Check Pulse, care in 1 logic semnifica prezenta semnalului de intrare iar pe 0 logic absenta acestuia. Cat timp Check Pulse este in 1 logic, pe PC1 primim un numar de pulsuri de la sistemul DSP pentru a semnifica consumul de kWh unde un puls semnifica un kWh.

Contorizarea timpului se realizeaza prin timer 0 un timer de 8 biti de frecventa de 19.531 kHz, modificat prin CodeVisionAVR care are valoarea implicita de 60, 0x3C, pentru a ne oferi o perioada de 10ms la o frecventa a microcontrolerului de 20MHz, de fiecare data cand timerul ajunge la valoarea de 0xFF, 255 creaza o intrerupere si reincepte contorizarea de la valoarea de 0x3C.

Valoarea primita este salvata intr-un array de 24 de elemente corespunzator celor 24 de ore dintr-o zi.

4.Subsistemul DSP

Lesirea de la Senzori este interpretata de CODEC cu timpi de esantionare determinati de SW7-0 din extensia DSP, afisorul din extensie ne afiseaza numarul corespunzator switchului indicator al

duratei de esantionare. Sistemul DSP calculeaza valoarea de kWh pana atinge pragul minim de 1kWh care apoi genereaza un puls pentru fiecare 1 kWh calculat. Trimite pe PF0 valoarea "1" logic pentru a semnaliza catre AVR ca va primi pulsurile pentru afisarea consumului. Pe porti PF2,3,4 vom trimite valoarea curentului interpretata de catre DSP: 0 A trimite codul "000", 5mA-100mA trimite codul "001" pentru a semnifica un nivel min de consum de curent, 100mA-2A trimite cod "010" pentru a semnifica un nivel mediu de consum, iar 2A-5A trimite cod "100" pentru a semnifica un nivel mare de consum.

Simulari AVR

Demonstare afisor nivel curent,

Nimic primit pe PinC0->7 PortD 4,3,2 sa fie 0

The image displays two screenshots of the AVR Studio IDE, showing the source code and the I/O View window.

Top Screenshot:

- Source Code:** The code is in C and defines a main loop. It initializes SPI, TWI, and TWCR. It then enters a while loop where it checks if PINC.0 is 1. If so, it increments a counter and updates the arrayOre[contorHr] and Afisor_2Cifre[ValAfisorButon()].
- I/O View:** The I/O View window shows the state of various I/O components. The PORTC register is highlighted, showing its value as 0x00. The PORTD register is also highlighted, showing its value as 0x00.

Bottom Screenshot:

- Source Code:** The code is identical to the top screenshot, but the while loop condition is changed to `if(PINC.0==1)`.
- I/O View:** The I/O View window shows the state of various I/O components. The PORTC register is highlighted, showing its value as 0x00. The PORTD register is also highlighted, showing its value as 0x00.

PinC2 devine "1" logic nivelul o sa fie PortD 4 =1 PortD.3,2 sa fie 0

The image displays two screenshots of an AVR microcontroller project, likely using the AVR Studio IDE. The left screenshot shows the main.c file with the following code:

```

// ADC initialization
// ADC disabled
ADCSRA=(0<<ADEN) | (0<<ADSC) | (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<<TWEN) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);

// Globally enable interrupts
asm("sei");

while (1)
{
    LED_State_Current();
    if(PINC.0==1)
    {
        cont_pulse=0;
        while(PINC.0!=0)
        {
            if(PINC.1==1)
            {
                cont_pulse=cont_pulse+1;
            }
        }
        arrayOre[contorHr]=cont_pulse;
        Afisor_2Cifre(ValAfisorButon());
    }
}
    
```

The right screenshot shows the I/O View window, which displays the current state of the microcontroller's I/O registers. The registers shown are:

- AD_CONVERTER: Port A Data Register (0x40), Port A Data Direction Register (0xFF), Port A Input Pins (0x40)
- CPU: BOOT_LOAD (0x00), CPU (0x00), EEPROM (0x00), EXTERNAL_INTERRUPT (0x00), JTAG (0x00)
- PORTA: Port A Data Register (0x40), Port A Data Direction Register (0xFF), Port A Input Pins (0x40)
- PORTB: Port B Data Register (0x10), Port B Data Direction Register (0xFF), Port B Input Pins (0x10)
- PORTC: Port C Data Register (0xFF), Port C Data Direction Register (0x00), Port C Input Pins (0x04)
- PORTD: Port D Data Register (0x10), Port D Data Direction Register (0xFF), Port D Input Pins (0x10)
- SPI: SPI (0x00)

The I/O View window also shows a table of registers with their addresses and values:

Name	Address	Value	Bits
DDRC	0x07 (0x27)	0x00	00000000
PINC	0x06 (0x26)	0x04	00000000
PORTC	0x08 (0x28)	0xFF	11111111

Demonstrare afisor 7 segmente;

Se primeste PinC0, PinC2 "1" logic, se trimite impulsuri pe Pinc1, PortA0->6 iau valorile corespunzatoare functiei

The image displays two screenshots of an AVR microcontroller project in Visual Studio, showing the code and the I/O View.

Top Screenshot:

- Code:** The code is in C and shows initialization for ADC, SPI, TWI, and TWCR. It includes a while loop that checks for logic high on Pinc0 and Pinc2, and sends a pulse to Pinc1. The code also calls a function to display the value on a 7-segment display.
- I/O View:** The I/O View shows the state of the microcontroller's I/O components. The PORTC input pins are shown with values 0x00 and 0x00. The PORTC output pins are shown with values 0x00 and 0x00.

Bottom Screenshot:

- Code:** The code is identical to the top screenshot.
- I/O View:** The I/O View shows the state of the microcontroller's I/O components. The PORTA input pins are shown with values 0x24, 0x24, 0x24, 0x24, 0x24, 0x24, and 0x24. The PORTA output pins are shown with values 0x24, 0x24, 0x24, 0x24, 0x24, 0x24, and 0x24.

5.Cod AVR

```
#include <io.h>

char arrayOre[24]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
char contorIn=0;
//initializare a variabilelor / contorului de timp
char contorSec=0;
char contorMin=0;
char contorHr=1;
char val_afisor;
char cont_pulse=0;
// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
    // Reinitialize Timer 0 value
    TCNT0=0x3C; //acesta este valoarea timerului pentru a obtine o perioada de 10ms.

    contorIn=contorIn+1;
    if(contorIn%100==0){
        contorSec=contorSec+1;
        contorIn=0;
    }
    if(contorSec%60==0){
        contorMin=contorMin+1;
        contorSec=0;
    }
    if(contorMin%60==0){
        contorHr=contorHr+1;
        contorMin=0;
    }
}
```

```

if(contorHr%24==0){
    contorHr=1;
}
// functie de contorizare a timpului
}

void LED_Stare_Curent(void)
{
    if(PINC.2==1){
        PORTD.4=1; // reprezinta o valoare scazuta 10mA-1.5ish A
        PORTD.3=0;
        PORTD.2=0;
    }else if(PINC.3==1){
        PORTD.4=0;
        PORTD.3=1; // reprezinta o valoare medie 1.501 A -3.5 A
        PORTD.2=0;
    }else if(PINC.4==1){
        PORTD.4=0;
        PORTD.3=0;
        PORTD.2=1; // reprezinta o valoare mare 3.501 A- 5A
    }
    else {
        PORTD.4=0;
        PORTD.3=0;
        PORTD.2=0; // nu ne afiseaza niciun led
    }
    //aceasta functie preia valoarea trimisa de dsp( valorile sunt intre anumite nivele
    // De curent primite de la DSP)
}

void Afisor_2Cifre(char value){
    //luam PORTA complet, PortD7,portB 0->4 14 biti pentur 2 cifre 7 biti din port A o sai setezi
    pentru prima cifra restul cifra 2

```



```
//tine minte afisorul este pe logica inversa
```

```
cifra0=value%10;
```

```
// prima cifra
```

```
cifra1=(value/10)%10;
```

```
//a doua cifra
```

```
if(cifra0==0)
```

```
{
```

```
PORTA.0=0; //pin 17 4E
```

```
PORTA.1=0; //pin 18 4D
```

```
PORTA.2=0; //pin 19 4C
```

```
PORTA.3=0; //pin 20 4B
```

```
PORTA.4=0; //pin 20 4A
```

```
PORTA.5=0; //pin 20 4F
```

```
PORTA.6=1; //pin 20 4G
```

```
}
```

```
else if(cifra0==1)
```

```
{
```

```
PORTA.0=1; //pin 17 4E
```

```
PORTA.1=1; //pin 18 4D
```

```
PORTA.2=0; //pin 19 4C
```

```
PORTA.3=0; //pin 20 4B
```

```
PORTA.4=1; //pin 20 4A
```

```
PORTA.5=1; //pin 20 4F
```

```
PORTA.6=1; //pin 20 4G
```

```
}
```

```
else if(cifra0==2)
```

```
{PORTA.0=0; //pin 17 4E
```

```
PORTA.1=0; //pin 18 4D
```

```
PORTA.2=1; //pin 19 4C
```

```
PORTA.3=0; //pin 20 4B
```

```
PORTA.4=0; //pin 20 4A
```

```

PORTA.5=1; //pin 20 4F
PORTA.6=0; //pin 20 4G
}
else if(cifra0==3)
{PORTA.0=1; //pin 17 4E
PORTA.1=0; //pin 18 4D
PORTA.2=0; //pin 19 4C
PORTA.3=0; //pin 20 4B
PORTA.4=0; //pin 20 4A
PORTA.5=1; //pin 20 4F
PORTA.6=0; //pin 20 4G
}
else if(cifra0==4)
{PORTA.0=1; //pin 17 4E
PORTA.1=1; //pin 18 4D
PORTA.2=0; //pin 19 4C
PORTA.3=0; //pin 20 4B
PORTA.4=1; //pin 20 4A
PORTA.5=0; //pin 20 4F
PORTA.6=0; //pin 20 4G
}
else if(cifra0==5)
{PORTA.0=1; //pin 17 4E
PORTA.1=0; //pin 18 4D
PORTA.2=0; //pin 19 4C
PORTA.3=1; //pin 20 4B
PORTA.4=0; //pin 20 4A
PORTA.5=0; //pin 20 4F
PORTA.6=0; //pin 20 4G
}

```

```

else if(cifra0==6)
{PORTA.0=0; //pin 17 4E
PORTA.1=0; //pin 18 4D
PORTA.2=0; //pin 19 4C
PORTA.3=1; //pin 20 4B
PORTA.4=0; //pin 20 4A
PORTA.5=0; //pin 20 4F
PORTA.6=0; //pin 20 4G
}
else if(cifra0==7)
{PORTA.0=1; //pin 17 4E
PORTA.1=1; //pin 18 4D
PORTA.2=0; //pin 19 4C
PORTA.3=0; //pin 20 4B
PORTA.4=0; //pin 20 4A
PORTA.5=1; //pin 20 4F
PORTA.6=1; //pin 20 4G
}
else if(cifra0==8)
{PORTA.0=0; //pin 17 4E
PORTA.1=0; //pin 18 4D
PORTA.2=0; //pin 19 4C
PORTA.3=0; //pin 20 4B
PORTA.4=0; //pin 20 4A
PORTA.5=0; //pin 20 4F
PORTA.6=0; //pin 20 4G
}
else if(cifra0==9)
{PORTA.0=1; //pin 17 4E
PORTA.1=0; //pin 18 4D

```

```

PORTA.2=0; //pin 19 4C
PORTA.3=0; //pin 20 4B
PORTA.4=0; //pin 20 4A
PORTA.5=0; //pin 20 4F
PORTA.6=0; //pin 20 4G
}
if(cifra1==0)
{
PORTA.7=0; // pin 13 3E
PORTD.7=0; // pin 14 3D
PORTB.0=0; // pin 15 3C
PORTB.1=0; // pin 24 3B
PORTB.2=0; // pin 25 3A
PORTB.3=0; // pin 26 3F
PORTB.4=1; // pin 27 3G
}
else if(cifra1==1)
{
PORTA.7=1; // pin 13 3E
PORTD.7=1; // pin 14 3D
PORTB.0=0; // pin 15 3C
PORTB.1=0; // pin 24 3B
PORTB.2=1; // pin 25 3A
PORTB.3=1; // pin 26 3F
PORTB.4=1; // pin 27 3G
}
else if(cifra1==2)
{
PORTA.7=0; // pin 13 3E
PORTD.7=0; // pin 14 3D

```

```

PORTB.0=1; // pin 15 3C
PORTB.1=0; // pin 24 3B
PORTB.2=0; // pin 25 3A
PORTB.3=1; // pin 26 3F
PORTB.4=0; // pin 27 3G
}
else if(cifra1==3)
{
PORTA.7=1; // pin 13 3E
PORTD.7=0; // pin 14 3D
PORTB.0=0; // pin 15 3C
PORTB.1=0; // pin 24 3B
PORTB.2=0; // pin 25 3A
PORTB.3=1; // pin 26 3F
PORTB.4=0; // pin 27 3G
}
else if(cifra1==4)
{
PORTA.7=1; // pin 13 3E
PORTD.7=1; // pin 14 3D
PORTB.0=0; // pin 15 3C
PORTB.1=0; // pin 24 3B
PORTB.2=1; // pin 25 3A
PORTB.3=0; // pin 26 3F
PORTB.4=0; // pin 27 3G
}
else if(cifra1==5)
{
PORTA.7=1; // pin 13 3E
PORTD.7=0; // pin 14 3D

```

```

PORTB.0=0; // pin 15 3C
PORTB.1=1; // pin 24 3B
PORTB.2=0; // pin 25 3A
PORTB.3=0; // pin 26 3F
PORTB.4=0; // pin 27 3G
}
else if(cifra1==6)
{
PORTA.7=0; // pin 13 3E
PORTD.7=0; // pin 14 3D
PORTB.0=0; // pin 15 3C
PORTB.1=1; // pin 24 3B
PORTB.2=0; // pin 25 3A
PORTB.3=0; // pin 26 3F
PORTB.4=0; // pin 27 3G
}
else if(cifra1==7)
{
PORTA.7=1; // pin 13 3E
PORTD.7=1; // pin 14 3D
PORTB.0=0; // pin 15 3C
PORTB.1=0; // pin 24 3B
PORTB.2=0; // pin 25 3A
PORTB.3=1; // pin 26 3F
PORTB.4=1; // pin 27 3G
}
else if(cifra1==8)
{
PORTA.7=1; // pin 13 3E
PORTD.7=1; // pin 14 3D

```



```

PORTB.0=1; // pin 15 3C
PORTB.1=1; // pin 24 3B
PORTB.2=1; // pin 25 3A
PORTB.3=1; // pin 26 3F
PORTB.4=1; // pin 27 3G
}
else if(cifra1==9)
{
PORTA.7=1; // pin 13 3E
PORTD.7=0; // pin 14 3D
PORTB.0=0; // pin 15 3C
PORTB.1=0; // pin 24 3B
PORTB.2=0; // pin 25 3A
PORTB.3=0; // pin 26 3F
PORTB.4=0; // pin 27 3G
}
}

char ValAfisorButon(void){
char SUMA;
char buffer_calc;
char i;
if(PIND.5==0)//buton apasat butonul este portD5, cu portD6 allways on pentru LED
{
SUMA=0;
if(contorHr>8)
{ buffer_calc=contorHr-8;
for(i=buffer_calc;i<=contorHr;i++)
{ SUMA=SUMA+arrayOre[i];
}
}
// ne vefica daca ora e mai mare sau nu de 8 ore si retureneaza valoarea de pe ultimele 8 ore

```

```

}
else{
buffer_calc=8-contorHr;
for(i=24-buffer_calc;i<=24;i++)
{ SUMA=SUMA+arrayOre[i];
}
for(i=1;i<=contorHr;i++){
SUMA=SUMA+arrayOre[i];}
}
return val_afisor=SUMA;
//sfarsitul functiei de 8 de ore
}
else
{SUMA=0;

if(contorHr>4)
{
buffer_calc=contorHr-4;
for(i=buffer_calc;i<=contorHr;i++)
{
SUMA=SUMA+arrayOre[i];
}
}
else {
buffer_calc=4-contorHr;
for(i=24-buffer_calc;i<=24;i++)
{
SUMA=SUMA+arrayOre[i];
}
for(i=1;i<=contorHr;i++)
{

```

```

SUMA=SUMA+arrayOre[i];
}
// ar trebui sa ne arate consumul de kWh pentru ultimele 4 ore cat timp apasam
}
return val_afisor=SUMA;
}
}

```

```

PIND.6=1;// port mereu pe "1" logic pentru LEDul conectat la acesta
while (1)
{
LED_Stare_Curent();
//Citeste starea curentului
if(PINC.0==1 )
//PC.0 este pentru puls_check
{
cont_pulse=0;
//initializam contorul de pulsuri la 0 mereu cand primim un pulse check nou
while(PINC.0!=0)
// cat timp primim pulsecheck se contorizeaza impulsurile primite
{if(PINC.1==1)
//PC.1 este pinul
//pe care primim pulsurile
{
cont_pulse=cont_pulse+1;
}
}
}
arrayOre[contorHr]=cont_pulse; //introducem in array cate pulsuri s-au numarat
Afisor_2Cifre(ValAfisorButon());
//afisarea consumului in ultimele 8/4 ore in functie de buton
}
}

```

6.Cod DSP

```
.section/dm data1;
// intrari
.var E;
.var U;
.var I;
.var P;
.var A;
.var timp;
.var prag;
.var N;

.var output[1000];

.section/pm IVreset;

jump start;nop;nop;nop;
rti;nop;nop;nop;
rti;nop;nop;nop;
rti;nop;nop;nop;
rti;nop;nop;nop;
rti;nop;nop;nop;
rti;nop;nop;nop;

.section/pm program;

calcul:
MR = 0 //registrul de rezultat = 0
MX0=DM(U); //in registrul x este incarcata valoarea tensiunii
MY0=DM(I); //in registrul y este incarcata valoarea curentului
MR=MX0*MY0; //in registrul de rezultat este calculata puterea
DM(P)=MR; //valoarea puterii este incarcata in memorie
```

```

MX0=DM(timp); //reg x ia valoarea timpului
MY0=DM(P); //reg y ia valoarea puterii
MR=MR+MX0*MY0; //calculul energiei (suma de produse)

DM(E)=MR; //valoarea energiei este incarcata in memorie
MX0=DM(E); //in reg x se incarca valoarea energiei
MY0=DM(prag); //in y este incarcata valoarea pragului
MR=DIVS MX0, MY0; //in reg de rezultat o sa apara numarul N de impulsuri
//care trebuie generate dupa impartirea energiei la pragul de 1KWh
//(3 KWh/1 KWh => 3 impulsuri)
DM(N)=MR; //numarul de impulsuri salvat in memorie
// daca N > 0 atunci se genereaza un puls

stop: jump stop;

```