Universitatea Politehnica Bucuresti
Facultatea de Electronica,Telecomunicatii si Tehnologia Informatiei

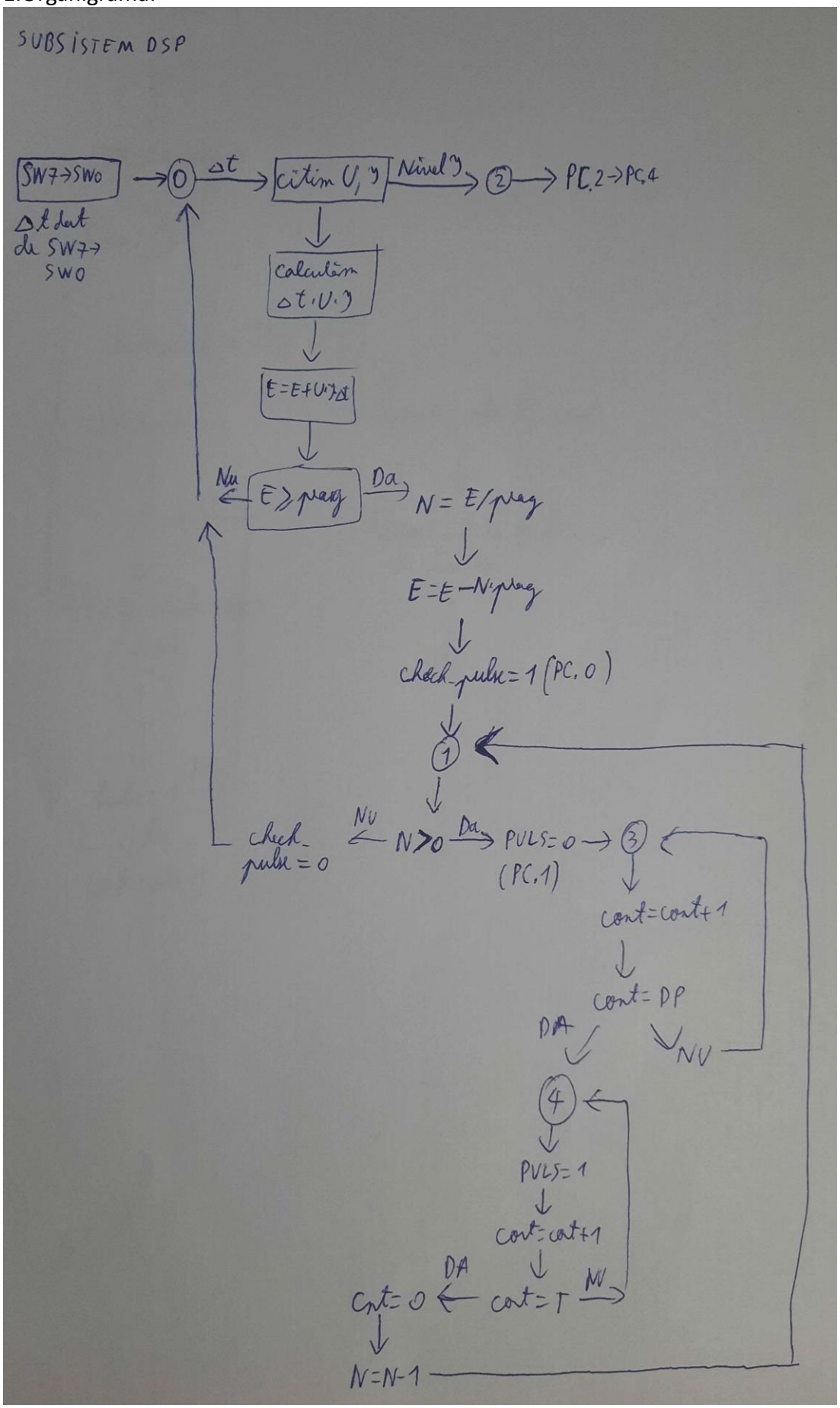
PROIECT 2

Profesor coodronator:Zoican Sorin

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

Grupa: 433Db

1.Organigrama:



SUBSISTEM AVR

Măsurarea timpului:

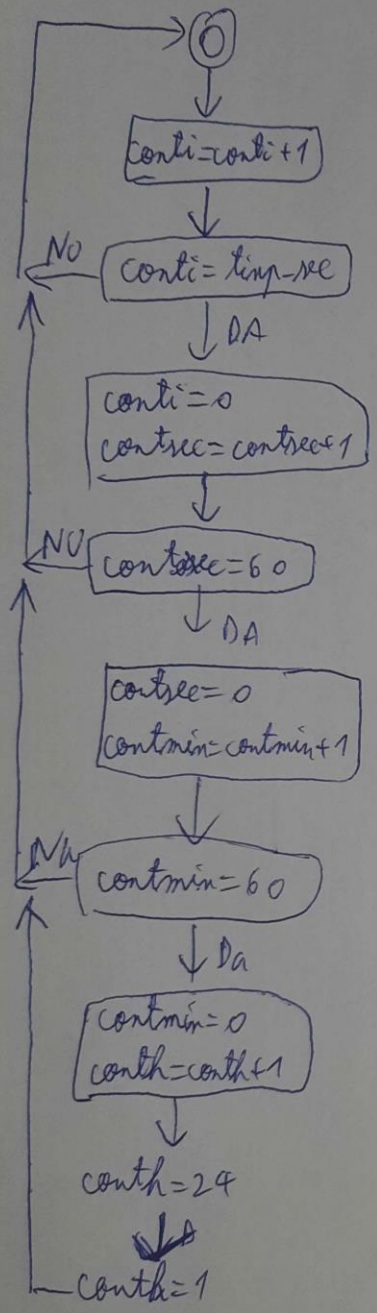
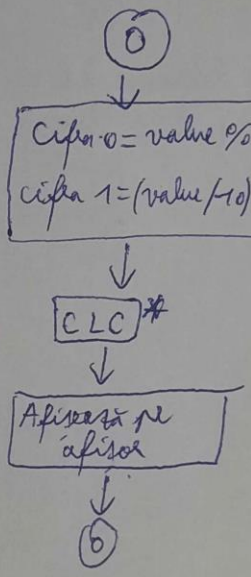


Tabela adrese LED-uri stare curentă:

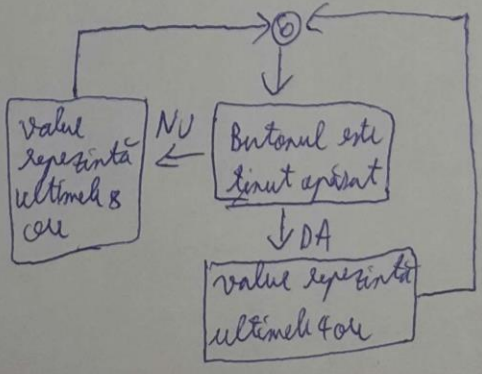
PC.2	PC.3	PC.4	LED 0	LED 1	LED 2
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0

Afisor 2 cifre:

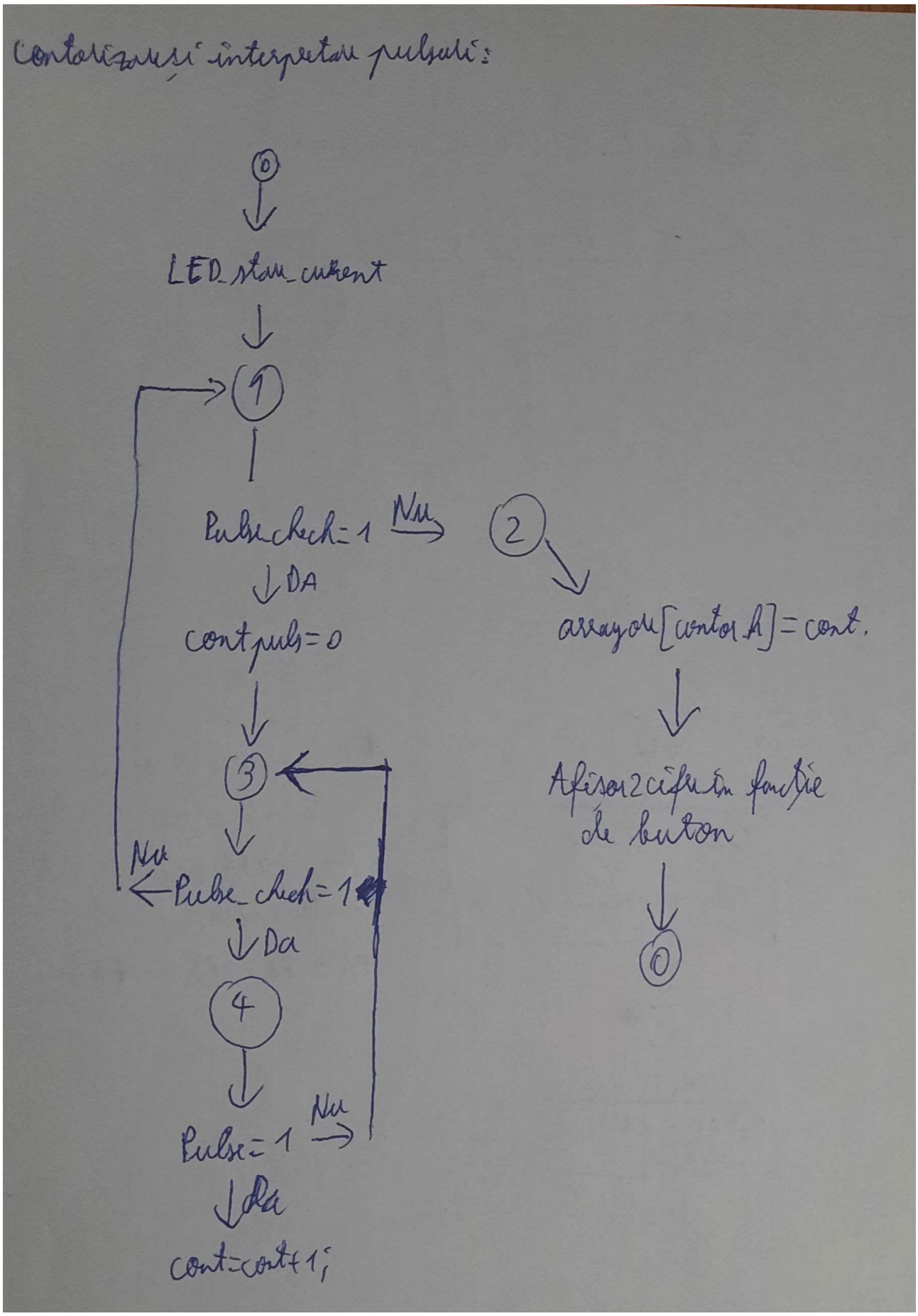


* Pentru cifra 0 s-au alocat pinii PA.0 → PA.6 iar pentru cifra 1 pinii PA.7; PD.7; PB.0 → PB.4

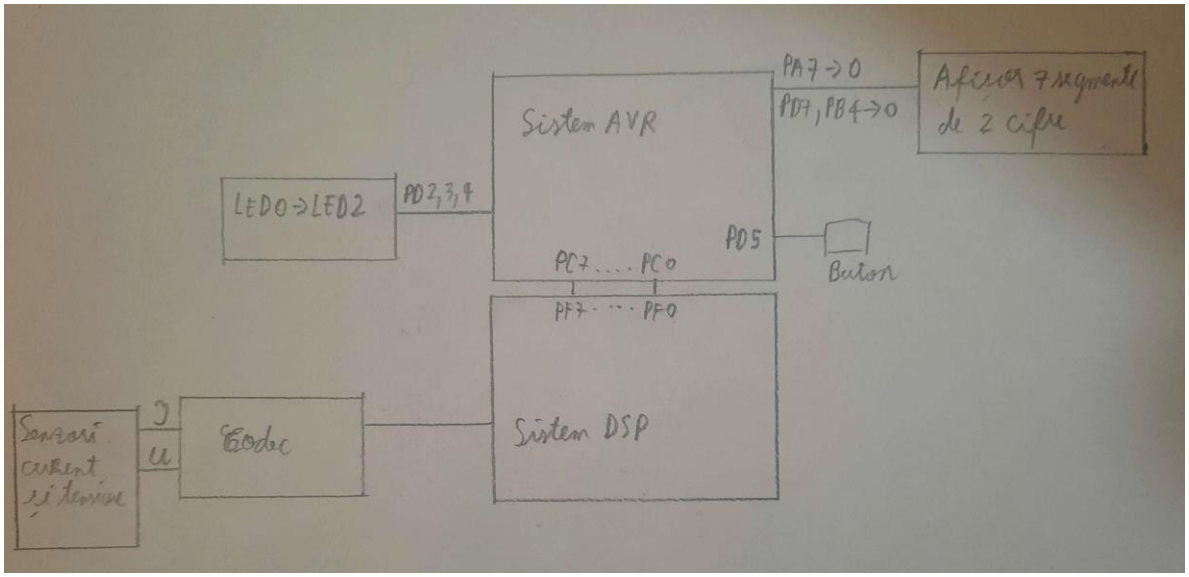
Butonul:



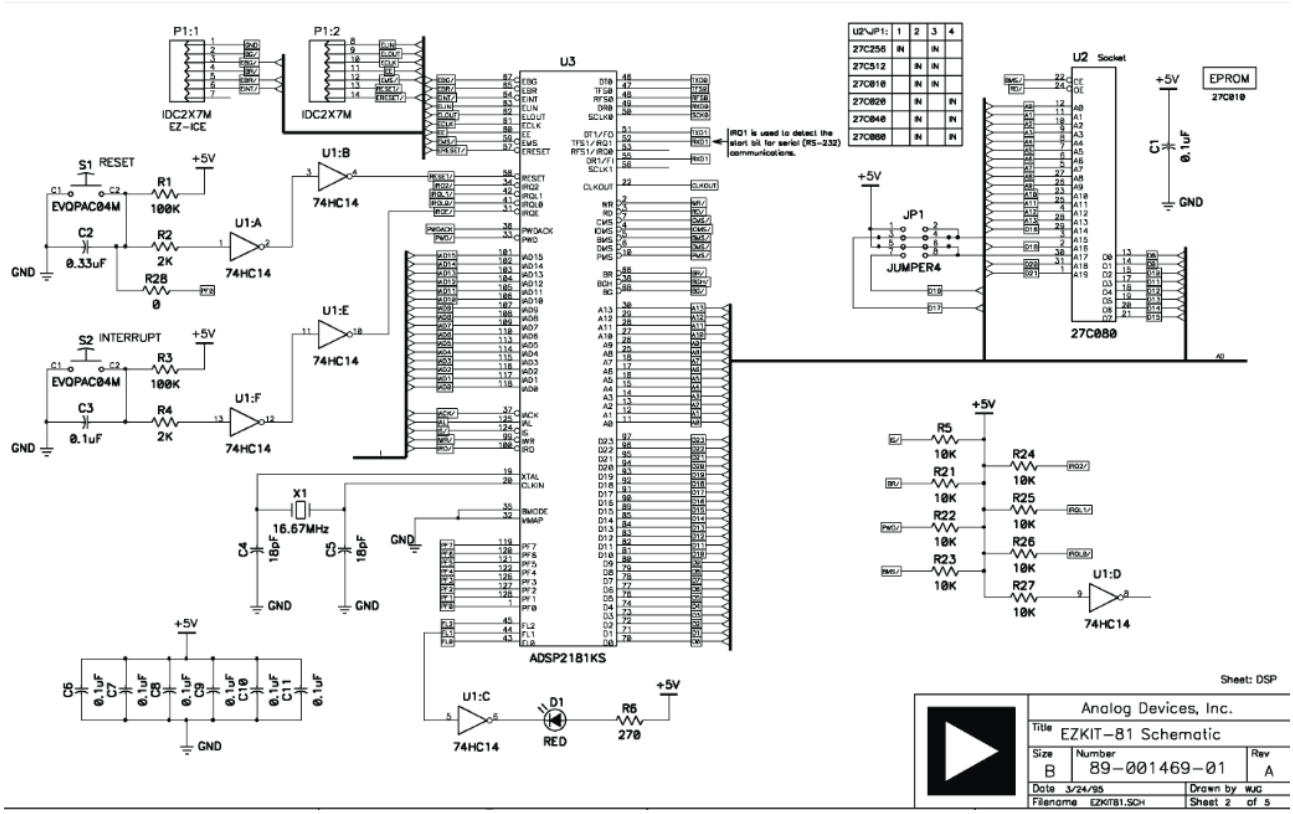
Contorizarea interpretarii pulsului:



2.Hardware
Schema bloc

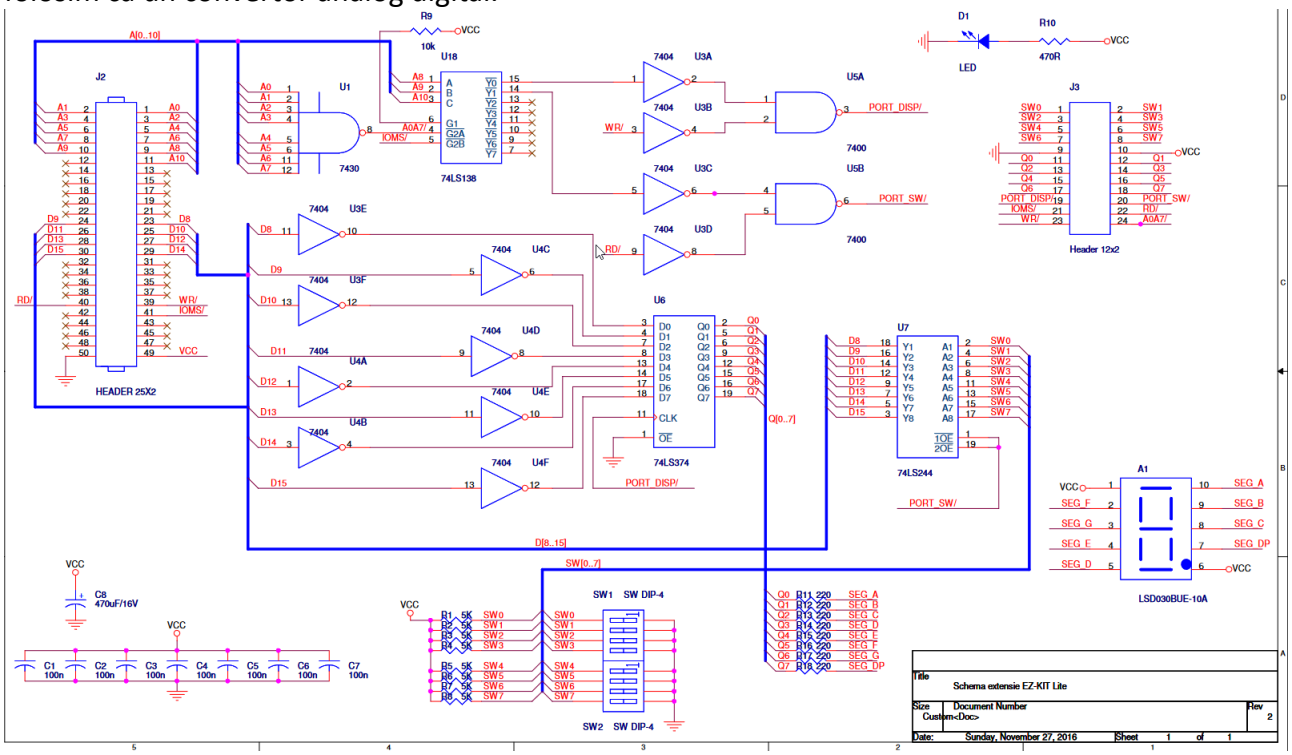


Sistemul DSP



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.

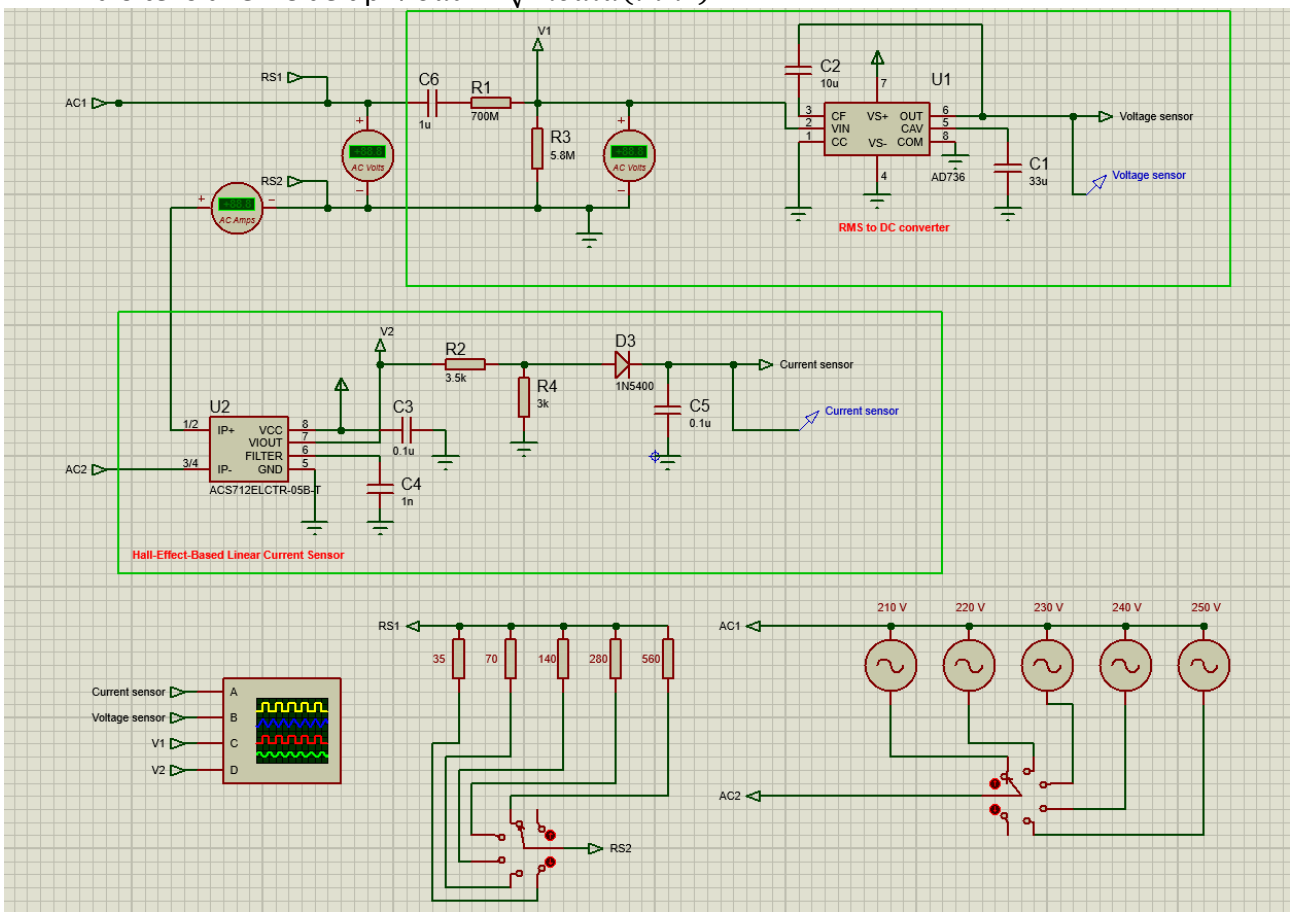
Codec-ul este utilizat pentru a converti semnale analogice in semnale dicitale, in sistemul DSP il folosim ca un convertor analog digital.



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

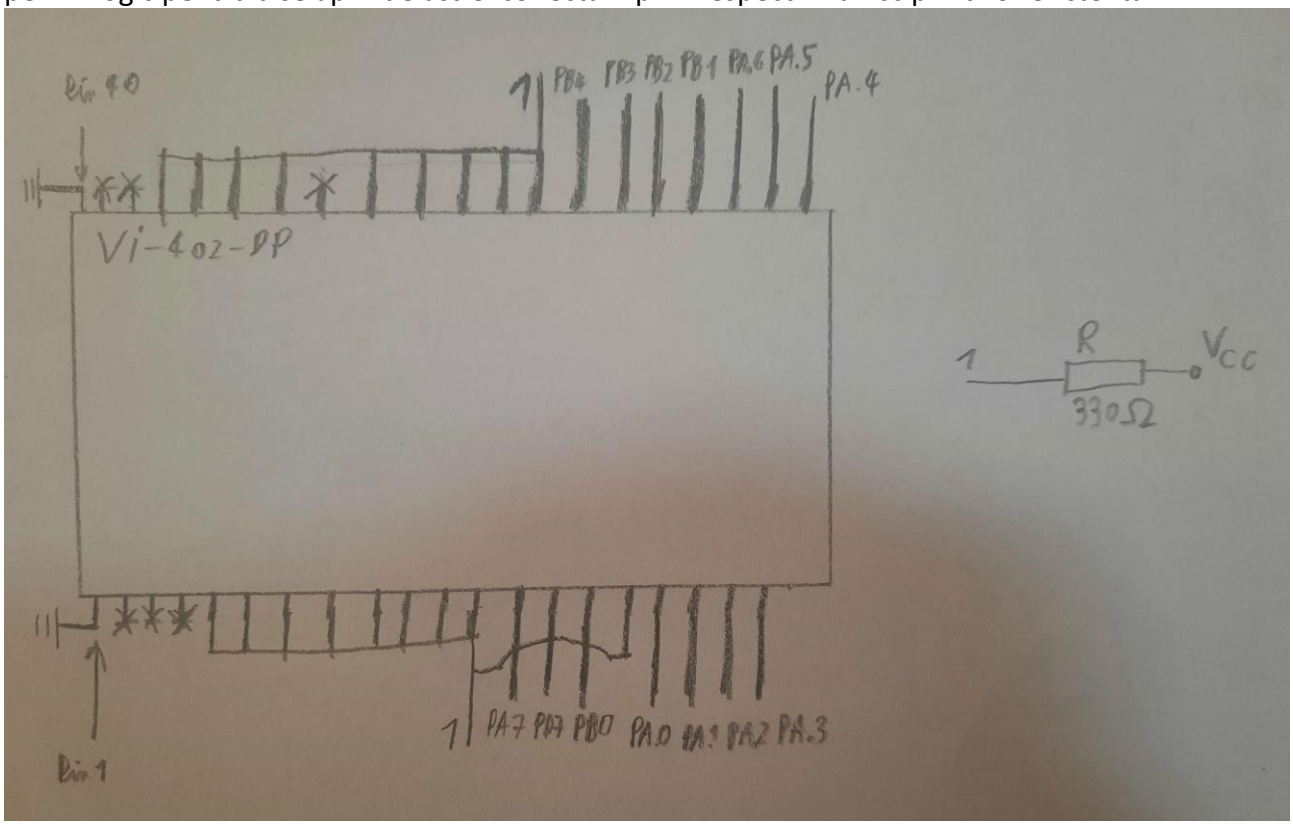
Senzori curent si tensiune

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

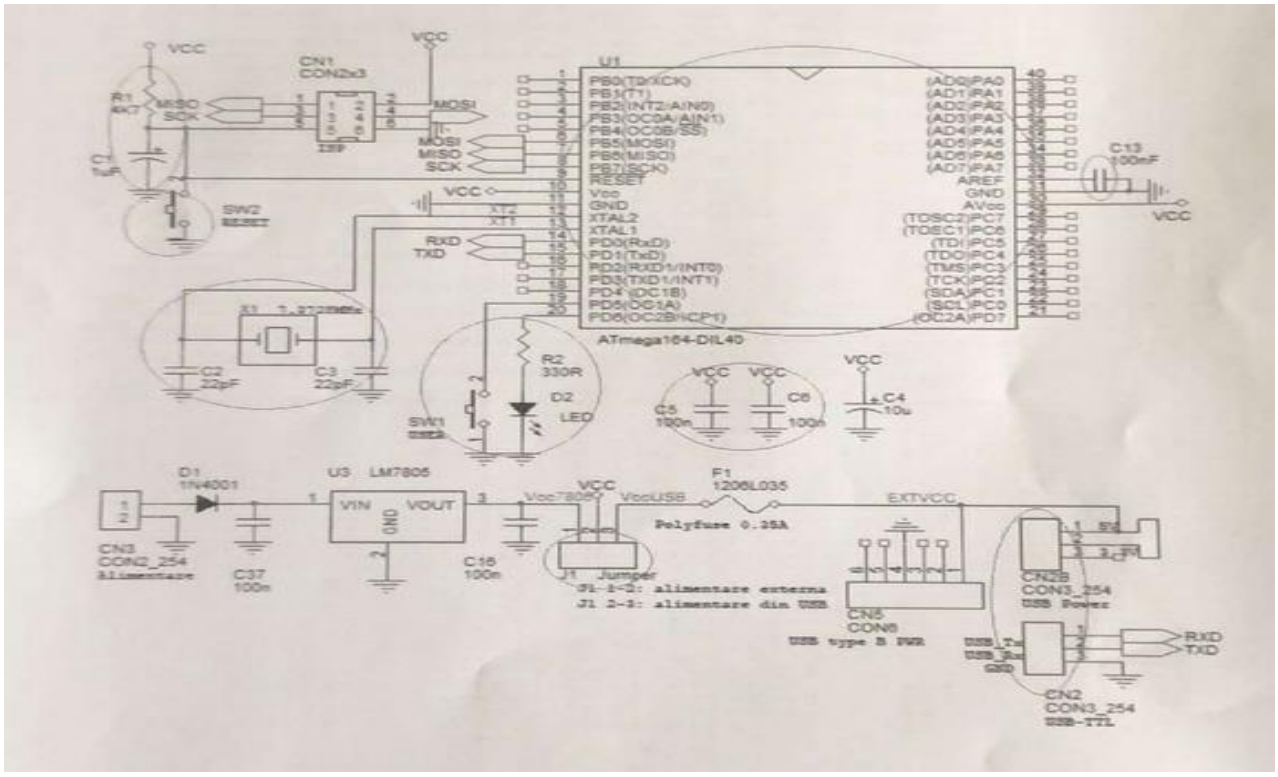


Afisorul pe 7 segmente cu 2 cifre

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 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.
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.Cod AVR

Codul din document exclude codul generat automat de CodeVision AVR.

Main

```
void main(void)
{
while (1)
{
//check_pulse=PINC.0;//portile pentru primirea datelor
//in_semnal_p=PINC.1;
LED_Stare_Curent();
if(PINC.0==1 ) //fal portC.0
{
cont_pulse=0;
while(PINC.0!=0) // fal port C.1
{if(PINC.1==1) // DSP ar trebui sa trimita semnal de genul 1110111
{
cont_pulse=cont_pulse+1;
}
}
}
arrayOre[contorHr]=cont_pulse;
Afisor_2Cifre(ValAfisorButon());
}
}
```

Configurarea temporizatorului TIM0 pentru a contoriza secundele minutele si orele.

```
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=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.
//vom folosi pentru simulare un artificiu deoarece simulatorul merge in 4 MHz
TCNT0=0x63;
// Place your code here
contorIn=contorIn+1;
```

```
if(contorIn%25==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;
}
if(contorSec%60==0){
contorMin=contorMin+1;
contorSec=0;
}
if(contorMin%60==0){
contorHr=contorHr+1;
contorMin=0;
}
if(contorHr%24==0){
contorHr=1;
}
}
```

Functiile pentru afisarea nivelului

```
void LED_Stare_Curent(void)
{
if(PINC.2==1){ //schimba in port D2, D3, D4
PORTD.4=1; // reprezinta o valoare scauzta 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
}
```

Functia pentru functionarea butonului.

```
char ValAfisorButon(void){
char SUMA;
//ar trebui sa ma mai gandesc cu butonul ca un fel de multi switch ca nu e un switch
//ar trebui sa il pun intr-o
char buffer_calc;
char i;
if(PIND.5==0)//buton apasat butonul este portD5, cu portD6 allways on
{
SUMA=0;
if(contorHr>8)
{ buffer_calc=contorHr-8;
for(i=buffer_calc;i<=contorHr;i++)
{ SUMA=SUMA+arrayOre[i];
}
}
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 24 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;
}
}
```

Pentru a ne afisa cele 2 cifre conform foi de catalog a (pune numele afisorului aici) acesta lucreaza pe o logica inversa.

```
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
//placeholder pentru porti liberi pentru 2 cifre
//tine minte afisorul este pe logica inversa
cifra0=value%10;
cifra1=(value/10)%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 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
}
}
```

6.Cod DSP

SQRGEN.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
MX0=DM(U);
MY0=DM(I);
MR=MX0*MY0;
DM(P)=MR;
MX0=DM(timp);
MY0=DM(P);
MR=MR+MX0*MY0;   //calculul energiei

DM(E)=MR;
MX0=DM(E);
MY0=DM(prag);
MR=DIVS MX0, MY0;
DM(N)=MR;
// daca N > 0 atunci se genereaza un puls



stop: jump stop;
```