LABORATOR 6 - Interfaţă pentru afişor cu cristale lichide (LCD), partea II-a

Scopul lucrării

În laboratorul anterior s-au implementat funcţiile sysinit(), rd_LCDreg(char vRS), wr_LCDreg(char vRS, char data), void ckbf() şi void initLCD().

În continuare se vor implementa şi testa următoarele funcţii:

void putchLCD(char ch)	scrie caracter
void putsLCD(char *ch)	scrie sir de caractere
void clrLCD()	şterge LCD şi mută cursorul în prima poziţie
void gotoLC(unsigned char line, unsigned char col)	mută cursorul

În secţiunea opţională se vor implementa şi alte funcţii. Aceste funcţii se vor scrie în fişierul IOfn.c.

Pasul 1: putchLCD(char ch)

Scrieţi funcţia *putchLCD* în IOfn.c. Pentru ca un caracter să apară pe LCD la poziţia cursorului scrieţi respectivul caracter în registrul **DRegWr.** Nu uitaţi ca înainte de scriere BF să fie ‘0’.

Apoi modificaţi LCDtest.c. Pentru a scrie un caracter LCDtest va avea următoarea structură:

```
int main(){

   sysinit();
   putchLCD('H');

   while(1){
   }
}
```

Dacă adăugați mai multe apeluri *putchLCD* puteţi scrie „Hello Micro”. Nu e obligatoriu.

Pasul 2: putsLCD(…)

Folosind *putchLCD* scrieţi funcţia *putsLCD* – scrie şir de caractere. Primul caracter al mesajului se va scrie la poziţia cursorului. Conform ideii de la pasul 1, cu un singur apel *putsLCD*, scrieți mesajul „Hello Micro”!

Informaţie utilă: în C un sir de caractere se termină 0x00=’\0’.

Pasul 3: clrLCD()

Scrieţi funcţia *clrLCD*. Folosiţi instrucţiunea „Clear Display”. Pentru a testa funcţionarea acestei comenzi, scriem un mesaj, așteptăm 1-2 secunde cu o buclă for şi apoi ștergem display-ul cu *clrLCD*. Structura programului de test este:

```
#define S2 250000UL
...
```

```
int main(){
   sysinit();

   while(1){
      putsLCD("Hello Micro!");
      wait(S2);
      clrLCD();
      wait(S2);
   }
}
```

Pasul 4: gotoLC

Funcţia *putsLCD* pe care aţi scris-o anterior scrie mesajul începând de la poziţia curentă a cursorului. Pentru a putea scrie mesaje pe orice linie, începând cu orice colană, este nevoie de o funcţie care să mute cursorul. Această funcţie se va numi *gotoLC* şi va avea ca parametrii pe *line* şi pe *col*:

void gotoLC(unsigned char line, unsigned char col)

unde *line* este linia pe care se va poziţiona cursorul iar *col* este coloana. **Implementarea funcției se va baza pe comanda „Set DDRAM Address”.**

Pentru a afla adresa DDRAM corespunzătoare unei locaţii LCD **analizaţi** figura 1 de la pagina următoare, linia „Ce se vede” şi linia „Conţinut DDRAM”.

Parametrul *line* poate lua valorile 1 sau 2. Parametrul *col* ia valori in intervalul [1, 16]. Linia Dacă vreunul din parametrii *line* şi *col* sunt invalizi, comanda va fi ignorată. De exemplu *gotoLC*(3,1) este invalidă şi nu va produce efecte.

Testaţi funcţia cu secvenţa:

```
int main(){
   unsigned char k;

   sysinit();

   while(1){
      putsLCD("0123456789abcdef");
      gotoLC(2,1);
      putsLCD("ghijklmnopqrstuv");

      for(k=16;k>0;k--){
         gotoLC(2,k);
         wait(S2);
      }

      for(k=16;k>0;k--){
         gotoLC(1,k);
         wait(S2);
      }

      clrLCD();
      wait(S2);
   }
}
```

Când funcţionează, chemaţi profesorul pentru validare. Dacă aţi ajuns aici aveți nota 5.

Secțiunea opţională

Pasul 5: Wrap

Funcţia *putchLCD*, scrisă anterior, nu verifică dacă adresa din DDRAM la care se scrie caracterul este afişabilă sau nu. Dacă din prima coloană a primei linii scriem un mesaj mai lung de 16 caractere, vor apare numai primele 16 caractere. Încercaţi!

Reamintim că interfaţă LCD are memoria DDRAM alcătuită din 80 de caractere şi poate comanda orice afişaj cu maxim două linii, fiecare linie cu maxim 40 de coloane. Cum afişorul de care dispunem are numai 16 coloane, tot ce se va scrie dincolo de coloana 16 va fi memorat în DDRAM, dar nu se va afişa. De exemplu, dacă scriem pe linia 1, din coloana 16, mesajul „56”. Va apare numai caracterul ‚5’ iar cursorul va dispare:

	Coloana	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16				
Linia 1	Adresa DDRAM	00h	01h	02h	03h	04h	05h	06h	07h	08h	09h	0Ah	0Bh	0Ch	0Dh	0Ef	0Fh	10h	11h	…	27h
	Conţinut DDRAM	20h	20h	20h	20h	20h	20h	20h	20h	20h	20h	20h	20h	20h	20h	20h	35h	36h	20h		20h
	Ce se vede																5	6	_		
Linia 2	Ce se vede																				
	Conţinut DDRAM	20h	20h	20h	20h	20h	20h	20h	20h	20h	20h	20h	20h	20h	20h	20h	20h	20h	20h		20h
	Adresa DDRAM	40h	41h	42h	43h	44h	45h	46h	47h	48h	49h	4Ah	4Bh	4Ch	4Dh	4Ef	4Fh	50h	51h	…	67h

figura 1

1) **Creaţi un nou proiect, identic cu proiectul LCD. Fie numele acestui proiect LCD2.**

2) În noul proiect modificaţi funcţia *putchLCD* astfel încât:

   a) După ce s-a scris caracterul din poziția 16 a liniei 1 cursorul să apară pe linia 2 coloana1.

   b) După ce s-a scris caracterul din poziția 16 a liniei 2 cursorul să apară pe linia 1 coloana1

Atenţie:

1. Nu aveți voie să folosiţi variabile auxiliare pentru a memora poziția cursorului. Poziţia cursorului se află citind SReg.
2. Imediat ce BF a devenit ‚0’ poziţia cursorului nu este cea corectă. Poziţia se va actualiza după cel puţin 6 us. Informaţiile despre acest comportament se găsesc în figura 10 de la pagina 25 din documentul HD44780U.pdf. Pentru a fi siguri că valoarea AC este corectă, aşteptaţi 8 microsecunde.
3. Pentru a aştepta perioade mici de timp, de ordinul microsecundelor, vom utiliza funcţia:

   ```
   void _delay_us(double __us)
   ```

   Această funcţie poate fi folosită dacă adăugăm următoarele două linii de cod **exact** în ordinea de mai jos:

   ```
   #define F_CPU 8000000UL
   #include <util/delay.h>
   ```

   Prototipul funcţiei este definit în *delay.h*. Pentru ca *_delay_us* să funcţioneze corect trebuie specificată frecvenţă ceasului procesor. Specificarea se face prin intermediul constantei `F_CPU`.
4. Cursorul nu trebuie să dispară în nici o situaţie! După ce scrieți un mesaj care are exact lungimea 16, cursorul trebuie să apară în pe linia 2, coloana 1.

Pentru testare, folosiţi următoarea secvenţă de cod :

```
while(1){
   clrLCD();                              //  Ce se vede pe LCD
   wait(S2);
   putsLCD("0123456789abcdef");           //0123456789abcdef
   wait(S2);                              //_

   putchLCD('g');                         //0123456789abcdef
   wait(S2);                              //g_

   clrLCD();                              //_
   wait(S2);                              //


   putsLCD("0123456789abcdefg");          //0123456789abcdef
   wait(S2);                              //g_

   putsLCD("hijklmnopqrstuv");            //0123456789abcdef
   wait(S2);                              //ghijklmnopqrstuv

   putchLCD('w');                         //w123456789abcdef
   wait(S2);                              //ghijklmnopqrstuv
}
```

Când funcţionează, chemaţi profesorul pentru validare. Dacă aţi ajuns aici între 1 şi 2 puncte, în funcție de calitatea implementării.

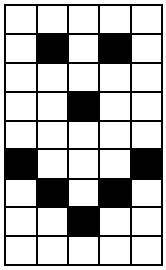
Pasul 6: Smiley

Pentru fiecare caracter se afişează o matrice de puncte. În tabelul 4 de la pagina 18 din documentul HD44780U.pdf se prezintă aceste matrice.

Este posibil ca utilizatorul să definească propriile matrice. Dacă dimensiunea matricei de afişare are dimensiunea 5x10, utilizatorul poate defini 4 matrice proprii. Aceste matrice se scriu în memoria grafică de caractere – CGRAM. Procesul de definire a acestor matrice este prezentat la pagina 20 din documentul HD44780U.pdf.

Corespondentul binar al liniilor care formează o configuraţie se scriu în **CGRAM** la adresa specificată de AC. Exact acelaşi mecanism este folosit şi pentru scrierea în DDRAM. Destinaţia datelor scrise în DReg (Write), DDRAM sau CGRAM, este decisă de ultima instrucţiune Set address executată. După executarea unei instrucţiuni Set DDRAM address până la executarea altei instrucţiuni Set address toate datele scrise în DR (Write) vor ajunge în DDRAM. Similar, după executarea unei instrucţiuni Set CGRAM address toate datele scrise după aceea în DR (Write) vor ajunge în CGRAM.

Se cere ca pentru caracterul cu **codul ASCII 0x02** să apară modelul din figura următoare:



Verificaţi apelând *putchLCD(0x02).* Trebuie să apară Smiley face din figura de mai sus.

```
while(1){
   clrLCD();
   wait(S2);

   putsLCD("This ");
   putchLCD(2);
   putsLCD(" is Smiley!");
   wait(S2);
}
```

Atenţie!

- Scrierea în CGRAM se face **o singura data**, după iniţializare (initLCD).
- Toate funcţiile scrise până acum trebuie să funcţioneze şi în continuare.

Când funcţionează, chemaţi profesorul pentru validare.

Dacă aţi ajuns aici între 1 şi 3 puncte, în funcție de calitatea implementării.

Pasul 7: Deconectare

La terminare executați următoarele operaţii:

1. Se opreşte execuţia programului cu **Break**,
2. Se opreşte depanatorul cu **Stop**
3. Se apasă **Con** şi apoi se şterge memoria de program cu **Erase**
4. Se închide AVR Studio.
5. Se opreşte alimentarea JTAG ICE.
6. Se opreşte sursa Hameg
7. Se desface montajul, numai componentele adăugate în acest laborator.