**UNIVERSITATEA POLITEHNICĂ DIN BUCUREȘTI**

FACULTATEA DE ELECTRONICĂ, TELECOMUNICAȚII ȘI TEHNOLOGIA INFORMAȚIEI

**PROIECT 2**

**SISTEM DE CONTORIZARE ȘI AFIȘARE A CONSUMULUI DE ENERGIE ELECTRICĂ**

**STUDENȚI:**

LEFTERACHE ALEXANDRU-GABRIEL 431A

CRISTESCU ANA-IOANA 433D

MARINESCU AURORA-CRISTINA 433D

OPREA LIVIA-DANIELA-MIHAELA 433D

**PROFESOR COORDONATOR:**

ZOICAN SORIN

**2022-2023**

**CUPRINS**

**PREZENTAREA TEMEI**

Tema constă în realizarea unui sistem care contorizează și afișează consumul de energie electrică și este compus din două subsisteme (AVR și DSP). Arhitectura acestora este reprezentată în figura 1:

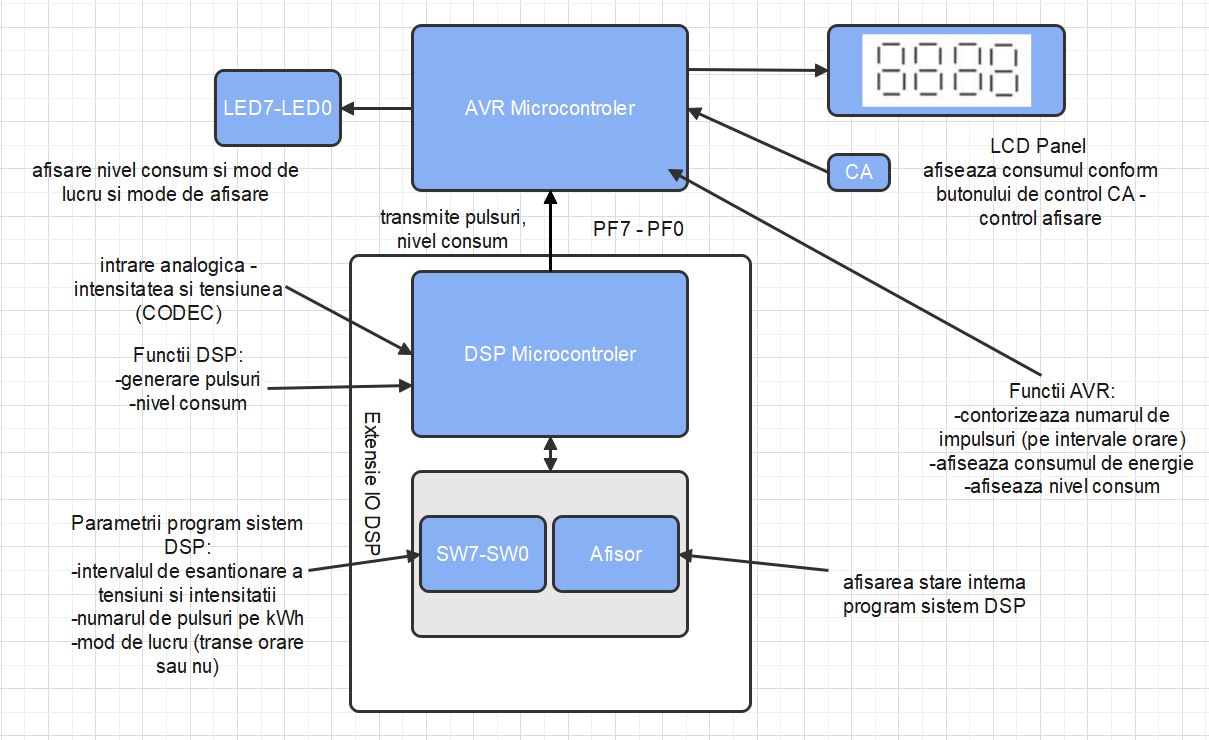


Figura 1

Subsistemul DSP are în componența sa placa de evaluare EZ-Kit LITE ADSP2181 și o interfață de intrare-ieșire, IO DSP. Acesta realizează măsurarea tensiunii și intensității, calcularea energiei consumate și generarea unui număr P de pulsuri pentru 1kWh. Pulsurile generate și valoarea intensității curentului consumat se vor transmite pe un pin al portului PF. Se consideră puterea maximă de 10 kW. Intervalul de eșantionare a tensiunii si intensității ΔT si numărul de pulsuri P per kWh, se vor stabili din SW7-SW0. În cadrul temei, s-a stabilit un mod de lucru al contorizării energiei cu/fără intervale orare.

Subsistemul AVR utilizează un microcontroler ATMega164. Acesta realizează contorizarea pulsurilor, afișarea și calcularea energiei consumate conform modului indicat de subsistemul DSP.

Implementarea la nivel hardware: subsistemul AVR (cu microcontroler ATMega164) și extensia IO DSP

Implementarea la nivel software: descrierea formală a programelor pentru subsistemele AVR și DSP, scrierea codului pentru cele 2 subsisteme ( în limbaj de programare C pentru AVR și în limbaj de asamblare ADSP2181 pentru subsitemul DSP) și testarea programelor în CVAVR și ASTUDIO, respectiv în Visual DSP++ 3.5.

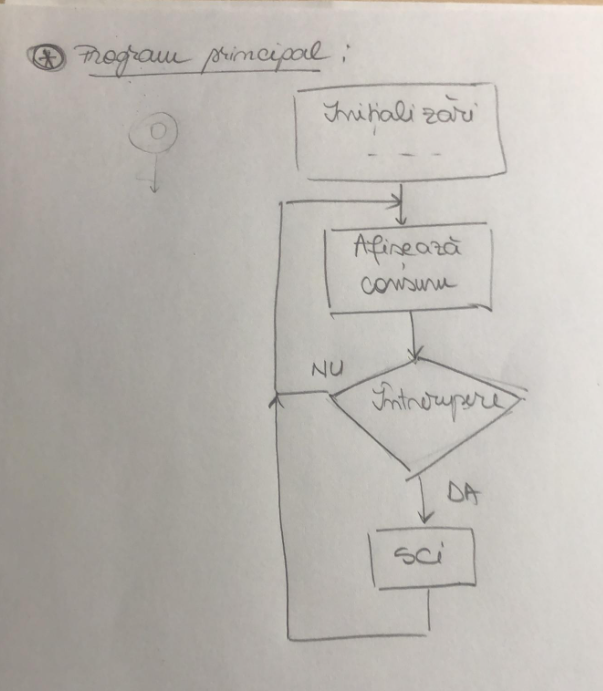
În final se va verifica funcționalitatea sistemului fizic realizat.

**DESCRIEREA PRELUCRĂRILOR - GRAFURI ȘI ORGANIGRAME**

**Organigrama program principal->poza**

**Explicatii: initializari variabile globale, tabele clc, tabele de semnale relevante (cls)**

**Afiseaza consum care are corespondentul display consumption fctia care face afisaeaa consumului apoi el asteapta intreruperi, el face asta in bucla pogramului principal, in main : de fiecare data cadb tpocesorul teece prin bucla while afiseazaa cate un digit, procesorul are f mare si pacaleste ochiul ca si cand ar fi continuu, el afiseaza pe rand dar repede; daca apare intrerupere, se intrerupe-rutina de intreruperi; dupa intrerupere se duce in rutina de servire a intreruperii, programel**



Pentru sistemul de contorizare și afișare a consumului de energie, se vor implementa două procese secvențiale (PS) astfel: unul pentru măsurarea energiei, după verificarea duratei cu un contor de pulsuri și altul pentru măsurarea timpului cu afișare pe 4 cifre pe intervalele de "+".

În cadrul procesului secvențial asociat contorizării timpului, cu organigrama în figura 2, vom defini câte o variabilă specifică fiecărei unități de timp:

H - hour; D - day; M - minutes; S - seconds;

Poza refacuta

Figura 2

De asemenea, celelalte variabile din cadrul organigramei au semnificații, după cum urmează:

* cnt\_time - contorul de timp
* T\_SEC - numărul de perioade necesare pentru a acoperi o secundă

Pseudocodul asociat programului pentru contorizarea timpului va începe din starea 0, având contorul cnt\_time=0.

Valorile inițiale ale variabilelor H, D, M, S pot fi inițializate cu valoarea 0 sau pot fi inițializate cu valorile reale la începutul funcționării.

void contorizare\_timp (){

cnt\_time + = 1; // incrementare contor de timp

dacă cnt\_time ! = T\_SEC

return ;

altfel // cnt\_time = T\_SEC

{

cnt\_time = 0; //resetează contorul

S + = 1; //incrementeză contor secunde

}

dacă S=60 //au trecut 60 de secunde

{

S = 0; //resetează contor secunde

M + = 1; // incrementează contor minute

}

altfel

return; //nu au trecut 60 de secunde, deci nu este necesară //incrementarea minutelor și prin urmare nu este necesar să //continuăm

dacă M = 60 //nu au trecut 60 de minute

{

M = 0; //resetează contor minute

H + = 1; // incrementează contor ore

}

altfel

return; //nu au trecut 60 de minute, deci nu este necesară //incrementarea orelor și prin urmare nu este necesar să //continuăm

dacă H = 24 //nu au trecut 24 de ore

{

H = 0; //resetează contor ore

Z + = 1;

}

altfel

return;

dacă Z =7 // am încheiat săptămâna

Z = 0; // resetare zi

return; //sfârșit funcție de contorizare timp

}

Codul în limbaj DSP/AVR este:

Alex