Universitatea Politehnica Timișoara

Facultatea de Automatică și Calculatoare

**Sisteme Încorporate**

**Implementarea unui calculator digital pe placa Dragon12**

Albișoru Carmen Ioana

Apetrei Cristina

Anul III, CTI

Semigrupa 1.1

Anul universitar 2017 - 2018

**Tema Proiectului:**

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);

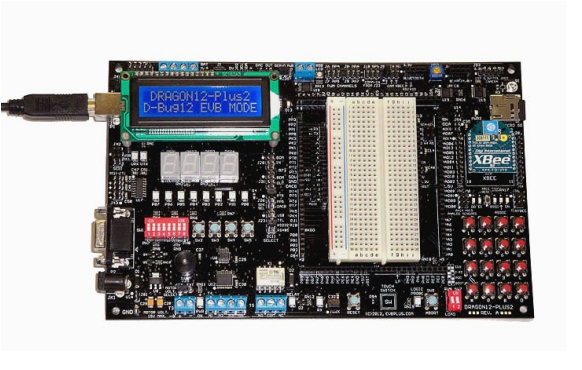
• Operanzii, operatorii şi rezultatul vor fi afişate pe LCD-ul plăcii;

• 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**

1. **Placa Dragon12-Plus2**

 Principalele caracteristici ale plăcii Dragon12-Plus2 sunt:

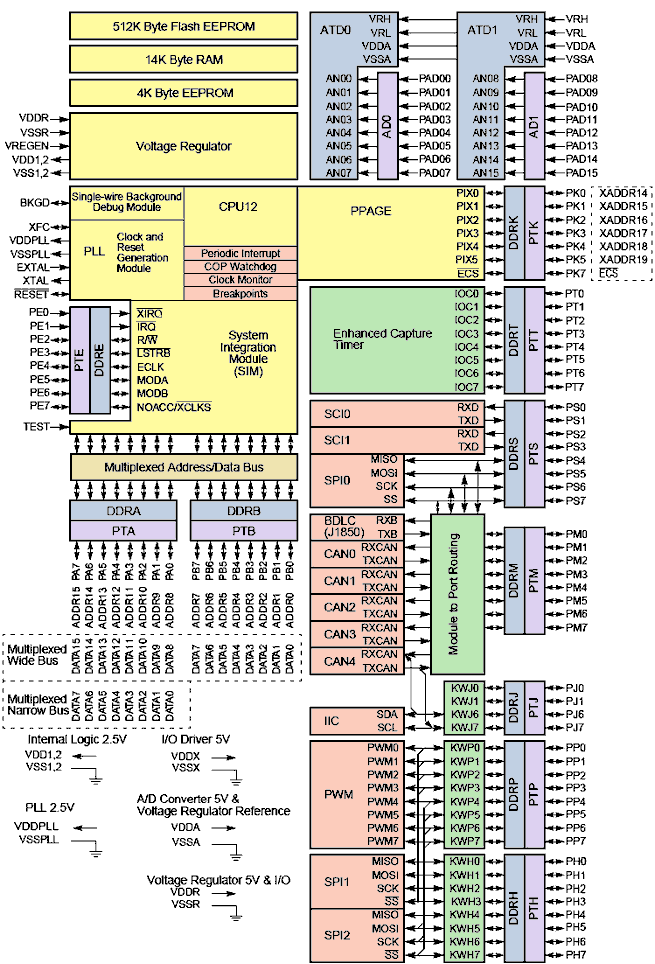
Interfață USB, port de comunicare RS485, port de extindere I2C, port CAN, port de expansiune SPI, afișaj LED cu 7 seg.,

opt LED-uri, comutator DIP cu opt poziții, 4 comutatoare cu butoane, LED de culoare RGB,

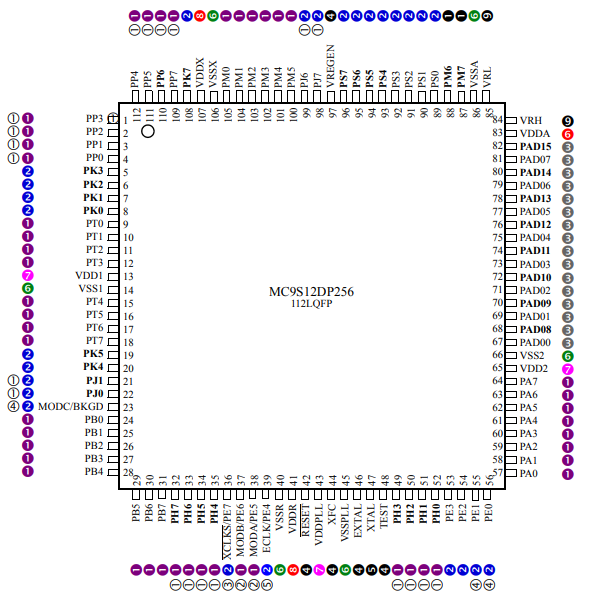
transmițător IR cu oscilator de 38 kHz la bord, sonde logice cu indicatoare LED, senzori de temperatură și de lumină, slot pentru cartele de memorie MicroSD, interfața TTL a camerei JPEG, interfețe XBee (Wi-Fi), Bluetooth, punte-H duală de înaltă eficiență pentru controlul motoarelor de curent continuu sau a unui motor pas cu pas, etc.

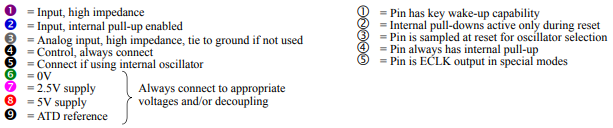
Placa Dragon12-Plus este livrată cu MC9S12DP256CCPV sau MC9S12DG256CVPE instalat. Vom folosi varianta cu DG256, care are 2 porturi CAN.

**Descrierea hardware pentru familia HCS12**

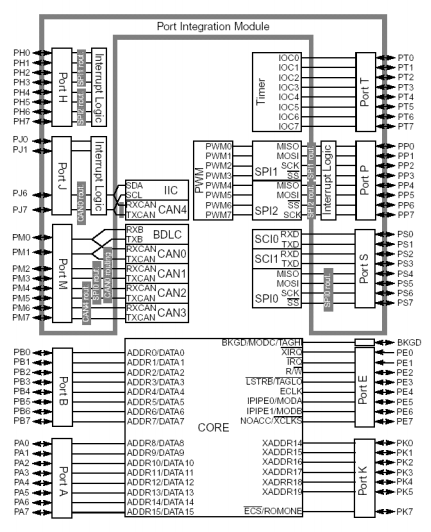


*Diagrama bloc a unității microcontrolerului*





*Pinii unității microcontrolerului (120 pini)*

****

*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.

**B.Microcontrolerul MC9S12DG256**

**Microcontrolerul MC9S12DG256** este 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

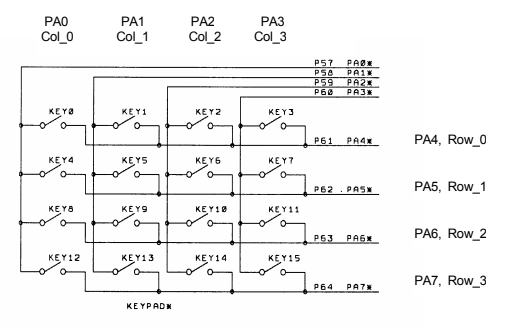
• BDM pentru programarea în circuit și depanare

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

|  |  |  |
| --- | --- | --- |
| NUME PIN | NUMARUL PINULUI | UTILIZARE I/O |
| PA0(ieșire) | 57 | Coloana0 a tastaturii |
| PA1(ieșire) | 58 | Coloana1 a tastaturii |
| PA2(ieșire) | 59 | Coloana2 a tastaturii |
| PA3(ieșire) | 60 | Coloana3 a tastaturii |
| PA4(intrare) | 61 | Rândul0 al tastaturii |
| PA5(intrare) | 62 | Rândul1 al tastaturii |
| PA6(intrare) | 63 | Rândul2 al tastaturii |
| PA7 (intrare) | 64 | Rândul3 al tastaturii |
| PK0 (ieșire) | 8 | pin RS pentru modulul LCD |
| PK1 (ieșire) | 7 | pin EN pentru modulul LCD |
| PK2 | 6 | DB4 pentru modulul LCD (bidirecțional) |
| PK3 | 5 | DB5 pentru modulul LCD (bidirecțional) |
| PK4 | 20 | DB6 pentru modulul LCD (bidirecțional) |
| PK5 | 19 | DB7 pentru modulul LCD (bidirecțional) |
| PK7 (ieșire) | 108 | pin R / W pentru modulul LCD |

***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.



***Scanarea tastaturii (CATOD COMUN) se face astfel:***

* Se sterge configuratia anterioara a randurilor si coloanelor;
* Se genereaza 1 logic pe coloana 3 (PA3 = 1) – verific tastele +, -, \*, / ;
  + Se citeste randul;
    - * Daca PA4 = 1 => s-a apasat tasta „+”;
      * Daca PA5 = 1 => s-a apasat tasta „-”;
      * Daca PA6 = 1 => s-a apasat tasta „\*”;
      * Daca PA7 = 1 => s-a apasat tasta „/”;
* Se sterge configuratia anterioara a randurilor si coloanelor;
* Se genereaza 1 logic pe coloana 2 (PA2 = 1) – verific tastele 3, 6, 9, . ;
  + Se citeste randul;
    - * Daca PA4 = 1 => s-a apasat tasta „3”;
      * Daca PA5 = 1 => s-a apasat tasta „6”;
      * Daca PA6 = 1 => s-a apasat tasta „9”;
      * Daca PA7 = 1 => s-a apasat tasta „.”;
* Se sterge configuratia anterioara a randurilor si coloanelor;
* Se genereaza 1 logic pe coloana 1 (PA1 = 1) – verific tastele 2, 5, 8, 0 ;
  + Se citeste randul;
    - * Daca PA4 = 1 => s-a apasat tasta „2”;
      * Daca PA5 = 1 => s-a apasat tasta „5”;
      * Daca PA6 = 1 => s-a apasat tasta „8”;
      * Daca PA7 = 1 => s-a apasat tasta „0”;
* Se sterge configuratia anterioara a randurilor si coloanelor;
* Se genereaza 1 logic pe coloana 0 (PA0 = 1) – verific tastele 1, 4, 7, = ;
  + Se citeste randul;
    - * Daca PA4 = 1 => s-a apasat tasta „1”;
      * Daca PA5 = 1 => s-a apasat tasta „4”;
      * Daca PA6 = 1 => s-a apasat tasta „7”;
      * Daca PA7 = 1 => s-a apasat tasta „=”;

***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.

Utilizare:

1. Comanda 0x28: Spune controller-ului ca display-ul foloseşte 4 biţi de date, 2 linii pentru afişare , font de 5x7 puncte

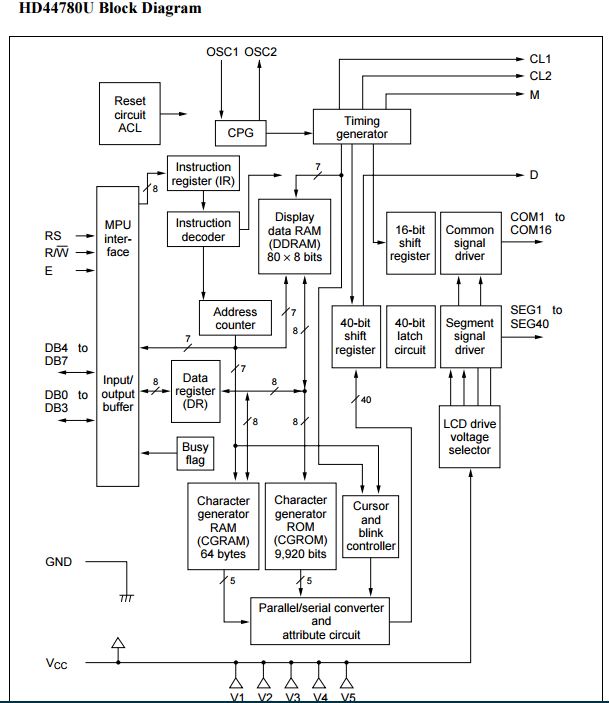
2. Comanda 0x0F: Aprinde display+ul, foloseşte cursorul, pâlpâie cursorul

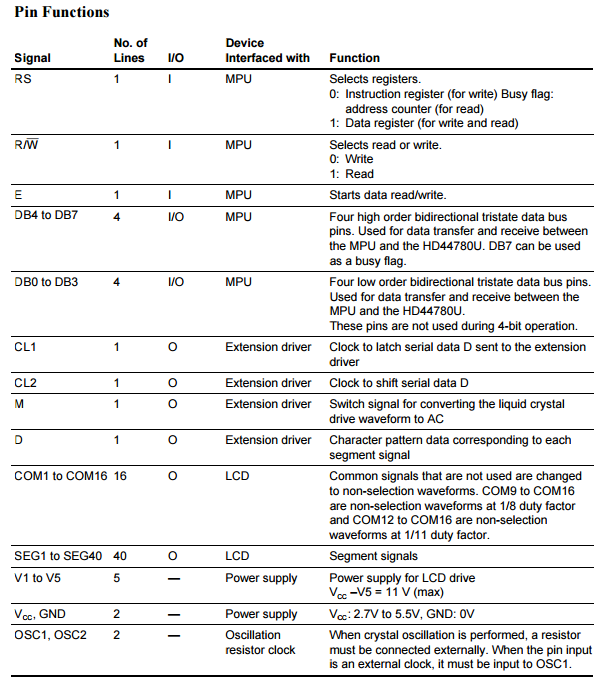
3. Comanda 0x06: Mută cursorul spre dreapta după ce scrie un caracter

4. Comanda 0x01: Şterge ecranul, mută cursorul în partea din stânga-sus

5. Aşteapta pentru cel puţin 1.64 ms

6. După ce display-ul a fost setat, se pot scrie caractere pe ecran





**Rutinele de programare**

|  |
| --- |
| #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 |
|  |  |
|  | void COMWRT4(unsigned char); |
|  | void DATWRT4(unsigned char); |
|  | void KEYPADSCAN(); // functie pentru scanarea tastaturii |
|  | //si afisare operatori/operanzi specifici |
|  | void MSDelay(unsigned int); //functie care genereaza timpul de asteptare |
|  | void OPERATIE(); // functie pentru calculul operatiilor |
|  | /\* |
|  | codificare operatii: |
|  | 0 -> adunare |
|  | 1 -> scadere |
|  | 2 -> inmultire |
|  | 3 -> impartire |
|  | 4 -> operatie nedefinita |
|  | \*/ |
|  | int operand1, operand2, operand3; |
|  | unsigned int operatie = 4, operatie2 = 4; |
|  | unsigned char rez[8]; |
|  |  |
|  |  |
|  | void main(void) |
|  | { |
|  |  |
|  | DDRK = 0xFF; // pinii portului K sunt setati ca iesiri |
|  | DDRA = 0x0F; // pinii 0-3 (coloanele tastaturii) ai portului A |
|  | //sunt setati ca iesiri |
|  | // pinii 4-7 (randurile tastaturii) ai portului A |
|  | //sunt setati ca intrari |
|  |  |
|  | 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); // cisplay set, disp on, cursor on, blink off |
|  | MSDelay(1); |
|  | COMWRT4(0x01); // clear display |
|  | MSDelay(1); |
|  | COMWRT4(0x80); // set start posistion, home position |
|  | MSDelay(1); |
|  |  |
|  | KEYPADSCAN(); // facem direct in interiorul functiei o bucla infinita |
|  | // pentru ca utilizatorul sa scrie continuu, |
|  | // pana se ajunge la "=" |
|  | } |
|  | 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 itoa(int rezultat){ |
|  | int i = 0; |
|  | if(rezultat == 0) |
|  | rez[i++] = '0'; |
|  | while(rezultat != 0){ |
|  | rez[i] = rezultat % 10 + '0'; |
|  | rezultat = rezultat / 10; |
|  | MSDelay(10); |
|  | i++; |
|  | } |
|  | rez[i] = '\0'; |
|  | } |
|  |  |
|  | void print(){ |
|  | int i; |
|  | for(i = strlen(rez) - 1; i >= 0; i--){ |
|  | DATWRT4(rez[i]); |
|  | MSDelay(50); |
|  | } |
|  | } |
|  |  |
|  | void printError(){ |
|  | COMWRT4(0xC0); |
|  | MSDelay(1); |
|  | DATWRT4('E'); |
|  | MSDelay(1); |
|  | DATWRT4('R'); |
|  | MSDelay(1); |
|  | DATWRT4('O'); |
|  | MSDelay(1); |
|  | DATWRT4('A'); |
|  | MSDelay(1); |
|  | DATWRT4('R'); |
|  | MSDelay(1); |
|  | DATWRT4('E'); |
|  | MSDelay(1); |
|  | } |
|  |  |
|  | void OPERATIE(){ |
|  | int rezultat, parteR = 0; |
|  | int i, parteReala; |
|  | float r = 0.0; |
|  |  |
|  | if(operatie2 == 2 && (operatie != 2 && operatie != 3)){ |
|  |  |
|  | operand2 = operand2 \* operand3; |
|  | operatie2 = 4; |
|  | } |
|  | else if(operatie2 == 3 && (operatie != 2 && operatie != 3)) |
|  | if(operand3 == 0){ |
|  | printError(); |
|  | exit(0); |
|  | } |
|  | else |
|  | if(operand2 % operand3 != 0){ |
|  | operand2 = operand2 / operand3; |
|  | parteReala = operand2 % operand3; |
|  | parteReala = parteReala \* 100; |
|  | r = r + parteReala/operand3; |
|  | parteR = (int)r; |
|  | operatie2 = 4; |
|  | } |
|  | else { |
|  | operatie2 = 4; |
|  | operand2 = operand2 / operand3; |
|  | } |
|  |  |
|  | if(operatie == 0){ |
|  | if(operatie2 != 4){ |
|  | MSDelay(10); |
|  | operand2 = operand1 + operand2; |
|  |  |
|  | } else{ |
|  | if(operand2 < 0){ |
|  | DATWRT4('-'); |
|  | } |
|  | rezultat = operand1 + operand2; |
|  | if(parteR != 0){ |
|  | itoa(rezultat); |
|  | print(); |
|  | DATWRT4('.'); |
|  | itoa(parteR); |
|  | print(); |
|  | } else{ |
|  | if(rezultat < 0){ |
|  | DATWRT4('-'); |
|  | } |
|  | itoa(rezultat); |
|  | print(); |
|  | } |
|  | } |
|  |  |
|  | } |
|  | else if(operatie == 1){ |
|  | if(operatie2 != 4){ |
|  | MSDelay(10); |
|  | operand2 = operand1 - operand2; |
|  | }else{ |
|  | if(operand2 > operand1){ |
|  | DATWRT4('-'); |
|  | rezultat = operand2 - operand1; |
|  | } else |
|  | rezultat = operand1 - operand2; |
|  | itoa(rezultat); |
|  | print(); |
|  |  |
|  | } |
|  |  |
|  | } |
|  | else if(operatie == 2){ |
|  | if(operatie2 != 4){ |
|  | operand2 = operand1 \* operand2; |
|  | itoa(operand2); |
|  | print(); |
|  | } else { |
|  | rezultat = operand1 \* operand2; |
|  | if(parteR != 0){ |
|  | itoa(rezultat); |
|  | print(); |
|  | DATWRT4('.'); |
|  | itoa(parteR); |
|  | print(); |
|  | } else{ |
|  | itoa(rezultat); |
|  | print(); |
|  | } |
|  | } |
|  | } |
|  | else if(operatie == 3){ |
|  | if(operatie2 == 4){ |
|  | if(operand2 == 0){ |
|  | printError(); |
|  | exit(0); |
|  | } |
|  | if(operand1%operand2 != 0){ |
|  | int parteIntreaga = operand1/operand2; |
|  | parteReala = operand1%operand2; |
|  | parteReala = parteReala \* 100; |
|  | r = r + parteReala/operand2; |
|  | itoa(parteIntreaga); |
|  | print(); |
|  | DATWRT4('.'); |
|  | itoa(r); |
|  | print(); |
|  | } |
|  | else{ |
|  | rezultat = operand1 / operand2; |
|  | itoa(rezultat); |
|  | print(); |
|  | } |
|  | } else { |
|  | operand2=operand1/operand2; |
|  | } |
|  | } |
|  | if(operatie2 == 0){ |
|  | rezultat = operand2 + operand3; |
|  | if(parteR != 0){ |
|  | itoa(rezultat); |
|  | print(); |
|  | DATWRT4('.'); |
|  | itoa(parteR); |
|  | print(); |
|  | } else{ |
|  | itoa(rezultat); |
|  | print(); |
|  | } |
|  | } |
|  | else if(operatie2 == 1){ |
|  | if(operand3 > operand2){ |
|  | DATWRT4('-'); |
|  | rezultat = operand3 - operand2; |
|  | } else rezultat = operand2 - operand3; |
|  |  |
|  | itoa(rezultat); |
|  | print(); |
|  |  |
|  | } else if(operatie2 == 2){ |
|  | rezultat = operand2 \* operand3; |
|  | itoa(rezultat); |
|  | print(); |
|  | } else if(operatie2 == 3){ |
|  | if(operand2%operand3 != 0){ |
|  | int parteIntreaga = operand2/operand3; |
|  | parteReala = operand2 % operand3; |
|  | parteReala = parteReala \* 100; |
|  | r = r + parteReala/operand3; |
|  | itoa(parteIntreaga); |
|  | print(); |
|  | DATWRT4('.'); |
|  | itoa(r); |
|  | print(); |
|  | } |
|  | else{ |
|  | rezultat = operand2 / operand3; |
|  | itoa(rezultat); |
|  | print(); |
|  | } |
|  |  |
|  | } |
|  | } |
|  |  |
|  | void KEYPADSCAN() { |
|  | unsigned char row; |
|  | unsigned int contorOperanzi = 1; |
|  |  |
|  | while(1){ |
|  | //trebuie facut initial deubounce-ul pentru a nu avea probleme |
|  | //la citire normala |
|  | //citim orice tasta |
|  | do{ |
|  | PORTA = PORTA | 0x0F; //setarea coloanelor |
|  | row = PORTA & 0xF0; //citirea randurilor |
|  | }while(!(row | 0x00));// asteapta pana o tasta este apasata |
|  |  |
|  | do{ |
|  | do{ |
|  | MSDelay(1); |
|  | row = PORTA & 0xF0; //citire rand |
|  | }while(!(row | 0x00));//verificarea apasarii |
|  | //unei taste |
|  | MSDelay(100);//asteapta pentru debounce |
|  | row = PORTA & 0xF0; |
|  | }while(!(row | 0x00)); // fake key press |
|  | //sfarsit initializare/debounce |
|  |  |
|  | //citire normala |
|  | while(1){ |
|  | //coloana 3 |
|  | PORTA = PORTA & 0xF0; //sterge configuratia anterioara |
|  | //a coloanelor |
|  | PORTA = PORTA | 0x08; //seteaza coloana 3 |
|  | MSDelay(10); |
|  | row = PORTA & 0xF0; //sterge configuratia anterioara |
|  | //a randurilor |
|  | if(row | 0x00){ //tasta apasata se afla in coloana 3 |
|  | if(row & 0x10){ //daca tasta apasata se afla |
|  | //pe randul 1 |
|  | DATWRT4('+'); //inseamna ca s-a apasat pe "+" |
|  | if(contorOperanzi == 2) |
|  | operatie2 = 0; |
|  | else operatie = 0; |
|  |  |
|  | contorOperanzi++; |
|  | } |
|  | else if(row & 0x20){ //daca tasta apasata se afla |
|  | //pe randul 2 |
|  | DATWRT4('-'); //inseamna ca s-a apasat pe "-" |
|  | if(contorOperanzi == 2) |
|  | operatie2 = 1; |
|  | else operatie = 1; |
|  |  |
|  | contorOperanzi++; |
|  | } |
|  | else if(row & 0x40){ //daca tasta apasata se afla |
|  | //pe randul 3 |
|  | DATWRT4('\*'); //inseamna ca s-a apasat pe "\*" |
|  | if(contorOperanzi == 2) |
|  | operatie2 = 2; |
|  | else operatie = 2; |
|  |  |
|  | contorOperanzi++; |
|  | } |
|  | else if(row & 0x80){ //daca tasta apasata se afla |
|  | //pe randul 4 |
|  | DATWRT4('/'); //inseamna ca s-a apasat pe "/" |
|  | if(contorOperanzi == 2) |
|  | operatie2 = 3; |
|  | else operatie = 3; |
|  |  |
|  | contorOperanzi++; |
|  | } |
|  | break; //iesi din bucla infinita |
|  | } |
|  |  |
|  |  |
|  | //coloana 2 |
|  | PORTA = PORTA & 0xF0; //sterge configuratia anterioara |
|  | //a coloanelor |
|  | PORTA = PORTA | 0x04; //seteaza coloana 2 |
|  | MSDelay(10); |
|  | row = PORTA & 0xF0; //sterge configuratia anterioara |
|  | //a randurilor |
|  | if(row | 0x00){ //tasta apasata se afla in coloana 2 |
|  | if(row & 0x10){//daca pe primul rand se afla 1 |
|  | DATWRT4('3'); //s-a apasat tasta 3 |
|  | if(contorOperanzi == 1) |
|  | operand1 = operand1 \* 10 + 3; |
|  | else if(contorOperanzi == 2) |
|  | operand2 = operand2 \* 10 + 3; |
|  | else if(contorOperanzi == 3) |
|  | operand3 = operand3 \* 10 + 3; |
|  | } |
|  | else if(row & 0x20){//daca pe al doilea rand se afla 1 |
|  | DATWRT4('6'); //s-a apasat tasta 6 |
|  | if(contorOperanzi == 1) |
|  | operand1 = operand1 \* 10 + 6; |
|  | else if(contorOperanzi == 2) |
|  | operand2 = operand2 \* 10 + 6; |
|  | else if(contorOperanzi == 3) |
|  | operand3 = operand3 \* 10 + 6; |
|  | } |
|  | else if(row & 0x40){//daca pe al trilea rand se afla 1 |
|  | DATWRT4('9'); //s-a apasat tasta 9 |
|  | if(contorOperanzi == 1) |
|  | operand1 = operand1 \* 10 + 9; |
|  | else if(contorOperanzi == 2) |
|  | operand2 = operand2 \* 10 + 9; |
|  | else if(contorOperanzi == 3) |
|  | operand3 = operand3 \* 10 + 9; |
|  | } |
|  | else if(row & 0x80){//daca pe al patrulea rand se afla 1 |
|  | DATWRT4('.'); //s-a apasat tasta . |
|  | } |
|  | break; //iesi din bucla infinita |
|  | } |
|  |  |
|  |  |
|  | //coloana 1 |
|  | PORTA = PORTA & 0xF0; //sterge configuratia anterioara |
|  | //a coloanelor |
|  | PORTA = PORTA | 0x02; //seteaza coloana 1 |
|  | MSDelay(10); |
|  | row = PORTA & 0xF0; |
|  | if(row | 0x00){//tasta apasata se afla in coloana 1 |
|  | if(row & 0x10){ // |
|  | DATWRT4('2'); |
|  | if(contorOperanzi == 1) |
|  | operand1 = operand1 \* 10 + 2; |
|  | else if(contorOperanzi == 2) |
|  | operand2 = operand2 \* 10 + 2; |
|  | else if(contorOperanzi == 3) |
|  | operand3 = operand3 \* 10 + 2; |
|  | } |
|  | else if(row & 0x20){ |
|  | DATWRT4('5'); |
|  | if(contorOperanzi == 1) |
|  | operand1 = operand1 \* 10 + 5; |
|  | else if(contorOperanzi == 2) |
|  | operand2 = operand2 \* 10 + 5; |
|  | else if(contorOperanzi == 3) |
|  | operand3 = operand3 \* 10 + 5; |
|  | } |
|  | else if(row & 0x40){ |
|  | DATWRT4('8'); |
|  | if(contorOperanzi == 1) |
|  | operand1 = operand1 \* 10 + 8; |
|  | else if(contorOperanzi == 2) |
|  | operand2 = operand2 \* 10 + 8; |
|  | else if(contorOperanzi == 3) |
|  | operand3 = operand3 \* 10 + 8; |
|  | } |
|  | else if(row & 0x80){ |
|  | DATWRT4('0'); |
|  | if(contorOperanzi == 1) |
|  | operand1 = operand1 \*10 + 0; |
|  | else if(contorOperanzi == 2) |
|  | operand2 = operand2 \* 10 + 0; |
|  | else if(contorOperanzi == 3) |
|  | operand3 = operand3 \*10 + 0; |
|  | } |
|  | break; //iesi din bucla infinita |
|  | } |
|  |  |
|  |  |
|  | //coloana 0 |
|  | PORTA = PORTA & 0xF0; //sterge configuratia anterioara |
|  | //a coloanelor |
|  | PORTA = PORTA | 0x01; //seteaza coloana 0 |
|  | MSDelay(10); |
|  | row = PORTA & 0xF0; |
|  | if(row | 0x00){ //tasta apasata se afla in coloana 0 |
|  | if(row & 0x10){ |
|  | DATWRT4('1'); |
|  | if(contorOperanzi == 1) |
|  | operand1 = operand1 \* 10 + 1; |
|  | else if(contorOperanzi == 2) |
|  | operand2 = operand2 \* 10 + 1; |
|  | else if(contorOperanzi == 3) |
|  | operand3 = operand3 \* 10 + 1; |
|  | } |
|  | else if(row & 0x20){ |
|  | DATWRT4('4'); |
|  | if(contorOperanzi == 1) |
|  | operand1 = operand1 \* 10 + 4; |
|  | else if(contorOperanzi == 2) |
|  | operand2 = operand2 \* 10 + 4; |
|  | else if(contorOperanzi == 3) |
|  | operand3 = operand3 \* 10 + 4; |
|  | } |
|  | else if(row & 0x40){ |
|  | DATWRT4('7'); |
|  | if(contorOperanzi == 1) |
|  | operand1 = operand1 \* 10 + 7; |
|  | else if(contorOperanzi == 2) |
|  | operand2 = operand2 \* 10 + 7; |
|  | else if(contorOperanzi == 3) |
|  | operand3 = operand3 \* 10 + 7; |
|  | } |
|  | else if(row & 0x80){ |
|  | DATWRT4('='); |
|  | OPERATIE(); |
|  | } |
|  | break; //iesi din bucla infinita |
|  | } |
|  |  |
|  | row = 0; //tasta negasita |
|  | break; //iesi din bucla |
|  | } |
|  | } |
|  | } |
|  |  |
|  | void MSDelay(unsigned int itime) |
|  | { |
|  | unsigned int i; unsigned int j; |
|  | for(i=0;i<itime;i++) |
|  | for(j=0;j<4000;j++); |
|  | } |

**Bibliografie**

* <https://www.nxp.com/docs/en/application-note/AN2727.pdf>