Hangman

**Anderca Georgiana**

**Mureșan Daniel-Lucian**

**Îndrumător: prof. Nimară Sergiu**

# Descriere proiect:

# Implementarea jocului Hangman (Spânzurătoarea) pe placa Dragon12 – Plus2 (2 studenţi)

# Caracteristici:

# • La începutul jocului se generează un cuvânt aleator de lungime variabilă (aleasă de studenţi) şi, pentru fiecare caracter din cuvânt, se afişează „\_” pe prima linie a afişajului LCD.

# • Utilizatorul trebuie să ghicească cuvântul, având 6 încercări.

# • Caracterul dorit este ales de utilizator, la un moment dat, prin folosirea a 3 butoane (pushbuttons): un buton este folosit pentru parcurgerea caracterelor în ordine alfabetică (fiecare apăsare determină trecerea la următoarea literă), un buton va parcurge caracterele în ordine inversă, iar cel de-al treilea buton va confirma selecţia caracterului dorit. Dacă respectiva literă se află în cuvânt, pe poziţiile corespunzătoare din cuvânt se va afişa acea literă, in loc de „\_”.

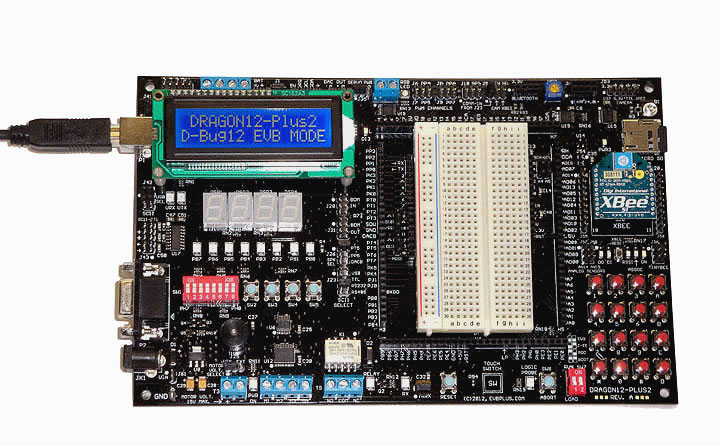
# • În cazul în care caracterul nu se află în cuvânt, numărul de încercări rămase este decrementat. Numărul de încercări se afişează pe LED-uri sau pe un rang din afişajul cu 7 segmente.

# • Jocul se termină în momentul în care cuvântul a fost ghicit sau numărul de încercări a ajuns la 0, afişându-se un mesaj corespunzător pe LCD.

# Descrierea plăcii utilizate

Placa de dezvoltare utilizată, Dragon12 Plus2, predispune de numeroase perficerice și conectori de comunicare și video:

* 1x LCD alfanumeric 16x2 caractere
* 1x afișaj 7-segmente
* USB Host/Device
* Conector VGA
* Slot Micro-SD
* Tastatură
* Comunicare CAN, SPI, I2C



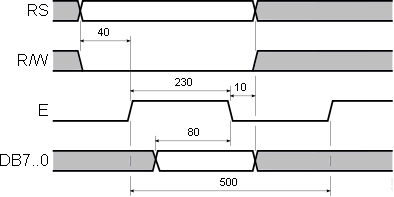
Pentru implementarea jocului Hangman vor fi folosite următoarele periferice: LCD-ul aflanumeric, tastatura și un afișaj 7-segmente.

# Modurile de utilizare a perifericelor

## Afișajul LCD

Afișajul alfanumeric LCD 16 coloane, 2 rânduri folosește un mod de comunicare paralel, linia de date având lungimea de 8 biți, 2 pini de control (RS; R/W), 1 pin de „Enable” (echivalent clock), 2 pini de alimentare și unul pentru ajustarea contrastului.

Adresarea către un afișaj LCD se face cu ajutorul unei diagrame de sincronizare (timing diagram):

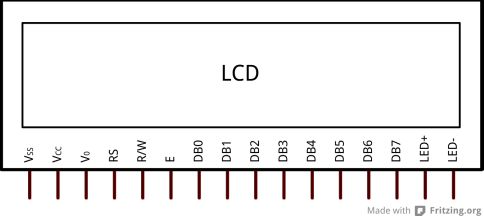


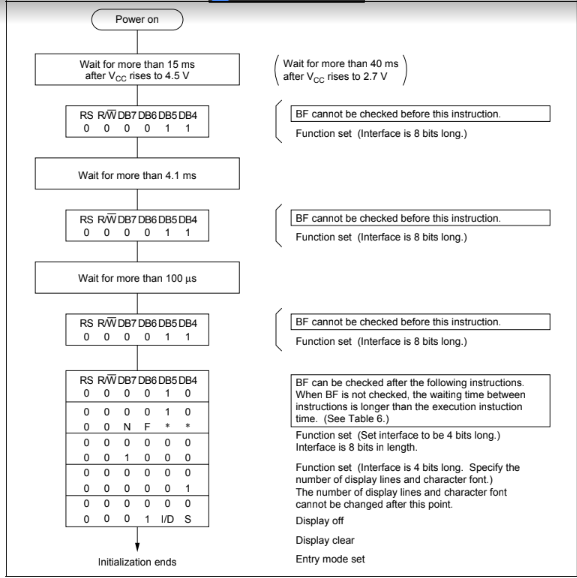
*Unitatea de măsură:* ***ns****.*

Un LCD alfanumeric are 4 categorii de instrucțiuni, fiecare depinzând de starea pinilor RS (Register Select) și R/W („0” sau „1”):

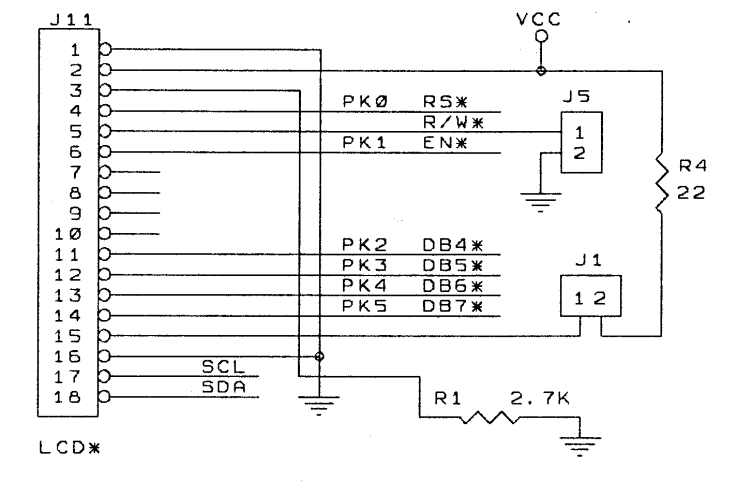
|  |  |  |
| --- | --- | --- |
| R/W | RS | Descriere |
| 0 | 0 | Se trimite pe linia de 8 biți o comandă către LCD: setare cursor, curățare ecran etc. |
| 0 | 1 | Se trimite pe linia de 8 biți un caracter codat ASCII; afișat la poziția cursorului |
| 1 | 0 | Se citește pe linia de 8 biți starea registrului de comandă. |
| 1 | 1 | Se citește pe linia de 8 biți caracterul de la poziția cursorului setată. |

De asemenea, afișajul poate lucra în 2 moduri: lungimea liniei de date să fie pe 4 sau 8-biți lungime (lungimea cuvântului va fi aceeași: 8-biți). Avantajul modului pe 4 biți este la numărul redus de pini utilizați. Accesând acest mod, regiștrii LCD-ului nu se vor mai pot citi (R/W permanent ”0„).





În cazul nostru, afișajul LCD este conectat strict pentru modul în 4 biți pentru a reduce numărul de pini folosiți. De asemenea, afișajul dispune și de un mod de comunicare serial I2C.

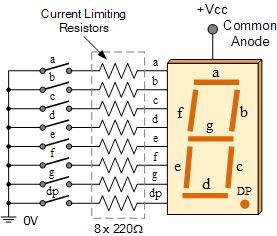


## Afișajul 7-segmente

Afișajul 7 segmente a fost menit pentru afișarea, în mare parte, a numerelor. Pe lângă acestea, pot fi create diferite caractere : A, b, C, d, E, F.

Acest tip de display nu necesită un protocol de comunicare, doar un port GPIO de la uP pentru a aprinde/stinge segmnetele dorite.

Afișarea unei cifre se face pe baza unui LUT (Look-Up Table).

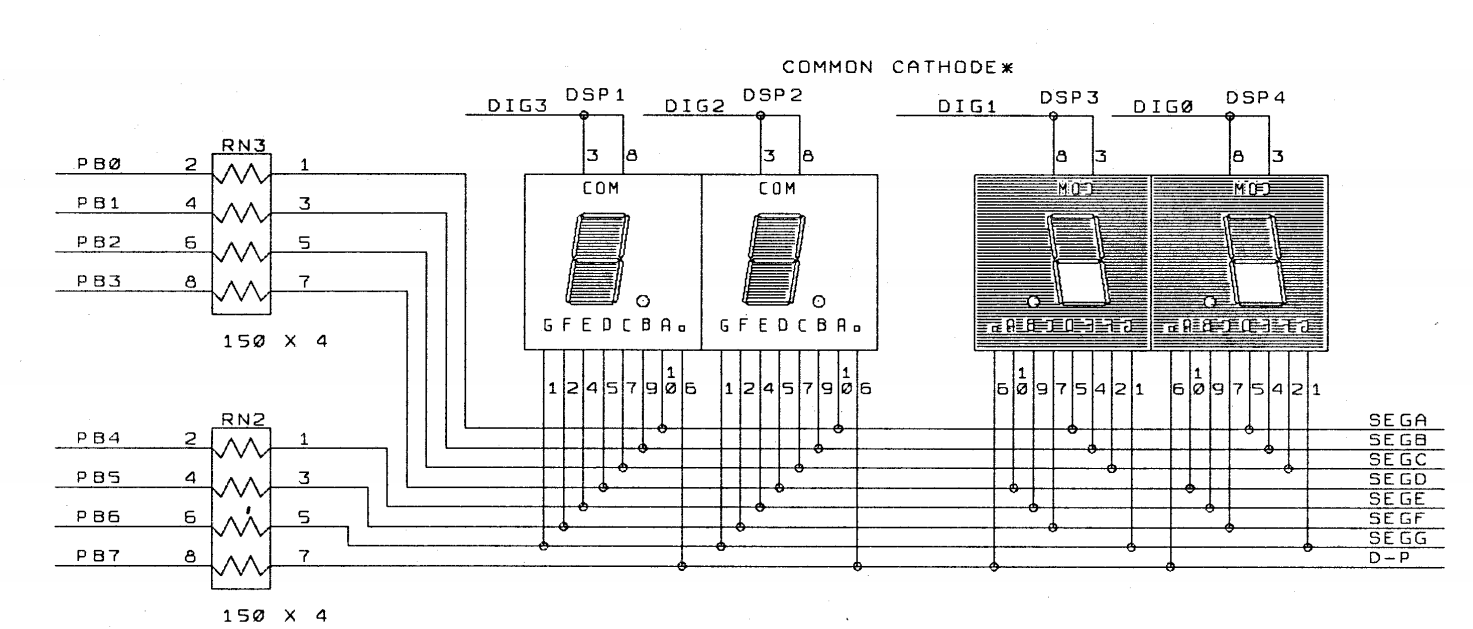


Sunt 2 tipuri de afișaje 7-segmente: AC (anod comun) sau CC (catod comun). Amândouă sunt folosite în absolut aceeași măsură. În cazul în care este folosit afișajul AC, anodul comun se conectează la +V (3.3V / 5V), iar pentru CC, catodul comun se conectează la GND (masă). Diferența apare și la cod: considerăm că a fost scris un LUT pentru CC, unde „1” – segment aprins, „0” – segment stins, atunci pentru AC este necesară inversarea datelor „1” -> „0”; „0” -> „1”. Pentru a nu modifica LUT-ul, se poate utiliza operația „NOT” la portul de ieșire.

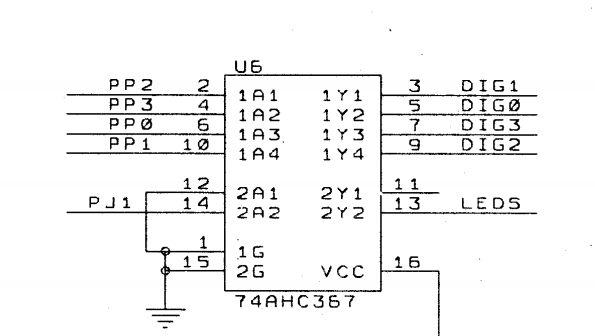
În cazul de față, placa conține 4 afișaje 7 segmente de tip catod comun.

Legăturile între segmentele unui afișaj și microcontroller sunt realizate astfel:

* SEGMENTUL A – PB0
* SEGMENTUL B – PB1
* SEGMENTUL C – PB2
* SEGMENTUL A – PB3
* SEGMENTUL A – PB4
* SEGMENTUL A – PB5
* SEGMENTUL A – PB6
* DOT – PB7



După cum se observă, afișajele sunt multiplexate cu ajutorul U6 (74AHC367), fiind doar un buffer capabil de a livra un curent suficient de mare pentru toate afișajele. Pinii PP[3..0] au rolul de a selecta care dintre afișaje să fie vizibil la un moment dat.

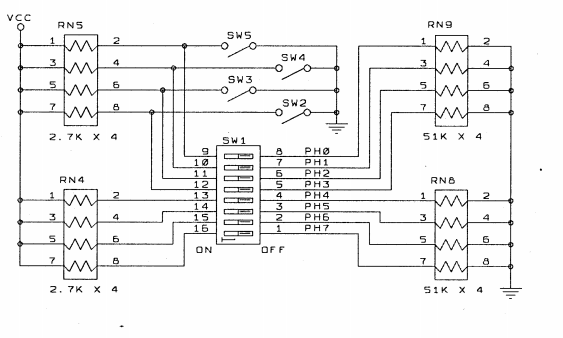


## Tastatura

Se vor folosi 3 din cele 4 butoane disponibile: SW5, SW4 și SW3 care vor fi citite din registrul Port H. Dacă se apasă o tastă, se va citi 0 în registrul PTH. Semnificația butoanelor în contextul aplicației:

* SW5 – selectează litera de la dreapta
* SW4 – validare literă curentă
* SW3 - selectează litera de la stânga

Pentru precizie, s-a introdus în cod o rutină care cere să țină butonul apăsat un anumit timp pentru ca apăsarea să fie o intrare validă. Acest mecanism are rolul de a înlătura zgomotele.



# Descrierea codului

Codul are următoarea structură:

* Definirea rutinelor pentru LCD: CMD4WRT() – pentru trimiterea unei comenzi; DATWRT4()-pentru trimiterea unui caracter, afis\_init() – inițializierea afișajului (setare cursor 0, curățare)
* Definirea rutinelor pentru joc: win(), lose() – afișează dacă jocul a fost câștigat sau pierdut
* Alegerea unui cuvânt dintr-un array pe baza butoanelor apăsate
* Odată inceput jocul, se verifică continuu care din butoane sunt apăsate, iar dacă se detectează un buton apăsat, litera corespunzătoare acestuia este căutată în cuvântul ales. Dacă litera a fost găsită în cuvântul dat, atunci acesta o afișează peste tot unde aceasta se regăsește în cuvânt. În cazul în care litera nu se regăsește, numărul de vieți se va decrementa, modificarea făcându-se vizibilă la afișajul 7 segmente. După verificare, codul trece din nou la verificarea tastaturii.
* Dacă toate literele au fost găsite, adică cuvântul a fost reconstruit, atunci se apelează subrutina win(). Altfel, dacă numărul de vieți este 0, atunci se va apela subrutina lose().

# Codul

//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>

#define LCD\_DATA PORTK

#define LCD\_CTRL PORTK

#define RS 0x01

#define EN 0x02

#define SIZE 10

#define ZGOMOT 50

void COMWRT4(unsigned char);

void DATWRT4(unsigned char);

void MSDelay(unsigned int);

void win();

void afis\_init();

void lose();

void afis\_vieti();

void resume();

int lives=5;

char spanz[11];//cuvantul ales random

void main(void){

char cuvinte[10][11]={"DANI\0","GEORDANI\0","PORTOCALA\0","CEAPA\0","UIUMIMW\0","DANI\0","GEORDANI\0","PORTOCALA\0","CEAPA\0","UIUMIMW\0"};

char alfabet[27]="ABCDEFGHIJKLMNOPQRSTUVWXYZ";

char aux[]="0";

char word\_att[11]="";//cuvant incercare

int i,j,sel=0,randd=0,exitt=0,tofind;

/\* randd - variabila care se incrementeaza pana sa apasa tasta

\*/

DDRK = 0xFF;

DDRH = 0x00; //port butoane

DDRB = 0xFF; //PORTB as output since LEDs are connected to it

DDRJ = 0x01; //PTJ as output to control Dragon12+ LEDs

PTJ=0x0F; //Allow the LEDs to display data on PORTB pins

DDRP=0x0F;

PTP = 0x07;

afis\_vieti();

while (!exitt){

j=0;

while ( PTH\_PTH0==1) //randd se incrementeaza pana apas tasta dreapta

if (randd>10)

randd=0;

else

randd++;

if (PTH\_PTH0==0)

while(PTH\_PTH0==0)

j++;

if (j>ZGOMOT)

exitt=1;

}

strcpy (spanz,cuvinte[randd%10]);

tofind=strlen(spanz); //tofind - nr de liniute

afis\_init();

for(;;){

resume();

if (tofind==0)

win();

if (lives==0)

lose();

//apas butonul drept

j=0;

{MSDelay(1);

if (PTH\_PTH0==0)

while(PTH\_PTH0==0)

j++;

if (j>ZGOMOT)

{

COMWRT4(0xC6);

MSDelay(1);

if (sel!=27)

sel++;

else

sel=0;

DATWRT4(alfabet[sel]);

}}

//apas butonul stang

j=0;

{MSDelay(1);

j=0;

if (PTH\_PTH2==0)

while(PTH\_PTH2==0)

j++;

if (j>ZGOMOT){

COMWRT4(0xC6);

MSDelay(1);

if (sel!=0)

sel--;

else

sel=27;

DATWRT4(alfabet[sel]);

}}

//apas butonul de validare

j=0;

{MSDelay(1);

if (PTH\_PTH1==0)

while(PTH\_PTH1==0)

j++;

if (j>ZGOMOT){

if (!strchr(spanz,alfabet[sel])) //daca n-am gasit litera in spanz

lives--;

else

for (i=0;i<=strlen(spanz);i++)

if(alfabet[sel]==spanz[i])

//trebuie pozitionat cursorul LCD pe i

{MSDelay(1);

COMWRT4(128+i); //128= 0x80, i stie sa puna i unde trebuie

MSDelay(1);

DATWRT4(alfabet[sel]);

tofind--;

aux[0]=alfabet[sel];//aux converteste char alfabet[sel] in string pentru a putea alipi cu strcat

if (strchr(word\_att,alfabet[sel])) //decrementez vietile chiar daca litera e in cuvant

lives--;

else

strcat(word\_att,aux);

MSDelay(1);

}

afis\_vieti();

}

}

}

}

void resume(){

//dupa actualizarea cuvantului, se revine la pozitia de selectare a literei

COMWRT4(0xC6);

}

void afis\_vieti(){

//functie de afisare pe segmente a vietilor

//segmentele sunt cu catod comun, deci se vor aprinde cu “1” logic

if (lives==0)

PORTB = 0x3F;

else

if (lives==1)

PORTB = 0x06;

else

if (lives==2)

PORTB = 0x5B;

else

if (lives==3)

PORTB = 0x4F;

else

if (lives==4)

PORTB = 0x66;

else

if (lives==5)

PORTB = 0x6D;

else

PORTB = 0x7D;

}

void afis\_init(){

//rutina de initializare pentru display si segmente

int i;

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);

MSDelay(1);

for (i=0;i<strlen(spanz);i++){

MSDelay(1);

COMWRT4(128+i);

DATWRT4('\_');

}

COMWRT4(0xC6); //pozitia unde se vor afisa literele de selectat

DATWRT4('A');

}

void lose(){

MSDelay(1);

COMWRT4(0x80);

MSDelay(1);

DATWRT4('A');

MSDelay(1);

COMWRT4(0x81);

MSDelay(1);

DATWRT4('I');

MSDelay(1);

COMWRT4(0x82);

MSDelay(1);

DATWRT4(' ');

MSDelay(1);

COMWRT4(0x83);

MSDelay(1);

DATWRT4('P');

MSDelay(1);

COMWRT4(0x84);

MSDelay(1);

DATWRT4('I');

MSDelay(1);

COMWRT4(0x85);

MSDelay(1);

DATWRT4('E');

MSDelay(1);

COMWRT4(0x86);

MSDelay(1);

DATWRT4('R');

MSDelay(1);

COMWRT4(0x87);

MSDelay(1);

DATWRT4('D');

MSDelay(1);

COMWRT4(0x88);

MSDelay(1);

DATWRT4('U');

MSDelay(1);

COMWRT4(0x89);

MSDelay(1);

DATWRT4('T');

}

void win(){

MSDelay(1);

COMWRT4(0x80);

MSDelay(1);

DATWRT4('A');

MSDelay(1);

COMWRT4(0x81);

MSDelay(1);

DATWRT4('I');

MSDelay(1);

COMWRT4(0x82);

MSDelay(1);

DATWRT4(' ');

MSDelay(1);

COMWRT4(0x83);

MSDelay(1);

DATWRT4('C');

MSDelay(1);

COMWRT4(0x84);

MSDelay(1);

DATWRT4('A');

MSDelay(1);

COMWRT4(0x85);

MSDelay(1);

DATWRT4('S');

MSDelay(1);

COMWRT4(0x86);

MSDelay(1);

DATWRT4('T');

MSDelay(1);

COMWRT4(0x87);

MSDelay(1);

DATWRT4('I');

MSDelay(1);

COMWRT4(0x88);

MSDelay(1);

DATWRT4('G');

MSDelay(1);

COMWRT4(0x89);

MSDelay(1);

DATWRT4('A');

MSDelay(1);

COMWRT4(0x8A);

MSDelay(1);

DATWRT4('T');

}

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 MSDelay(unsigned int itime){

unsigned int i; unsigned int j;

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

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

}

# Bibliografie

* MC9S12DP256 Port Integration Module (PIM) Block User Guide
* HD44780U (LCD-II)
* Dragon12 Schematic C4