Sisteme Încorporate

**Implementarea unui calculator pentru operaţii aritmetice de bază pe placa Dragon12**

Proiect realizat de:

Cerna Silviu-Vlad

Fronea Alexandru

Grupa 3.1, Anul 3 CTI

Implementarea unui calculator pentru operaţii aritmetice de bază pe placa Dragon12 (2 studenţi)

Caracteristici:

• Se va folosi tastatura integrată pe placa Dragon12 pentru introducerea operanzilor şi a operatorilor;

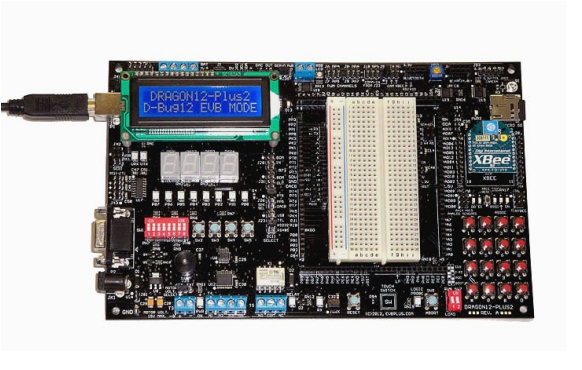
• Se vor implementa cel puţin cele 4 operaţii aritmetice de bază (adunare, scădere, înmulţire, împărţire);

• Operanzii, operatorii şi rezultatul vor fi afişate pe LCD-ul plăcii;

• Se va asigura afişarea rezultatelor care sunt numere întregi, precum şi a celor reale;

• Se va afisa un mesaj de eroare în cazul împărţirii la 0.

Caracteristici generale



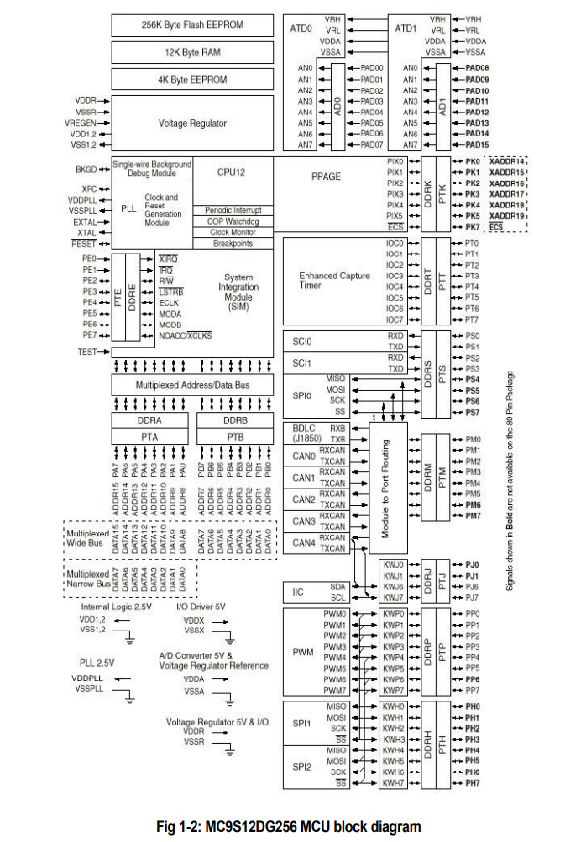
Principalele caracteristici ale plăcii Dragon12-Plus2 sunt:

Interfață USB, port de comunicare RS485, port de extindere I2C, port CAN, port de expansiune SPI, afișaj LED cu 7 seg., opt LED-uri, comutator DIP cu opt poziții, 4 comutatoare cu butoane, LED de culoare RGB, transmițător IR cu oscilator de 38 kHz la bord, sonde logice cu indicatoare LED, senzori de temperatură și de lumină, slot pentru cartele de memorie MicroSD, interfața TTL a camerei JPEG, interfețe XBee (Wi-Fi), Bluetooth, punte-H duală de înaltă eficiență pentru controlul motoarelor de curent continuu sau a unui motor pas cu pas, etc.

Placa Dragon12-Plus este livrată cu MC9S12DP256CCPV sau MC9S12DG256CVPE instalat. Vom folosi varianta cu DG256, care are 2 porturi CAN.

Descrierea hardware pentru familia HCS12

( Diagrama bloc a unității microcontrolerului)



Pinii unității microcontrolerului

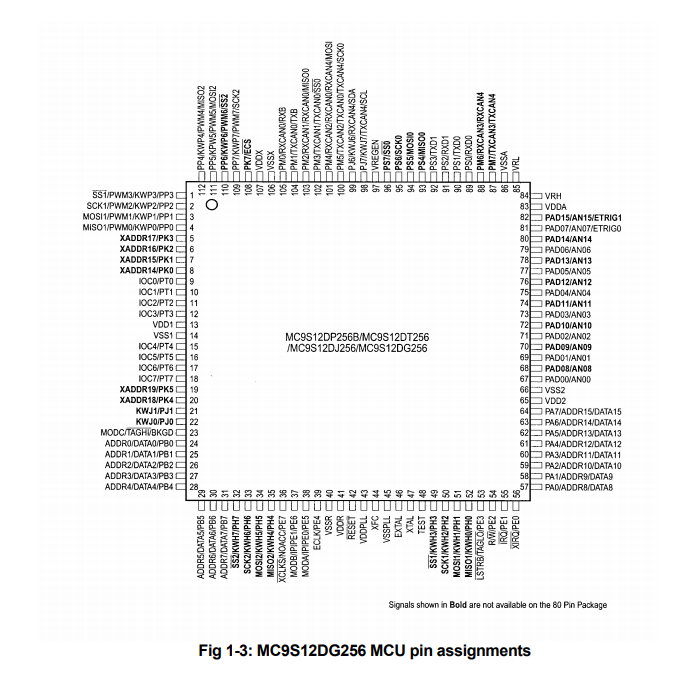
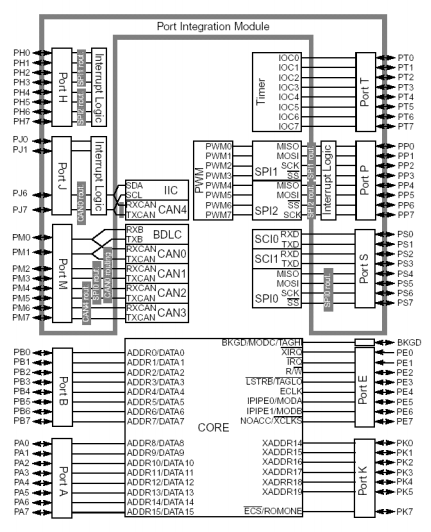


Diagrama bloc pentru PIM (Port Integration Module)

****

Porturile A, B și K pot fi utilizate ca porturi generale de intrare/ieșire (GPIO). Registrele pentru aceste porturi sunt situate în nucleul HCS12.O mare parte din portul E poate fi folosit pentru GPIO. Un număr din pinii portului E, au funcții suplimentare de control și configurare. Funcțiile legate de portul E sunt configurate în registrul central al portului E (PEAR).

Biții de control pentru porturile de intrare / ieșire din miez sunt în Registrul de Control al Controlului Pull-up(PUCR). Porturile AD0 și AD1 au funcționalități de intrare analogice și digitale. Registrele pentru aceste porturi sunt localizate în cele două module cu 8 canale analogice- digitale, ATD0 și ATD1. Porturile H, J, P, M și S suportă funcționalitatea întreruperii hardware și funcționalitatea periferică alternativă.

Microcontrolerul MC9S12DG256

Microcontrolerul MC9S12DG256este alcătuit dintr-un *procesor puternic de 16 biți* (unitate de procesare centrală), 256K octeți de *memorie flash*, 12K octeți de *memorie RAM*, 4K octeți de EEPROM și multe periferice pe chip.

Principalele caracteristici ale MC9S12DG256:

• Porturi SCI, SPI și CAN 2.0

• Interfața I2C

• Cronometre pe 8 canale pe 16 biți

• PWM de 8 canale pe 8 biți sau 4 canale pe 16 biți

• Convertor A / D cu 10 canale pe 16 canale

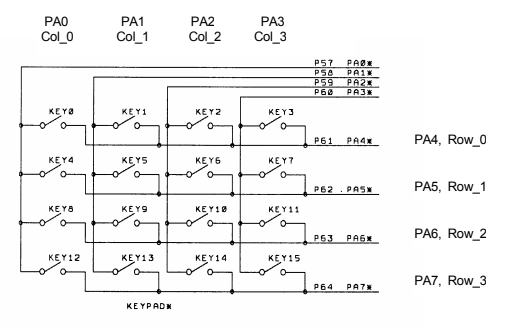
• Turație rapidă a magistralei de 25 MHz prin buclă de blocare a fazelor pe cip

• BDM pentru programarea în circuit și depanare

• Pachetul LQFP cu 112 pini oferă până la 91 intrări / ieșiri într-o amprentă redus

Tastatura cu 16 contacte

Portul A este un port bidirecțional pe 8 biți. Utilizarea sa principală este pentru tastatura 4X4. Dacă portul nu este utilizat pentru tastatură, acesta poate fi utilizat ca intrare / ieșire generală.



Citirea se face in felul urmator:

1. Stergem configuratia anterioara a coloanelor si a randurilor

2. Setam coloana

3. Daca tasta apasata se afla in coloana respectiva:

3.1. Verificam care este randul cu respectivul buton

3.2. Scriem caracterul respectiv

3.2.1. Daca caracterul respectiv este ’=’, apelam functia pt operatie

3.3. Daca nu s-a gasit niciun rand, variabila row preia valoarea 0

Afișorul LCD

Modulul afișajului cu cristale lichide (LCD) de pe dispozitivul Dragon12 este un afișaj alfanumeric pe 2 linii, cu 16 caractere. Modulul este conectat la portul K al unității microcontrolerului. Spre deosebire de afișajul cu LED-uri, modulul LCD nu necesită o manipulare specială pentru reîmprospătarea afișajului. Caracterele scrise la memoria de caractere a modulului vor rămâne afișate până când acestea sunt suprascrise, modulul este resetat sau alimentarea este oprită.

Modulul a fost proiectat pentru a comunica cu un procesor folosind un ciclu tradițional de transfer al magistralei CPU. Pentru a simplifica cerințele hardware externe, cuvântul de date pentru module poate fi restrâns la 4 biți, transferând fiecare octet în două nibble-uri (înalt și apoi scăzut). Deoarece MCU este în modul cu un singur chip, acesta nu are interfață externă a magistralei, deci sincronizarea ciclului de magistrală trebuie să fie "falsificată" cu secvențierea software-ului pentru semnalele de pe portul paralel.

Portul K este un port bidirecțional pe 8 biți. Este folosit pentru modulul de afișare LCD. Dacă portul nu este utilizat pentru afișajul LCD, acesta poate fi utilizat ca port I / O cu general. Modulul are 4 linii de date DB [7: 4] conectate la pinii portului K. Există o linie de adresă RS pentru a selecta unul dintre cele două registre interne (comandă și date). Intrarea EN este pulsată pentru a finaliza o operație de scriere a modulului. Intrarea R / W a modulului este legată la masă, plasând modulul în modul de scriere pentru totdeauna - MCU nu poate citi înapoi datele scrise în memoria de afișare. În schimb, software-ul trebuie să țină evidența caracterelor scrise în buffer-ul de afișare în variabilele dedicate.

Modulul LCD este conectat la portul K după cum urmează:

Pin 1 GND

Pin 2 VCC (5V)

Pin 3 Prin intermediul unui rezistor de 220 Ohm la GND

Pin 4 PK0 pin RS pentru modulul LCD

Pin 5 PK7 pin R / W pentru modulul LCD

Pin 6 PK1 pin EN pentru modulul LCD

Pin 7 Nu este utilizat

Pin 8 Nu este utilizat

Pin 9 Nu este utilizat

Pin 10 Mot folosit

Pin 11 PK2 pin DB4 pentru modulul LCD

Pin 12 PK3 pin DB5 pentru modulul LCD

Pin 13 PK4 pin DB6 pentru modulul LCD

Pin 14 PK5 pin DB7 pentru modulul LCD

Pin 15 Printr-un rezistor de 22 Ohmi la VCC lumina de fundal LED pentru LCD

Pin 16 GND

Software-ul muta biții de date pentru a se alinia cu pinul corespunzator al portului. Biții care reprezintă semnalele de control EN și RS sunt îmbinați cu biții de date, apoi scriși în registrul de date al portului.

Rutina de programare

//Displaying "HELLO" on LCD for Dragon12+ Trainer Board

//with HCS12 Serial Monitor Program installed. This code is for CodeWarrior IDE

//Modified from Mazidi's book with contribution from Travis Chandler

//On Dragon12+ LCD data pins of D7-D4 are connected to Pk5-Pk2, En=Pk1,and RS=Pk0

#include <hidef.h> /\* common defines and macros \*/

#include "mc9s12dg256.h" /\* derivative-specific definitions \*/

#include<string.h>

#include<stdlib.h>

#define LCD\_DATA PORTK

#define LCD\_CTRL PORTK

#define RS 0x01

#define EN 0x02

//#define KEYPAD PORTA folosim direct PORTA la verificari

void COMWRT4(unsigned char);

void DATWRT4(unsigned char);

void MSDelay(unsigned int);

void CITIRE(); // functie pentru scanarea tastaturii si afisare operator/operand specific

void CALCUL();

void DIVBYZERO();

unsigned int nr1, nr2, nr3;

/\*

codificare operatii:

1 -> adunare

2 -> scadere

3 -> inmultire

4 -> impartire

\*/

unsigned int operatie,operatie2;

unsigned int contor;

unsigned char rez[8];

void main(void)

{

DDRK = 0xFF;

DDRA = 0x0F; //randuri - intrare; coloane - iesiri

COMWRT4(0x33); //reset sequence provided by data sheet

MSDelay(1);

COMWRT4(0x32); //reset sequence provided by data sheet

MSDelay(1);

COMWRT4(0x28); //Function set to four bit data length

//2 line, 5 x 7 dot format

MSDelay(1);

COMWRT4(0x06); //entry mode set, increment, no shift

MSDelay(1);

COMWRT4(0x0E); //Display set, disp on, cursor on, blink off

MSDelay(1);

COMWRT4(0x01); //Clear display

MSDelay(1);

COMWRT4(0x80); //set start posistion, home position

MSDelay(1);

// Se face in functia CITIRE() o bucla infinita

CITIRE();

}

void COMWRT4(unsigned char command)

{

unsigned char x;

x = (command & 0xF0) >> 2; //shift high nibble to center of byte for Pk5-Pk2

LCD\_DATA =LCD\_DATA & ~0x3C; //clear bits Pk5-Pk2

LCD\_DATA = LCD\_DATA | x; //sends high nibble to PORTK

MSDelay(1);

LCD\_CTRL = LCD\_CTRL & ~RS; //set RS to command (RS=0)

MSDelay(1);

LCD\_CTRL = LCD\_CTRL | EN; //rais enable

MSDelay(5);

LCD\_CTRL = LCD\_CTRL & ~EN; //Drop enable to capture command

MSDelay(15); //wait

x = (command & 0x0F)<< 2; // shift low nibble to center of byte for Pk5-Pk2

LCD\_DATA =LCD\_DATA & ~0x3C; //clear bits Pk5-Pk2

LCD\_DATA =LCD\_DATA | x; //send low nibble to PORTK

LCD\_CTRL = LCD\_CTRL | EN; //rais enable

MSDelay(5);

LCD\_CTRL = LCD\_CTRL & ~EN; //drop enable to capture command

MSDelay(15);

}

void DATWRT4(unsigned char data)

{

unsigned char x;

x = (data & 0xF0) >> 2;

LCD\_DATA =LCD\_DATA & ~0x3C;

LCD\_DATA = LCD\_DATA | x;

MSDelay(1);

LCD\_CTRL = LCD\_CTRL | RS;

MSDelay(1);

LCD\_CTRL = LCD\_CTRL | EN;

MSDelay(1);

LCD\_CTRL = LCD\_CTRL & ~EN;

MSDelay(5);

x = (data & 0x0F)<< 2;

LCD\_DATA =LCD\_DATA & ~0x3C;

LCD\_DATA = LCD\_DATA | x;

LCD\_CTRL = LCD\_CTRL | EN;

MSDelay(1);

LCD\_CTRL = LCD\_CTRL & ~EN;

MSDelay(15);

}

void DIVBYZERO() { //functie pt impartirea la 0, se pune mesajul sub forma de caractere in vector, dupa care sa parcurge intr-un for

unsigned char mess[9];

int i=0;

mess[8]='\0';

mess[0]='D';

mess[1]='I';

mess[2]='V';

mess[3]=' ';

mess[4]='B';

mess[5]='Y';

mess[6]=' ';

mess[7]='0';

for(i=0;i<8;i++) {

DATWRT4(mess[i]);

MSDelay(50);

}

exit(0);

}

void Transform(unsigned int rezultat){ //functie pt transformarea rezultatului in forma vectoriala pentru a putea fi afisat

int c=0;

int aux=rezultat;

while(aux!=0) { //contorizam cate cifre are

aux=aux/10;

c++;

}

contor=c;

rez[c]='\0';

c--;

while(rezultat!=0) { //scriem vectorul corespunzator rezultatului de la sfarsit spre inceput

rez[c]=rezultat%10+'0';

rezultat=rezultat/10;

MSDelay(10);

c--;

}

}

void CALCUL(){ //functie pentru calcularea rezultatului

unsigned int rezultat;

int i;

unsigned int piv,pdv,rest;

if(operatie2>0) //daca avem mai mult de un operator

{

if(operatie==1 && operatie2==1) //adunare, adunare

{

rezultat = nr1+nr2+nr3;

Transform(rezultat);

}

if(operatie==1 && operatie2==2) //adunare, scadere

{

if(nr3>nr1+nr2)

{

rezultat = nr3 - nr1 - nr2;

MSDelay(1);

DATWRT4('-');

MSDelay(1);

Transform(rezultat);

}

else {

rezultat = nr1+nr2-nr3;

Transform(rezultat);

}

}

if(operatie==1 && operatie2==3) //adunare, inmultire

{

rezultat = nr1+nr2\*nr3;

Transform(rezultat);

}

if(operatie==1 && operatie2==4) //adunare, impartire

{

if(nr3 == 0){ //daca al doilea operand e 0, se apeleaza functia pt exceptie

//COMWRT4(0x80);

DIVBYZERO();

}

if(nr2%nr3!=0) //daca numerele nu se impart exact, se calculeaza pe rand partea dinainte de virgula si cea de dupa virgula si se afiseaza despartite de virgula. Lucram cu 2 zecimale

{

piv=nr2/nr3; //piv - partea inainte de virgula, pdv - partea dupa virgula

rest=nr2%nr3;

rest=rest\*100;

pdv=rest/nr3;

piv=piv+nr1;

Transform(piv);

for(i=0;i<contor;i++)

{

DATWRT4(rez[i]);

MSDelay(50);

}

MSDelay(1);

DATWRT4(',');

MSDelay(1);

Transform(pdv);

for(i=0;i<contor;i++)

{

DATWRT4(rez[i]);

MSDelay(50);

}

exit(0);

}

rezultat = nr1+nr2/nr3;

Transform(rezultat);

}

if(operatie==2 && operatie2==1) // scadere, adunare

{

if(nr2>nr1+nr3) // pt cazul in care scaderea da rezultat negativ, inversam modul de calculare pt a obtine un rezultat pozitiv si punem semnul '-' in fata rezultatului la afisare

{

rezultat = nr2 - nr3 - nr1;

MSDelay(1);

DATWRT4('-');

MSDelay(1);

Transform(rezultat);

}

else {

rezultat = nr1-nr2+nr3;

Transform(rezultat);

}

}

if(operatie==2 && operatie2==2) //scadere, scadere

{

if(nr2+nr3>nr1)

{

rezultat = nr2 + nr3 - nr1;

MSDelay(1);

DATWRT4('-');

MSDelay(1);

Transform(rezultat);

}

else {

rezultat = nr1-nr2-nr3;

Transform(rezultat);

}

}

if(operatie==2 && operatie2==3) //scadere, inmultire

{

if(nr2\*nr3>nr1)

{

rezultat = nr2 \* nr3 - nr1;

MSDelay(1);

DATWRT4('-');

MSDelay(1);

Transform(rezultat);

}

else {

rezultat = nr1-nr2\*nr3;

Transform(rezultat);

}

}

if(operatie==2 && operatie2==4) //scadere, impartire

{

if(nr3 == 0){

//COMWRT4(0x80);

DIVBYZERO();

}

if(nr2%nr3!=0)

{

piv=nr2/nr3;

rest=nr2%nr3;

rest=rest\*100;

pdv=rest/nr3;

piv=nr1-piv;

if(piv<0)

{

piv=piv\*(-1);

MSDelay(1);

DATWRT4('-');

MSDelay(1);

}

Transform(piv);

for(i=0;i<contor;i++)

{

DATWRT4(rez[i]);

MSDelay(50);

}

MSDelay(1);

DATWRT4(',');

MSDelay(1);

Transform(pdv);

for(i=0;i<contor;i++)

{

DATWRT4(rez[i]);

MSDelay(50);

}

exit(0);

}

rezultat = nr1-nr2/nr3;

Transform(rezultat);

}

if(operatie==3 && operatie2==1) // inmultire, adunare

{

rezultat = nr1\*nr2+nr3;

Transform(rezultat);

}

if(operatie==3 && operatie2==2) //inmultire, scadere

{

if(nr1\*nr2<nr3)

{

rezultat = nr3 - nr1\*nr2;

MSDelay(1);

DATWRT4('-');

MSDelay(1);

Transform(rezultat);

}

else {

rezultat = nr1\*nr2-nr3;

Transform(rezultat);

}

}

if(operatie==3 && operatie2==3) //inmultire, inmultire

{

rezultat = nr1\*nr2\*nr3;

Transform(rezultat);

}

if(operatie==3 && operatie2==4) //inmultire, impartire

{

if(nr3 == 0){

DIVBYZERO();

}

if(nr2%nr3!=0)

{

nr2=nr1\*nr2;

piv=nr2/nr3;

rest=nr2%nr3;

rest=rest\*100;

pdv=rest/nr3;

Transform(piv);

for(i=0;i<contor;i++)

{

DATWRT4(rez[i]);

MSDelay(50);

}

MSDelay(1);

DATWRT4(',');

MSDelay(1);

Transform(pdv);

for(i=0;i<contor;i++)

{

DATWRT4(rez[i]);

MSDelay(50);

}

exit(0);

}

rezultat = nr1\*nr2/nr3;

Transform(rezultat);

}

if(operatie==4 && operatie2==1) // impartire, adunare

{

if(nr3 == 0){

//COMWRT4(0x80);

DIVBYZERO();

}

if(nr1%nr2!=0)

{

piv=nr1/nr2;

rest=nr1%nr2;

rest=rest\*100;

pdv=rest/nr2;

piv=piv+nr3;

Transform(piv);

for(i=0;i<contor;i++)

{

DATWRT4(rez[i]);

MSDelay(50);

}

MSDelay(1);

DATWRT4(',');

MSDelay(1);

Transform(pdv);

for(i=0;i<contor;i++)

{

DATWRT4(rez[i]);

MSDelay(50);

}

exit(0);

}

rezultat = nr1/nr2+nr3;

Transform(rezultat);

}

if(operatie==4 && operatie2==2) // impartire, scadere

{

if(nr3 == 0){

//COMWRT4(0x80);

DIVBYZERO();

}

if(nr1%nr2!=0)

{

piv=nr1/nr2;

rest=nr1%nr2;

rest=rest\*100;

pdv=rest/nr2;

piv=piv-nr3;

if(piv<0)

{

piv=piv\*(-1);

MSDelay(1);

DATWRT4('-');

MSDelay(1);

}

Transform(piv);

for(i=0;i<contor;i++)

{

DATWRT4(rez[i]);

MSDelay(50);

}

MSDelay(1);

DATWRT4(',');

MSDelay(1);

Transform(pdv);

for(i=0;i<contor;i++)

{

DATWRT4(rez[i]);

MSDelay(50);

}

exit(0);

}

rezultat = nr1/nr2-nr3;

Transform(rezultat);

}

if(operatie==4 && operatie2==3) //impartire, inmultire

{

if(nr3 == 0){

//COMWRT4(0x80);

DIVBYZERO();

}

if(nr1%nr2!=0)

{

piv=nr1\*nr3;

piv=piv/nr2;

rest=nr1%nr2;

rest=rest\*100;

pdv=rest/nr2;

Transform(piv);

for(i=0;i<contor;i++)

{

DATWRT4(rez[i]);

MSDelay(50);

}

MSDelay(1);

DATWRT4(',');

MSDelay(1);

Transform(pdv);

for(i=0;i<contor;i++)

{

DATWRT4(rez[i]);

MSDelay(50);

}

exit(0);

}

rezultat = nr1/nr2\*nr3;

Transform(rezultat);

}

if(operatie==4 && operatie2==4) //impartire, impartire

{

rezultat = nr1/nr2/nr3;

Transform(rezultat);

}

}

else { //daca avem doar un operator

if(operatie == 1 && operatie2==0){ //adunare

rezultat = nr1 + nr2;

Transform(rezultat);

}

else if(operatie == 2 && operatie2==0){ //scadere

if(nr2>nr1)

{

rezultat = nr2 - nr1;

MSDelay(1);

DATWRT4('-');

MSDelay(1);

Transform(rezultat);

}

else

{

rezultat = nr1 - nr2;

Transform(rezultat);

}

}

else if(operatie == 3 && operatie2==0){ //inmultire

rezultat = nr1 \* nr2;

Transform(rezultat);

}

else if(operatie == 4 && operatie2==0){ //impartire

if(nr2 == 0){

//COMWRT4(0x80);

DIVBYZERO();

}

if(nr1%nr2!=0)

{

piv=nr1/nr2;

rest=nr1%nr2;

rest=rest\*100;

pdv=rest/nr2;

Transform(piv);

for(i=0;i<contor;i++)

{

DATWRT4(rez[i]);

MSDelay(50);

}

MSDelay(1);

DATWRT4(',');

MSDelay(1);

Transform(pdv);

for(i=0;i<contor;i++)

{

DATWRT4(rez[i]);

MSDelay(50);

}

exit(0);

}

rezultat = nr1 / nr2;

Transform(rezultat);

}

}

for(i=0;i<contor;i++) {

DATWRT4(rez[i]);

MSDelay(50);

}

exit(0);

}

void CITIRE() { // functie pt citire de la tastatura

unsigned char row;

unsigned int contorOperanzi = 1;

while(1){

//facem mai intai deubounce-ul pentru a nu avea probleme la citire normala

do{

PORTA = PORTA | 0x0F; //setarea coloanelor

row = PORTA & 0xF0; //citirea randurilor

}while(!(row | 0x00)); // asteptam apasarea unei taste

do{

do{

MSDelay(1);

row = PORTA & 0xF0; //citire rand

}while(!(row | 0x00)); // verificam daca o tasta a fost apasata

MSDelay(50); //asteptare pentru debounce

row = PORTA & 0xF0;

}while(!(row | 0x00));

//sfarsitul partii de initializare si debounce

//citirea normala

while(1){

PORTA = PORTA & 0xF0; //sterge configuratia anterioara a coloanelor

PORTA = PORTA | 0x01; //setam coloana 0

MSDelay(1);

row = PORTA & 0xF0;

if(row | 0x00){ //tasta apasata se afla pe coloana 0

if(row & 0x10){ //pentru primele 3 coloane, verificam care tasta este apasata si in functie de cati operanzi avem pana la acel moment, punem respectiva cifra la finalul aceluia din urma

DATWRT4('1');

if(contorOperanzi == 1)

nr1 = nr1 \* 10 + 1;

else if(contorOperanzi == 2)

nr2 = nr2 \* 10 + 1;

else if(contorOperanzi > 2)

nr3 = nr3 \* 10 + 1;

}

else if(row & 0x20){

DATWRT4('4');

if(contorOperanzi == 1)

nr1 = nr1 \* 10 + 4;

else if(contorOperanzi == 2)

nr2 = nr2 \* 10 + 4;

else if(contorOperanzi > 2)

nr3 = nr3 \* 10 + 4;

}

else if(row & 0x40){

DATWRT4('7');

if(contorOperanzi == 1)

nr1 = nr1 \* 10 + 7;

else if(contorOperanzi == 2)

nr2 = nr2 \* 10 + 7;

else if(contorOperanzi > 2)

nr3 = nr3 \* 10 + 7;

}

else if(row & 0x80){ // daca intalnim semnul '=' se apeleaza functia CALCUL()

DATWRT4('=');

CALCUL();

}

break; //iesire din bucla infinita

}

PORTA = PORTA & 0xF0; //sterge configuratia anterioara a coloanelor

PORTA = PORTA | 0x02; //setam coloana 1

MSDelay(1);

row = PORTA & 0xF0;

if(row | 0x00){ //tasta apasata se afla pe coloana 1

if(row & 0x10){

DATWRT4('2');

if(contorOperanzi == 1)

nr1 = nr1 \* 10 + 2;

if(contorOperanzi == 2)

nr2 = nr2 \* 10 + 2;

if(contorOperanzi > 2)

nr3 = nr3 \* 10 + 2;

}

else if(row & 0x20){

DATWRT4('5');

if(contorOperanzi == 1)

nr1 = nr1 \* 10 + 5;

if(contorOperanzi == 2)

nr2 = nr2 \* 10 + 5;

if(contorOperanzi > 2)

nr3 = nr3 \* 10 + 5;

}

else if(row & 0x40){

DATWRT4('8');

if(contorOperanzi == 1)

nr1 = nr1 \* 10 + 8;

if(contorOperanzi == 2)

nr2 = nr2 \* 10 + 8;

if(contorOperanzi > 2)

nr3 = nr3 \* 10 + 8;

}

else if(row & 0x80){

DATWRT4('0');

if(contorOperanzi == 1)

nr1 = nr1 \* 10 + 0;

if(contorOperanzi == 2)

nr2 = nr2 \* 10 + 0;

if(contorOperanzi > 2)

nr3 = nr3 \* 10 + 0;

}

break; //iesire din bucla infinita

}

PORTA = PORTA & 0xF0; //sterge configuratia anterioara a coloanelor

PORTA = PORTA | 0x04; //setam coloana 2

MSDelay(1);

row = PORTA & 0xF0;

if(row | 0x00){ //tasta apasata se afla pe coloana 2

if(row & 0x10){

DATWRT4('3');

if(contorOperanzi == 1)

nr1 = nr1 \* 10 + 3;

if(contorOperanzi == 2)

nr2 = nr2 \* 10 + 3;

if(contorOperanzi > 2)

nr3 = nr3 \* 10 + 3;

}

else if(row & 0x20){

DATWRT4('6');

if(contorOperanzi == 1)

nr1 = nr1 \* 10 + 6;

if(contorOperanzi == 2)

nr2 = nr2 \* 10 + 6;

if(contorOperanzi > 2)

nr3 = nr3 \* 10 + 6;

}

else if(row & 0x40){

DATWRT4('9');

if(contorOperanzi == 1)

nr1 = nr1 \* 10 + 9;

if(contorOperanzi == 2)

nr2 = nr2 \* 10 + 9;

if(contorOperanzi > 2)

nr3 = nr3 \* 10 + 9;

}

break; //iesire din bucla infinita

}

PORTA = PORTA & 0xF0; //sterge configuratia anterioara a coloanelor

PORTA = PORTA | 0x08; //setam coloana 3

MSDelay(1);

row = PORTA & 0xF0;

if(row | 0x00){ //tasta apasata se afla pe coloana 3

if(row & 0x10){ //daca gasim apasata o tasta pt o operatie crestem contorul operanzilor

DATWRT4('+'); //verificam faptul ca operatia este prima sau a doua din ecuatie si crestem numarul operanzilor

if(operatie>0)

{

operatie2=1;

contorOperanzi++;

}

else

{

operatie = 1;

contorOperanzi++;

}

}

else if(row & 0x20){

DATWRT4('-');

if(operatie>0)

{

operatie2=2;

contorOperanzi++;

}

else

{

operatie = 2;

contorOperanzi++;

}

}

else if(row & 0x40){

DATWRT4('\*');

if(operatie>0)

{

operatie2=3;

contorOperanzi++;

}

else

{

operatie = 3;

contorOperanzi++;

}

}

else if(row & 0x80){

DATWRT4('/');

if(operatie>0)

{

operatie2=4;

contorOperanzi++;

}

else

{

operatie = 4;

contorOperanzi++;

}

}

break; //iesire din bucla infinita

}

row = 0; //tasta negasita

break; //iesire din bucla

}

}

}

void MSDelay(unsigned int itime) // functie de delay

{

unsigned int i; unsigned int j;

for(i=0;i<itime;i++)

for(j=0;j<4000;j++);

}