Universitatea Politehnica Bucuresti
Facultatea de Electronica,Telecomunicatii si Tehnologia Informatiei

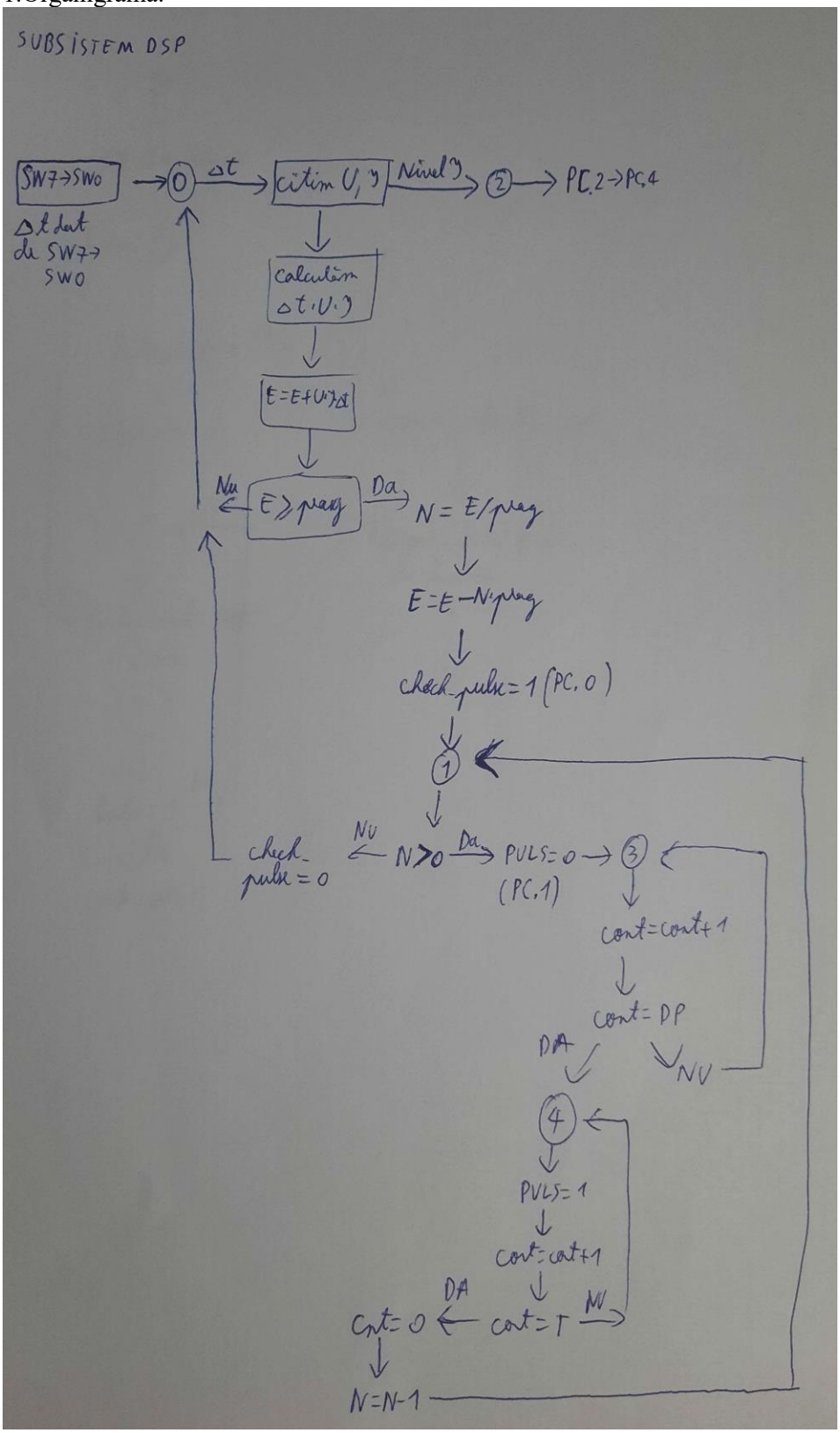
PROIECT 2

Profesor coordonator:Zoican Sorin

Realizat de:
Burlacel George
Busicescu Mihai

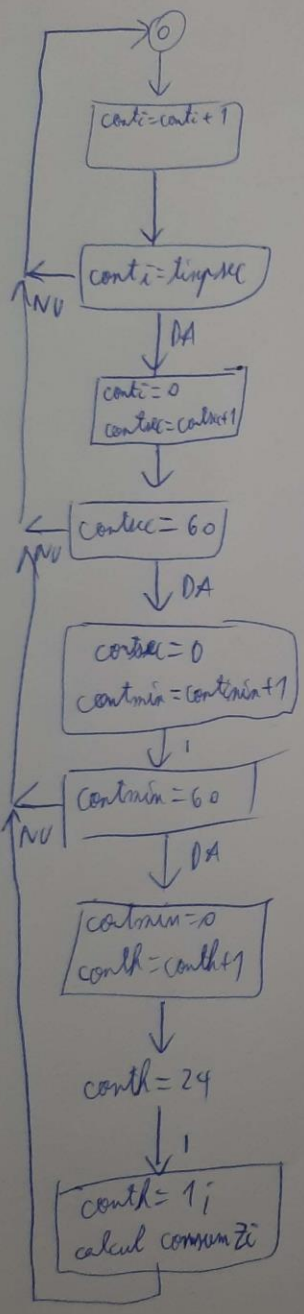
Grupa: 433Db

1.Organigrama:



SUBSISTEM AVR

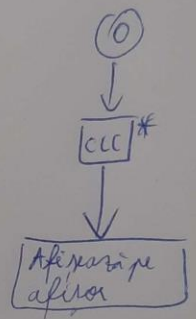
Măsurarea timpului



Tabel adevăr LED-uri stare curentă:

PC.2	PC.3	PC.4	LED 0	LED 1	LED2
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0

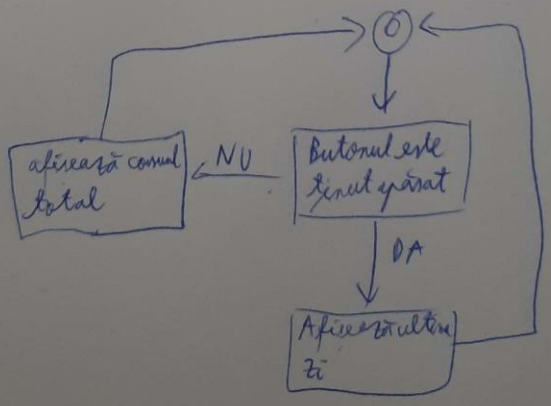
Afișare 2 cifre:



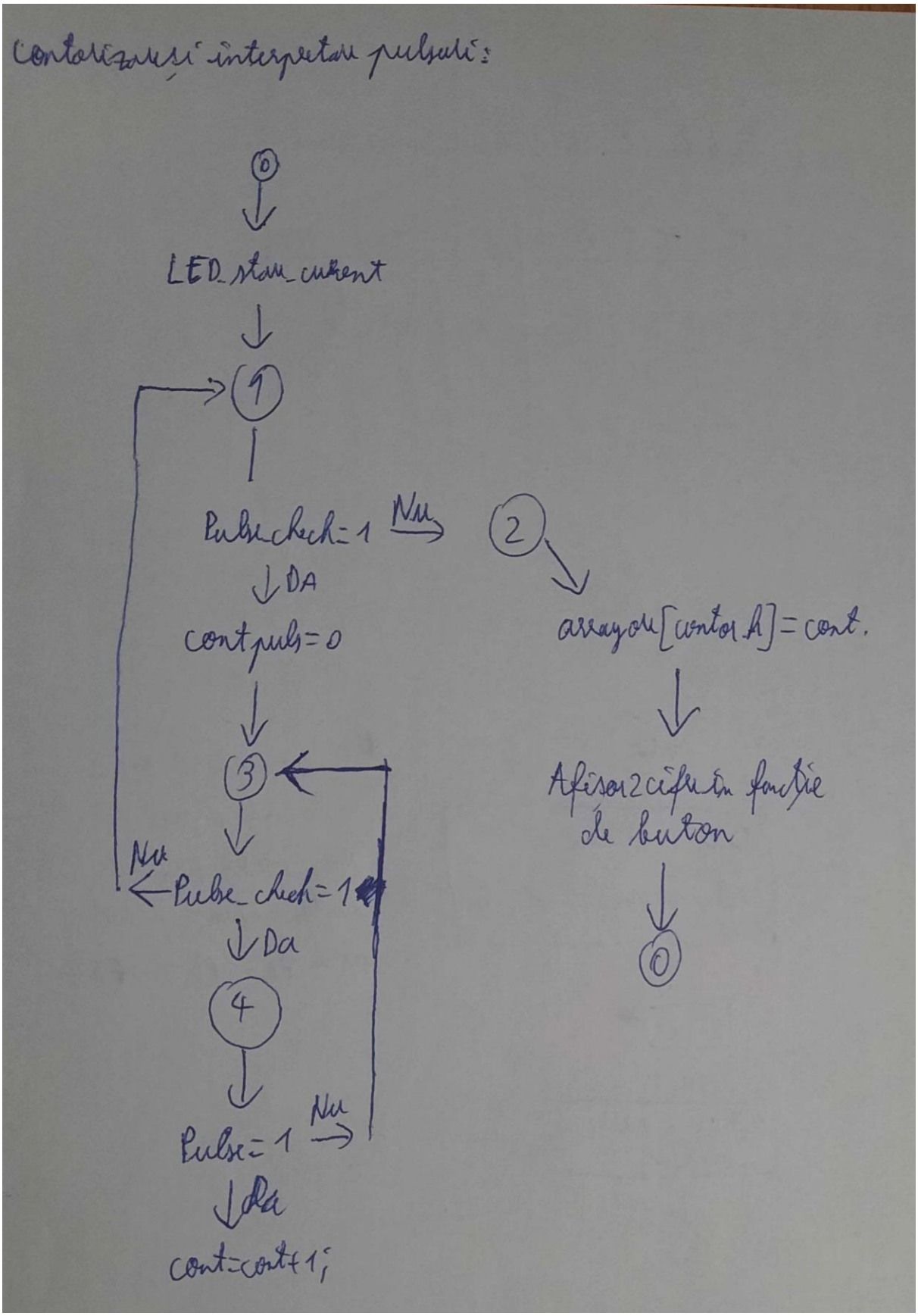
* cifra 0 = valoare % 10;
cifra 1 = (valoare/10) % 10;

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

Butonul:

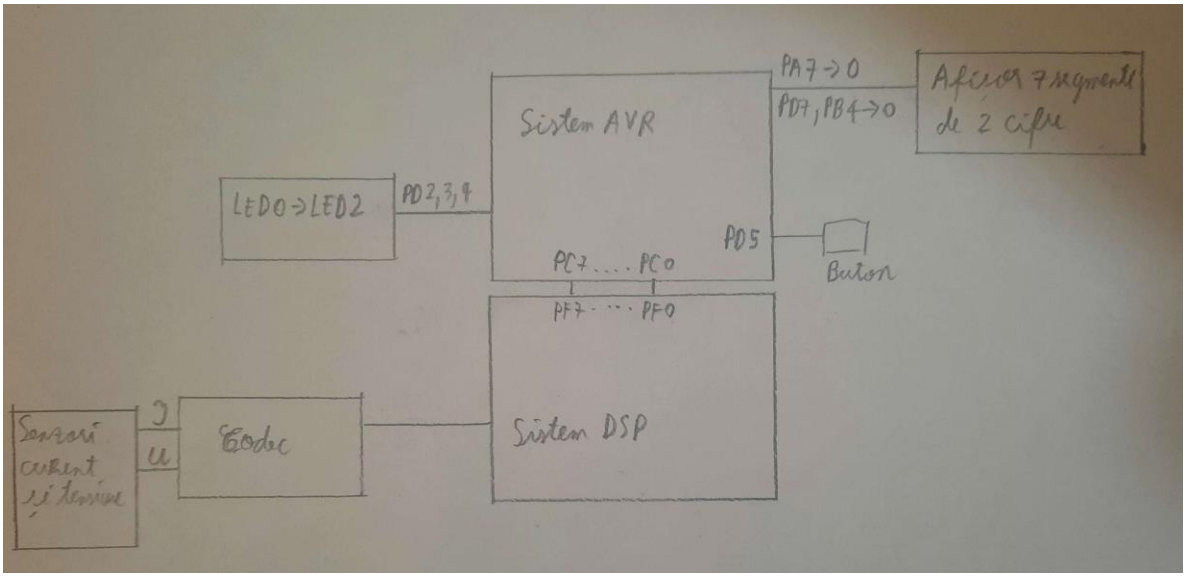


Contorizarea interpretarii pulsului:

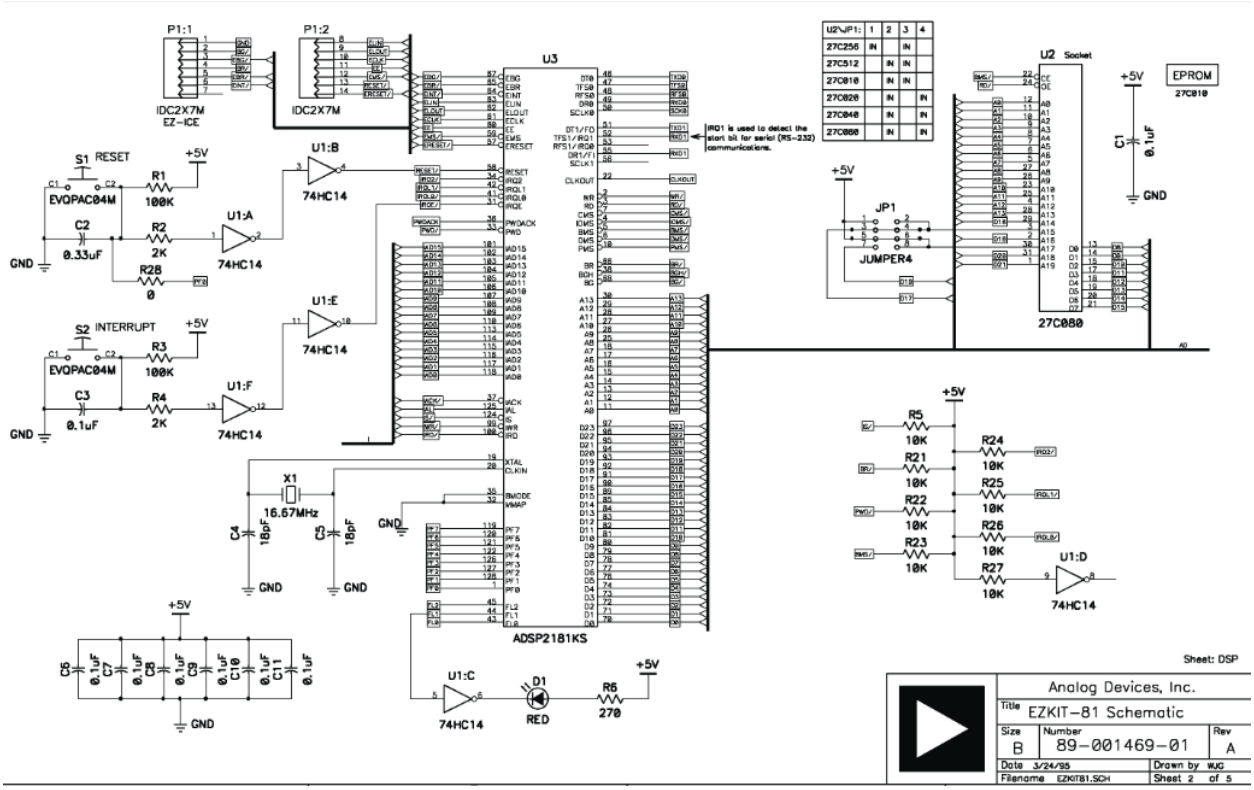


2.Hardware

Schema bloc

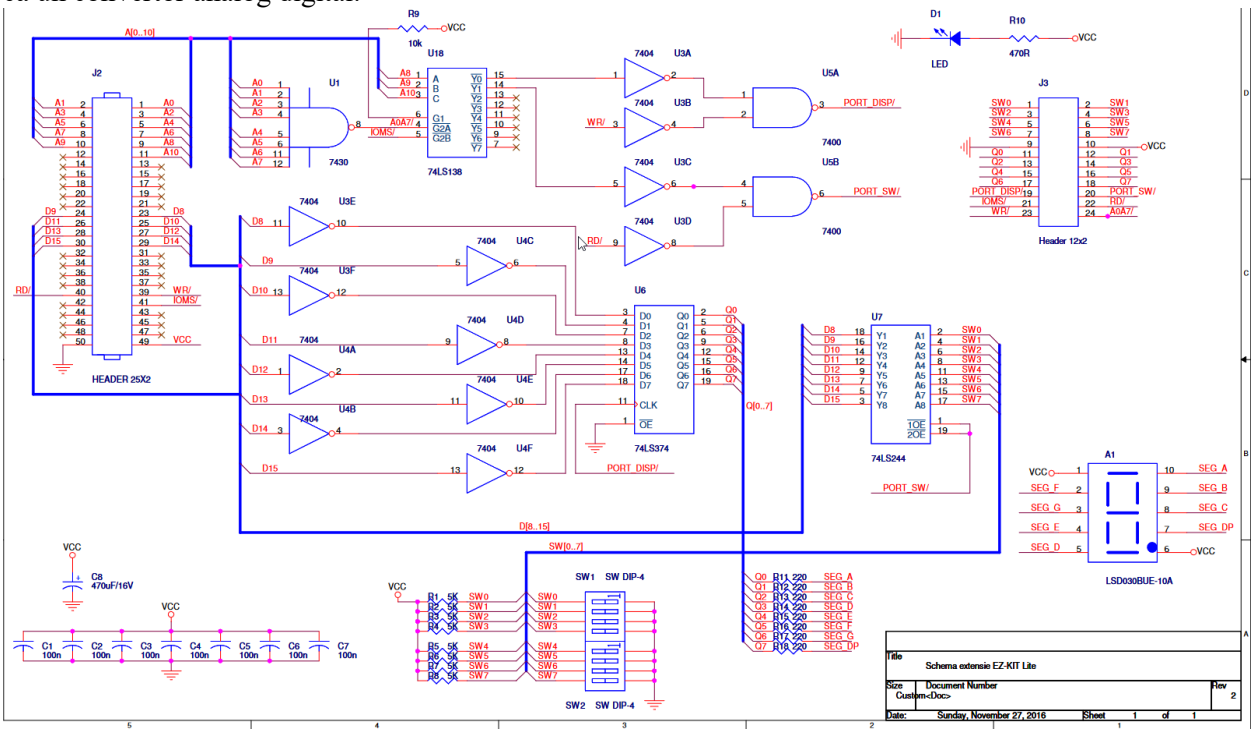


Sistemul DSP



SW1 este folosit pentru a reseta microcontrolerul, acesta muta punctul de masa in fata rezistentei R2 astfel trimite un semnal de “0” logic.
SW2 creaza ointrerupere externa care la apasare trimite “0” logic, acesta muta punctul de masa in fata rezistentei R4.
Cristalul de cuart X1 si condensatoarele C4,C5 formeaza un oscilator de cuart.

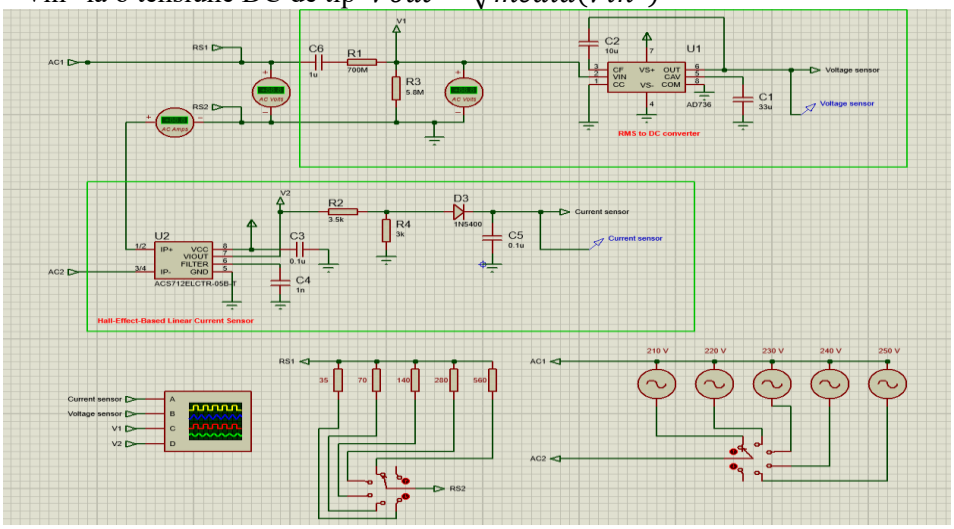
Codec-ul este utilizat pentru a converti semnale analogice in semnale digitale, in sistemul DSP il folosim ca un convertor analog digital.



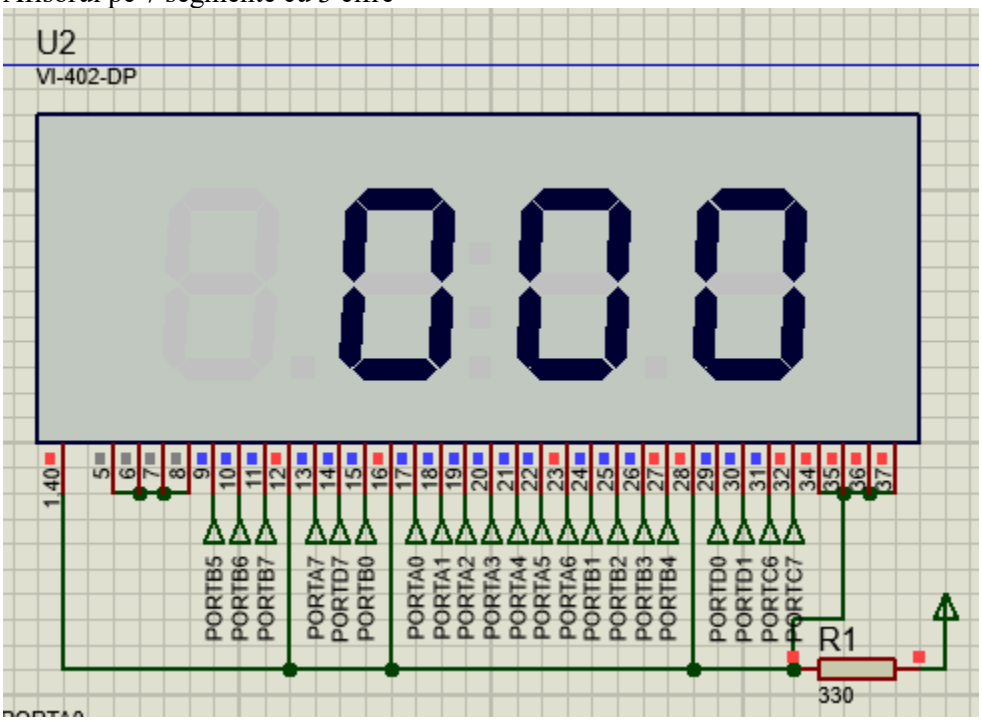
Extensia sistemului DSP ne ofera Switchurile SW7-0 si afisorul de 7 segmente pentru afisarea starii.

Senzori curent si tensiune

Senzorul de Curent conform documetatiei 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 conform documentatiei).
Senzorul de tensiune, conform documentatiei, face conversia de la un semnal de amplitudine "Vin" la o tensiune DC de tip $Vout = \sqrt{media(Vin^2)}$

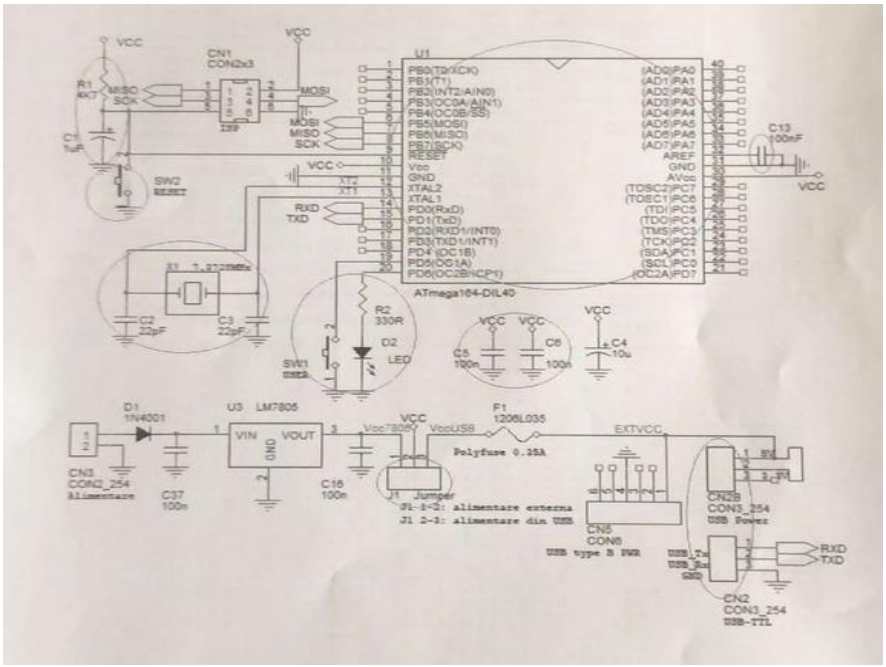


Afisorul pe 7 segmente cu 3 cifre



Cum acesta functioneaza pe logica inversa, pini pentru celelalte cifre si simboluri trebuie mentinuti pe “1” logic pentru a nu 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 condesnatorul C1 se incarca, transmitanduse 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 cuart X1 alaturi de condensatoarele C2,C3 formeaza un oscilator de cuart.
- 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 poziti: 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 conencati la LED0->2, Pini PB0->4, PD7 si PA 7->0 sunt conectati la afisorul de 4 cifre.

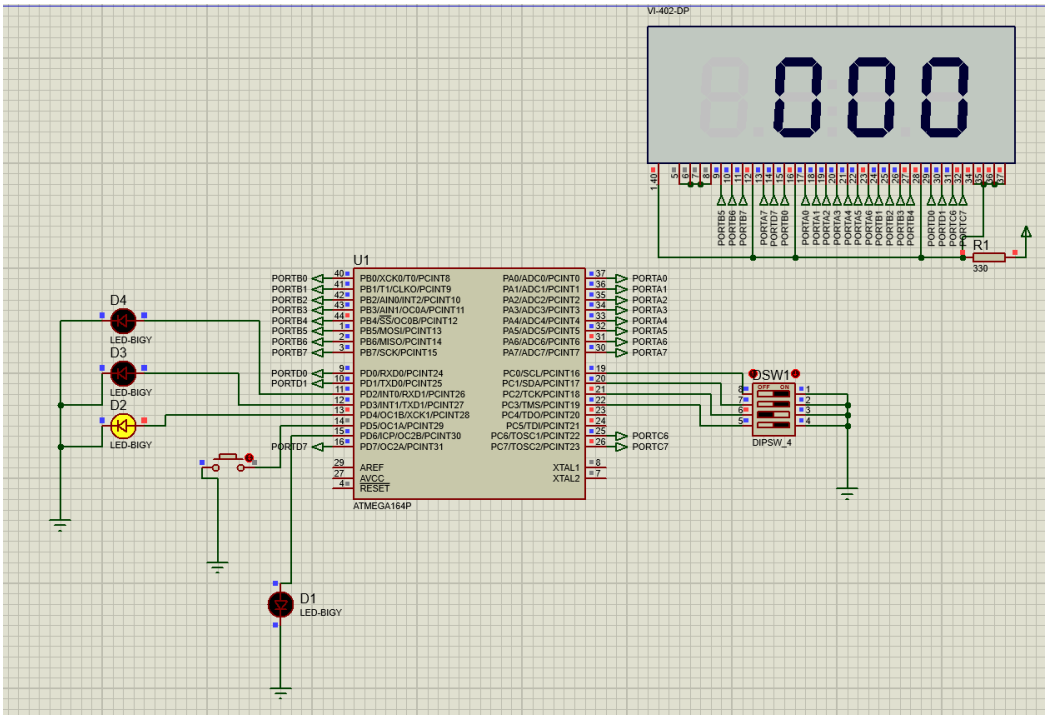
3.Subsistemul AVR

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 reincepe 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.

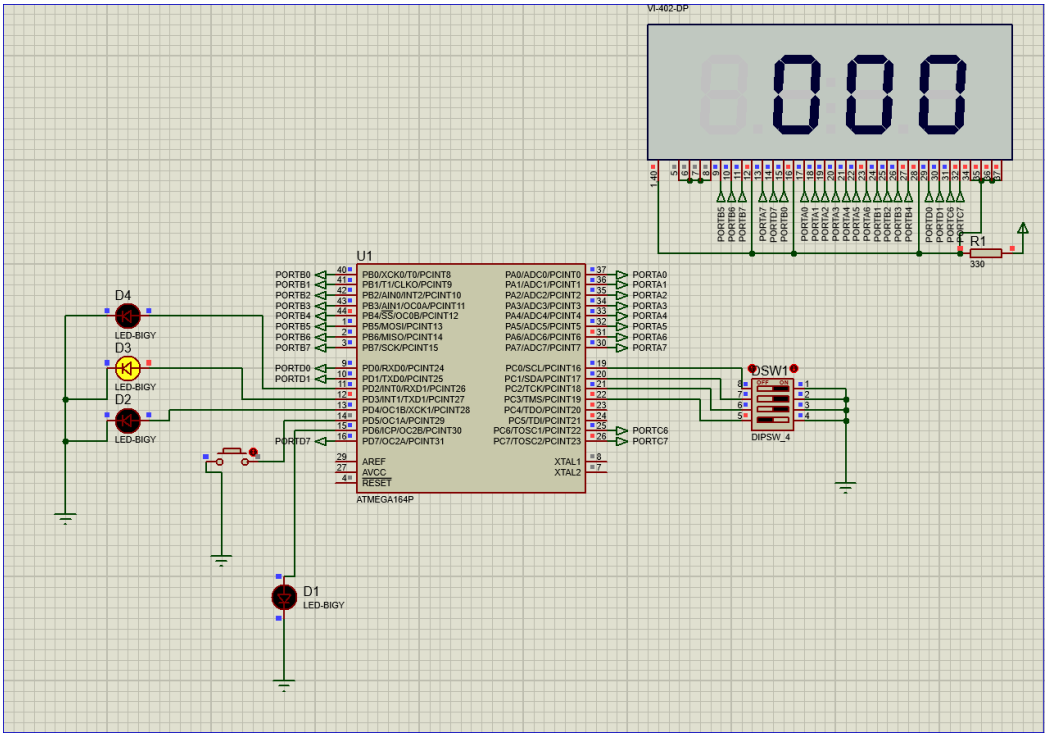
4.Subsistemul DSP
Iesirea de la Senzori este interpretata de CODEC cu timpi de esantionare determinati de SW7-0 din extensia DSP, afisorul din extensie ne afiseaza numarul corespunzator switchului indicator al duratei de esantionare. Sistemul DSP calculeaza valoarea de kWh pana atinge pragul minim de 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.

5.Simulari Proteus

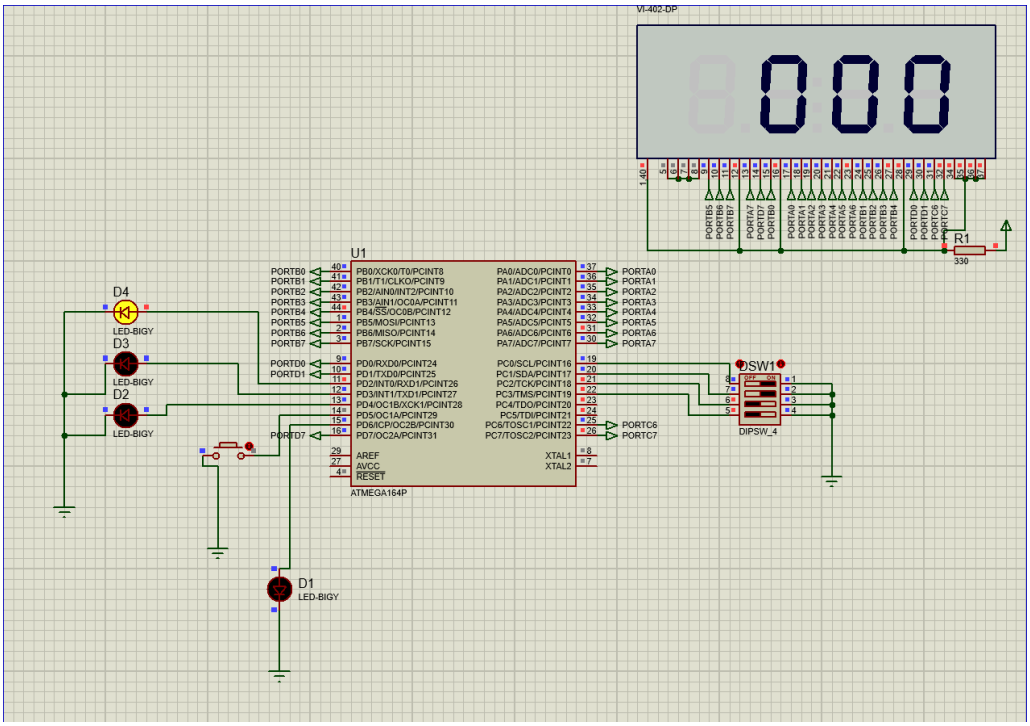
Consum mic de curent (10)



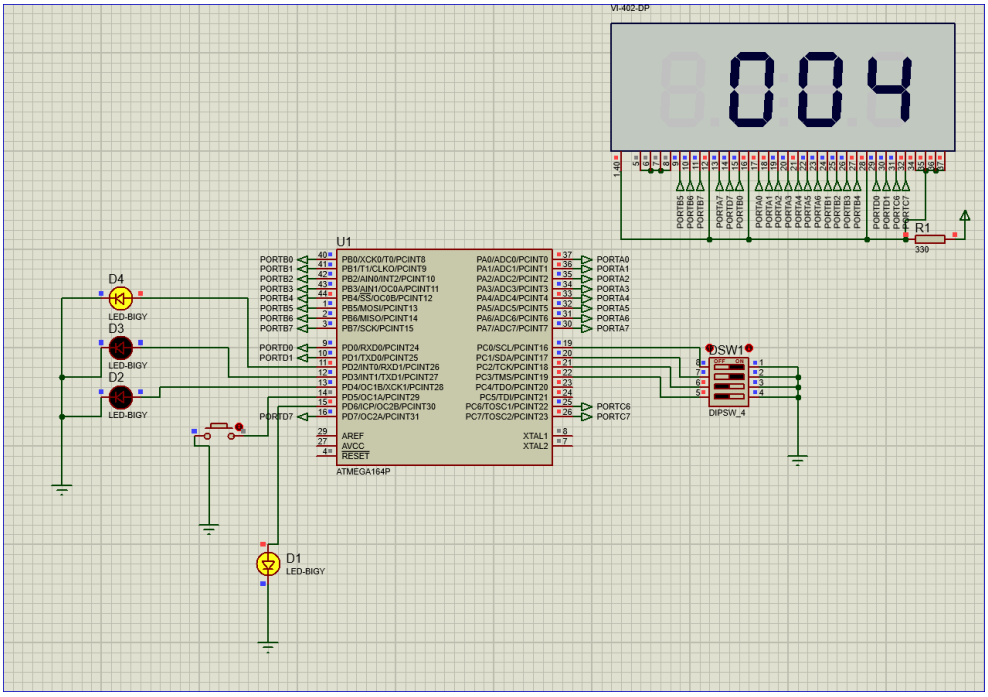
Consum mediu de curent (01)



Consum mare de curent (11)



Afisarea a 4 impulsuri primite.



6.Cod AVR

```
#include <io.h>

// Declare your global variables here
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;
char contorSec=0;
char contorMin=0;
char contorHr=0;
char val_afisor;
char cont_pulse=0;
char totalConsum=0;
char consZi=0;
char j=0;

int ultimStare=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.
//vom folosi pentru simulare un artificiu deoarece simulatorul merge in 4 MHz
TCNT0=0x63;
// Place your code here
contorIn=contorIn+1;
PIND.6=0;
if(contorIn%100==0){
// contorIn pentru 4MHz si TCNT0 ales numara o secunda odata ce ajunge la valoarea de 25
//pentru 20Mhz ar trebui sa fie 100
    contorSec=contorSec+1;
```

```
    contorIn=0;
    PIND.6=1;
}
if(contorSec%60==0){
    contorMin=contorMin+1;
    contorSec=0;

}
if(contorMin%60==0){
contorHr=contorHr+1;
contorMin=0;
}
// o sa ne caculeze consumul la sfarsitul zilei si apoi o sa ne
// treaca in cealalta zi
 if(contorHr%24==0){
 consZi=0;
 for(j=0;j<24;j++)
 {consZi=consZi+arrayOre[j];
 }
  contorHr=0;
  }
}
void LED_Stare_Curent(void)
{
 if(PINC.2==1 && PINC.3==0){   //schimba in port D2, D3, D4   cod 10
 PORTD.4=1;  // reprezinta o valoare scauzta 10mA-1.5ish A
 PORTD.3=0;
 PORTD.2=0;
 }else if(PINC.3==1 && PINC.2==0){
 PORTD.4=0;
```

```
 PORTD.3=1;     // reprezinta o valoare medie  1.501 A -3.5 A   cod 01
 PORTD.2=0;
 }else if(PINC.3==1 && PINC.2==1){
 PORTD.4=0;
 PORTD.3=0;
 PORTD.2=1;   // reprezinta o valoare mare 3.501 A- 5A  cod 11


 }
 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
}
char cifra0;
char cifra1;
char cifra2;


void Afisor_2Cifre(char value){
//vei lua 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 Doamne ajuta
//tine minte afisorul este pe logica inversa
 cifra0=value%10;
 cifra1=(value/10)%10;
 cifra2=(value/100)%10;
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 21 4A
PORTA.5=0;  //pin 22 4F
PORTA.6=1;  //pin 23 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
}

if(cifra2==0){
PORTB.5=0;  // pin 9 2E
PORTB.6=0;  // pin 10 2D
PORTB.7=0;  // pin 11 2C
PORTD.0=0;  // pin 29 2B
PORTD.1=0;  // pin 30 2A
PORTC.6=0;  // pin 31 2F
PORTC.7=1;  // pin 32 2G
}
else if(cifra2==1)
{
PORTB.5=1;  // pin 9 2E
PORTB.6=1;  // pin 10 2D
PORTB.7=0;  // pin 11 2C
PORTD.0=0;  // pin 29 2B
PORTD.1=1;  // pin 30 2A
PORTC.6=1;  // pin 31 2F
PORTC.7=1;  // pin 32 2G

}
else if(cifra2==2)
```

```
{
PORTB.5=0;  // pin 9 2E
PORTB.6=0;  // pin 10 2D
PORTB.7=1;  // pin 11 2C
PORTD.0=0;  // pin 29 2B
PORTD.1=0;  // pin 30 2A
PORTC.6=1;  // pin 31 2F
PORTC.7=0;  // pin 32 2G
}
else if(cifra2==3)
{
PORTB.5=1;  // pin 9 2E
PORTB.6=0;  // pin 10 2D
PORTB.7=0;  // pin 11 2C
PORTD.0=0;  // pin 29 2B
PORTD.1=0;  // pin 30 2A
PORTC.6=1;  // pin 31 2F
PORTC.7=0;  // pin 32 2G

}
else if(cifra2==4)
{
PORTB.5=1;  // pin 9 2E
PORTB.6=1;  // pin 10 2D
PORTB.7=0;  // pin 11 2C
PORTD.0=0;  // pin 29 2B
PORTD.1=1;  // pin 30 2A
PORTC.6=0;  // pin 31 2F
PORTC.7=0;  // pin 32 2G
}
```

```
else if(cifra2==5)
{
PORTB.5=1;  // pin 9 2E
PORTB.6=0;  // pin 10 2D
PORTB.7=0;  // pin 11 2C
PORTD.0=1;  // pin 29 2B
PORTD.1=0;  // pin 30 2A
PORTC.6=0;  // pin 31 2F
PORTC.7=0;  // pin 32 2G
}
else if(cifra2==6)
{
PORTB.5=0;  // pin 9 2E
PORTB.6=0;  // pin 10 2D
PORTB.7=0;  // pin 11 2C
PORTD.0=1;  // pin 29 2B
PORTD.1=0;  // pin 30 2A
PORTC.6=0;  // pin 31 2F
PORTC.7=0;  // pin 32 2G
}
else if(cifra2==7)
{
PORTB.5=1;  // pin 9 2E
PORTB.6=1;  // pin 10 2D
PORTB.7=0;  // pin 11 2C
PORTD.0=0;  // pin 29 2B
PORTD.1=0;  // pin 30 2A
PORTC.6=1;  // pin 31 2F
PORTC.7=1;  // pin 32 2G
}
```

```
else if(cifra2==8)
{
PORTB.5=1;  // pin 9 2E
PORTB.6=1;  // pin 10 2D
PORTB.7=1;  // pin 11 2C
PORTD.0=1;  // pin 29 2B
PORTD.1=1;  // pin 30 2A
PORTC.6=1;  // pin 31 2F
PORTC.7=1;  // pin 32 2G
}
else if(cifra2==9)
{
PORTB.5=1;  // pin 9 2E
PORTB.6=0;  // pin 10 2D
PORTB.7=0;  // pin 11 2C
PORTD.0=0;  // pin 29 2B
PORTD.1=0;  // pin 30 2A
PORTC.6=0;  // pin 31 2F
PORTC.7=0;  // pin 32 2G
}
}



char ValAfisorButon(void){
char SUMA=0;
char i;
if(PIND.5==0)//buton apasat   butonul este portD5, cu portD6 allways on pentru LED
 {// ne va afisa ultimele 24 de ore de consum
  SUMA=0;
  if(consZi==0)
```

```
 {if(contorHr<24)
    { for(i=0;i<contorHr+1;i++)
     {   SUMA=SUMA+arrayOre[i];
     // ne va arata consumul pana la ora curenta de la pornire
     }
     val_afisor=SUMA;
     }
  else
  { for(i=0;i<contorHr;i++)
     {   SUMA=SUMA+arrayOre[i];
     // ne va arata consumul pana la ora curenta de la pornire in caz ca consZi e 0
     }
     val_afisor=SUMA;
  }
 }
 else{
 val_afisor=consZi;
 }

 //sfarsitul buton apasat
}
  else
 {val_afisor=totalConsum;
      }
  return val_afisor;
 }


void PinStareMod(int pinStare)
{
```

```
if(pinStare!=ultimStare){
if(pinStare==0){
 cont_pulse=cont_pulse;
 }
 else{
  cont_pulse=cont_pulse+1;
 }
 ultimStare=pinStare;
 }
 }
```

```
void main(void)
{
// Declare your local variables here
 //DSP ar trebui sa trimita ceva de genul 00100 pentru un puls simplu
// Crystal 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=In Bit6=In Bit5=In 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=T Bit6=T Bit5=T 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=(1<<DDC7) | (1<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) | (0<<DDC1) |
(0<<DDC0);

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

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


// Port D initialization

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

```
DDRD=(1<<DDD7) | (1<<DDD6) | (0<<DDD5) | (1<<DDD4) | (1<<DDD3) | (1<<DDD2) | (1<<DDD1)
| (1<<DDD0);

// State: Bit7=1 Bit6=0 Bit5=T Bit4=0 Bit3=0 Bit2=0 Bit1=T Bit0=T

PORTD=(1<<PORTD7) | (1<<PORTD6) | (0<<PORTD5) | (1<<PORTD4) | (1<<PORTD3) |
(1<<PORTD2) | (1<<PORTD1) | (1<<PORTD0);


// Timer/Counter 0 initialization

// Clock source: System Clock

// Clock value: 19.531 kHz

// Mode: Normal top=0xFF

// OC0A output: Disconnected

// OC0B output: Disconnected

// Timer Period: 10.035 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)
    {
    LED_Stare_Curent();
```

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

Cod DSP

```
.section/dm data1;
// intrari
.var E;
.var U;
.var I;
.var P;
.var A;
.var ;
.var invprag;
.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(invprag);  //in y este incarcata valoarea pragului
MR= 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
```

```
CNTR=DM(N);
DO sqr UNTIL CE
sqr:
AY0=0x0FFF;
AR=AX0, AF=AX0 AND AY0;
output=AR;
AR=AX0, AF=AX0 AND AY0;
NOT AR;
output=AR;

stop: jump stop;
```