Calculator pentru implementarea unor operatii aritmetice de baza

Greab Ioan-Victor

Gumbinger Eduard

Grupa 3.1 CTI III RO

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

**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,

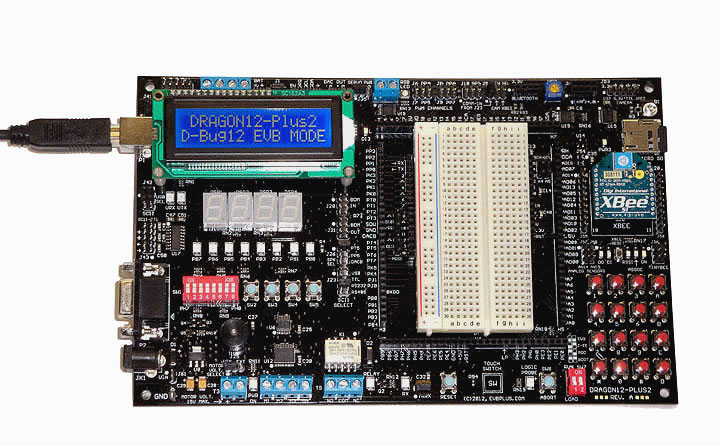
scădere, înmulţire, împărţire);

* 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 afișa un mesaj de eroare în cazul împărţirii la 0.

1. **Descrierea placii de dezvoltare Dragon12 Plus2**



Placa DragonS12-Plus2 este o placa

de buget cu multe caracteristici pentru

familia de microcontrolere HCS12.

Pentru ingineri este un sistem prototip

numai bun pentru a crea aplicatii pentru arhitectura HCS12. Pentru studenti este

utilizata ca o platforma pentru proiecte.

Microcontrolerul folosit este:

**MC9S12DG256.**

Kitul Dragon12-Plus2 vine cu urmatoarele facilitati:

* Placa in sine
* Software specializat pentru dezvoltare ( AsmIDE )
* Cablu USB de tip B
* Alimentare la 9V, 500 mA

Pe aceasta placa regasim urmatoarele module:

1. RGB Led
2. Port de expansiune I2C pentru devicuri externe
3. Port CAN
4. Display-uri cu 7 segemente in numar de 4
5. 8 leduri
6. 4 butoane
7. O miniboxa
8. Detector IR
9. Minitastatura 4x4
10. Display 16x2
11. Bridge-uri de tip H in numar de 2
12. Senzor de temperatura
13. Senzor de lumina
14. Potentiometru
15. **Microcontrolerul MC9S12DG256**

Placa de dezvoltare Dragon12-Plus2 vine cu microcontrolerul MC9S12DG256 atasat. Acest microcontroler are un CPU pe 16 biti, o memorie flash de 256K octeti, RAM de 12K octeti, EEPROM de 4K octeti si multe periferice.

Alte module incluse:

* SCI ports
* SPI ports
* CAN 2.0 ports
* I2C interface
* 8-ch 16-bit timers
* 8-ch 8-bit or 4-ch 16 bit PWM
* 16-channel 10-bit A/D converter
* Fast 25 MHz bus speed via on-chip Phase Lock Loop
* BDM for in-circuit programming and debugging
* 112-pin LQFP package offers up to 91 I/O in a small footprint

Schema Bloc:



1. **Modulele Calculatorului Numeric**

Module utilizate in realizarea calculatorului numeric pentru operatii aritmetice sunte:

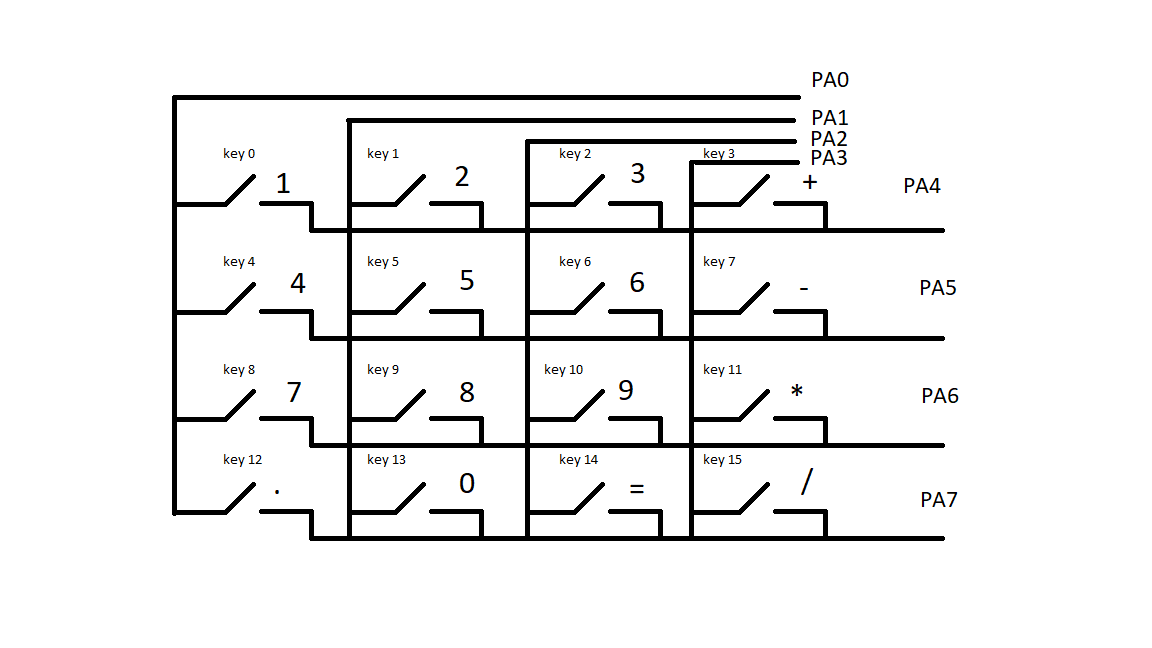
* + Minitastatura
  + LCD

**Minitastatura**

Pentru citirea inputurilor am utilizat o tastatura 4x4. Portul A este un port bidirectional si este folosit in cadrul acestui proiect pentru citirea tastaturii.

Citirea ministastaturii se face pe coloane acest lucru fiind posibil setand ca output rangurile PA0-PA3 ale portului A. Cand una dintre coloane este activata, iar restul sunt dezactivate se asteapta input pe coalana de la rangurile

PA4 – PA7 (ex. PA0 = 1 si PA1-PA3=0 =>asteptam input pentru coloana 1 ).

Schema pentru mini-tastatura:

Rutina de scanare:

Scanare coloana 1: PA3 PA2 PA1 PA0 = 0 0 0 1

Daca PA4=1 => Key 0 este apasa => ch 1

Daca PA5=1 => Key 0 este apasa => ch 2

Daca PA6=1 => Key 0 este apasa => ch 7

Daca PA7=1 => Key 0 este apasa => ch .

Scanare coloana 2: PA3 PA2 PA1 PA0 = 0 0 1 0

Daca PA4=1 => Key 0 este apasa => ch 2

Daca PA5=1 => Key 0 este apasa => ch 5

Daca PA6=1 => Key 0 este apasa => ch 8

Daca PA7=1 => Key 0 este apasa => ch 0

Scanare coloana 3: PA3 PA2 PA1 PA0 = 0 1 0 0

Daca PA4=1 => Key 0 este apasa => ch 3

Daca PA5=1 => Key 0 este apasa => ch 6

Daca PA6=1 => Key 0 este apasa => ch 9

Daca PA7=1 => Key 0 este apasa => ch =

Scanare coloana 4: PA3 PA2 PA1 PA0 = 1 0 0 0

Daca PA4=1 => Key 0 este apasa => ch +

Daca PA5=1 => Key 0 este apasa => ch -

Daca PA6=1 => Key 0 este apasa => ch \*

Daca PA7=1 => Key 0 este apasa => ch /

**LCD-ul**

Pentru afisarea pe LCD vom folosi portul K , care este un port biriectional pe 8 biti.

LCD-ul prezent pe placa este unul de tipul 1602 , adica are 16 casute pentru caractere ( si de asemenea 16 pini ) si 2 randuri.

Pe aceasta placa modul LCD este folosit doar pentru scrierea caracterelor obtinute din cod , de pe acesta nu se pot citi caracterele decat daca se fac niste legaturi speciale.

**Modul de reprezentare a caracterelor pe LCD**

* Fiecare caracter este reprezentat sub forma unei matrici cu 5 linii si 7 coloane .
* In interiorul modulului LCD se afla un controller Hitachi HD44780 care converteste caracterul primit in cod , din cod ASCII in matricea 5x7.
* Controllerul este legat la Portul K care foloseste bitii de mai jos pentru a trimite comenzi :

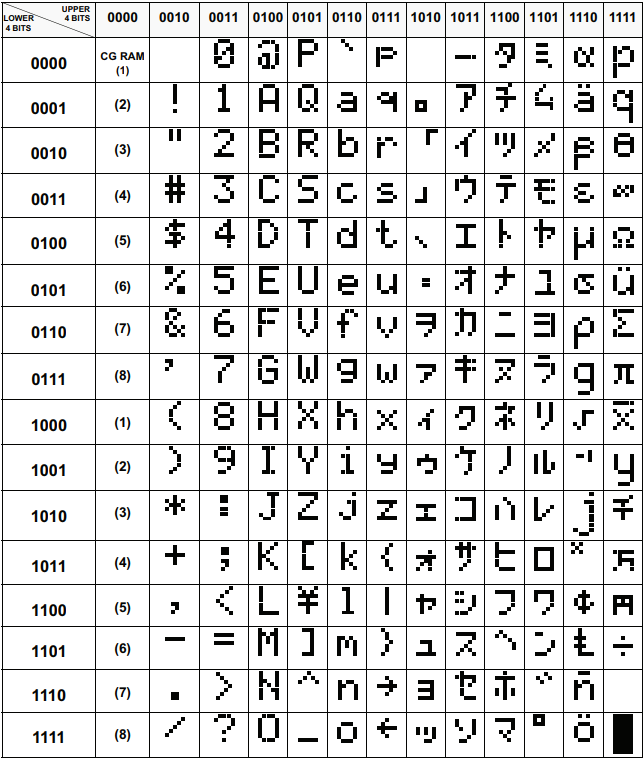
1. Bitul 0 (PK0) este folosit pentru comenzi(1) sau date(0);
2. Bitul 1 (PK1) este folosit pentru transferul de date;
3. Bitul 5 (PK5-2) contine datele;
4. Bitul 7(PK7) este folosit pentru citire/scriere.In cazul in care nu se fac legaturile speciale pentru legarea LCD-ului , pe display se pot doar scrie date.

* Comenzi ce se pot trimite display-ului:

1. Clear display and return cursor to home (upper left)
2. Cursor home (don’t clear display)
3. Entry mode (move cursor left or write)
4. Display on/off – turns display on or off, cursor on or off, cursor blink
5. Cursor/display shift – move cursor or shift display, which direction,etc

Pentru ca sa afisam pe display anumite caractere , vom folosi 8 biti , care sunt impartiti in doua grupe de cate 4 , si in functie de combinatia aleasa , pe display apare un caracter.

**Tabel pentru crearea caracterelor :**



**Codul aferent calculatorului**

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

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

#include <string.h>

#include <stdlib.h>

#include <ctype.h>

#include <stdlib.h>

#include <math.h>

#define LCD\_DATA PORTK

#define LCD\_CTRL PORTK

#define RS 0x01

#define EN 0x02

#define MAX 100

void COMWRT4(unsigned char);

void DATWRT4(unsigned char);

void MSDelay(unsigned int);

/\* Functii principale\*/

void afisString(char\* p);

void evaluateExpr(char\* p);

void afisIntreg(int number);

char\* p;

char expr[MAX]; //buffer in care retinem expresia aritmetica

int contor;

void main(void)

{

char\* p=expr;

DDRK = 0xFF; //setare port K ca si output

DDRA = 0x0F; // testam liniile(butoanele) P4,P5,P6,P7 ->inputs

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);

/\* CITIRE TASTATURA \*/

for(;;){

PORTA=0; //dezactivare coloane

//COLOANA 4

PORTA=PORTA|0x08;

if(PORTA == 0x18) { //testare linie 1 coloana 4

while((PORTA & (1<<4))!=0); //debouncing

afisString("+");

MSDelay(200);

strcat(p,"+");

//MSDelay(1);

}

if(PORTA == 0x28) { //testare linie 2 coloana 4

while((PORTA & (1<<5))!=0);

afisString("-");

strcat(p,"-");

// MSDelay(1);

}

if(PORTA == 0x48) { //testare linie 3 coloana 4

while((PORTA & (1<<6))!=0);

afisString("\*");

strcat(p,"\*");

//SDelay(1);

}

if(PORTA == 0x88) { //testare linie 4 coloana 4

while((PORTA & (1<<7))!=0);

afisString("/");

strcat(p,"/");

//Delay(1);

}

PORTA=0; //dezactivare coloane

// COLOANA 3

PORTA=PORTA|0x04;

if(PORTA == 0x14) { //testare linie 1 coloana 3

while((PORTA & (1<<4))!=0);

afisString("3");

MSDelay(200);

strcat(p,"3");

//MSDelay(1);

}

if(PORTA == 0x24) { //testare linie 2 coloana 3

while((PORTA & (1<<5))!=0);

afisString("6");

strcat(p,"6");

//MSDelay(1);

}

if(PORTA == 0x44) { //testare linie 3 coloana 3

while((PORTA & (1<<6))!=0);

afisString("9");

strcat(p,"9");

//MSDelay(1);

}

if(PORTA == 0x84) { //testare linie 4 coloana 3

while((PORTA & (1<<7))!=0);

;

//MSDelay(1);

//afisString(p);

afisString("=");

evaluateExpr(p);

}

PORTA=0; //dezactivare coloane

// COLOANA 2

PORTA=PORTA|0x02;

if(PORTA == 0x12) { //testare linie 1 coloana 2

while((PORTA & (1<<4))!=0);

afisString("2");

MSDelay(200);

strcat(p,"2");

//MSDelay(1);

}

if(PORTA == 0x22) { //testare linie 2 coloana 2

while( (PORTA & (1<<5))!=0);

afisString("5");

strcat(p,"5");

//MSDelay(1);

}

if(PORTA == 0x42) { //testare linie 3 coloana 2

while( (PORTA & (1<<6))!=0);

afisString("8");

strcat(p,"8");

// MSDelay(1);

}

if(PORTA == 0x82) { //testare linie 4 coloana 2

while( (PORTA & (1<<7))!=0);

afisString("0");

strcat(p,"0");

//MSDelay(1);

}

PORTA=0;//dezactivare coloane

// COLOANA 1

PORTA=PORTA|0x01;

//testare buton

if(PORTA == 0x11) { //testare linie 1 coloana 1

while( (PORTA & (1<<4) )!=0);

afisString("1");

MSDelay(200);

strcat(p,"1");

//MSDelay(1);

}

if(PORTA == 0x21) { //testare linie 2 coloana 1

while( (PORTA & (1<<5) )!=0);

afisString("4");

strcat(p,"4");

// MSDelay(1);

}

if(PORTA == 0x41) { //testare linie 3 coloana 1

while( (PORTA & (1<<6) )!=0);

afisString("7");

strcat(p,"7");

// MSDelay(1);

}

if( PORTA == 0x81) { //testare linie 4 coloana 1

while( (PORTA & (1<<7) )!=0);

afisString(".");

strcat(p,".");

//MSDelay(1);

}

PORTA=0; //dezactivare coloana

}

}

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);

}

//functie afisare String pe display

void afisString(char\* string){

while(\*string!='\0'){

DATWRT4(\*string);

MSDelay(1);

string++;

}

}

void MSDelay(unsigned int itime)

{

unsigned int i; unsigned int j;

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

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

}

//functia de calcul a expresiei introduse la calculator

void evaluateExpr(char\* expr){ //expr este este expresia introdusa prin intermediul butoanelor

//maxim 3 operanzi

char operatori[2] = {0}; //vector in care retinem operatorii aritmetici +,-,/,\*

float operanzi[3]; //vector in care retinem operanzii

unsigned k = 0;

unsigned nrop = 0; //contor pentru a stii numarul de operatii care trebuie efectuat

float rez = 0; //variabila in care retinem rezultatul final

float aux = 0; //variabila intermediara pentru calcul

char\* expresie = expr; //variabila intermediara

char\* p; //folosit pentru obtinere operanzi

//variabile folosite pentru scrierea pe display a rezultatului "rez"

int intreg=0; //parte intreaga a numarului

float afterv=0; //parte fractionare a rezultatului

int virgula=0; //partea zecimala

int flag=0;

//Caz particular in care primul numar este negativ

if(\*expresie == '-'){ //daca primul caracter este cu minus setam flagul pe 1

flag=1;

strcpy(expresie,expresie+1); // eliminare

expresie[strlen(expresie)]='\0'; //setare sfarsit de sir

}

//obtinere operatori

while (\*expresie != '\0'){ //parcugem caracter cu caracter si retinem operatorii in vector

if (!isdigit(\*expresie) && \*expresie !='.')

operatori[nrop++] = \*expresie; //DETERMINAM NUMARUL DE OPERATII

++expresie;

}

p = strtok(expr, "/+-\*");

//obtinere operanzi

while (p!=NULL){ //separam fiecare operand

//operanzi[k]= strtod(p,NULL);

operanzi[k]=atof(p); //convertire din char[] la float

k++;

p = strtok(NULL, "/+-\*");

}

//flag == 1 primul numar trebuie sa devina negativ

if(flag==1){ //daca flagul este 1 atunci primul operand este negativ

operanzi[0]\*=(-1); //il inmultim cu -1

}

//testare impartire la 0

if(((operatori[0] =='/') && (operanzi[1]==0)) || ((operatori[1] =='/') && (operanzi[2]==0))){

COMWRT4(0x01); //clear

MSDelay(1);

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

MSDelay(1);

afisString("Eroare /0");

}

/\* Calcul expresie \*/

//daca avem un operator

//avem 2 operanzi

if (nrop == 1){

switch (operatori[0]){

case '+':

rez = operanzi[0] + operanzi[1];

break;

case '-':

rez = operanzi[0] - operanzi[1];

break;

case '\*':

rez = operanzi[0] \* operanzi[1];

break;

case '/':

rez = operanzi[0] / operanzi[1];

break;

default:

break;

}

//2 operatori => 3 operanzi

}else if ((operatori[1] == '\*' || operatori[1] == '/' )&& (operatori[0] == '+' || operatori[0] == '-')){

//daca al doilea operator este \* sau / si primul operator este + sau - atunci trebuie sa se respecte ordinea operatiilor

//Pentru a se respecta ordinea operatiilor in cazul mentionat mai sus se calculeaza mai intai operatia ditre operand 2 si operand 3

switch (operatori[1]){

case '\*':

aux = operanzi[1] \* operanzi[2];

break;

case '/':

aux = operanzi[1] / operanzi[2];

break;

}

switch (operatori[0]){

case '+':

rez = aux + operanzi[0];

break;

case '-':

rez = aux - operanzi[0];

break;

}

}

else{ //daca nu exista exceptie ordinea operatiilor se desfasoara in mod normal

switch (operatori[0]){

case '\*':

aux = operanzi[0] \* operanzi[1];

break;

case '/':

aux = operanzi[0] / operanzi[1];

break;

case '+':

aux = operanzi[0] + operanzi[1];

break;

case '-':

aux = operanzi[0] - operanzi[1];

break;

}

switch (operatori[1]){

case '+':

rez = aux + operanzi[2];

break;

case '-':

rez = aux - operanzi[2];

break;

case '\*':

rez = aux \* operanzi[2];

break;

case '/':

rez = aux / operanzi[2];

break;

}

}

/\* PRELUCRARE REZULTAT \*/

intreg=(int)rez; //retinem partea intreaga

if(intreg<0){ //daca numarul este negativ

DATWRT4('-'); //afisam '-'

intreg\*=(-1);

afisIntreg(intreg); //afisam partea intreaga

rez\*=(-1);

afterv=(float)(rez - intreg);//extragem partea fractionara

afterv=afterv\*100;

afterv=ceil(afterv);//aproxim primele 2 cifre din parte fractionara

if(afterv!=0){ //daca are parte fractionara

DATWRT4('.'); //afisam '.'

virgula=(int)afterv;//obtinem primele 2 cifre

afisIntreg(virgula); //afisam primele 2 cifre din partea fractionara

}

}

else{ //daca numarul este pozitiv

afisIntreg(intreg); //afisam partea intreaga

afterv=(float)(rez - intreg); //aflam partea fractionara

afterv=afterv\*100;//primele 2 cifre din partea fractionara

afterv=ceil(afterv); //rotunjim superior partea fractionara

if(afterv!=0){ //daca exista parte fractionara

DATWRT4('.'); //print '.'

virgula=(int)afterv;

afisIntreg(virgula); //afiasare primele 2 cifre din partea fractionara

}

}

}

//afisare numar intreg cifra cu cifra

void afisIntreg(int nr){

if(!nr)

return;

afisIntreg(nr/10);

DATWRT4((unsigned char)(nr%10 + '0'));

}

**Bibliografie**

* Lecture\_24.pdf
* dragon12-plus2\_hcs12\_manual.pdf
* dragon12\_sch\_c4.pdf
* 9S12DP256BDGV2-userguide.pdf