

# LABORATOR 5 - Afișor cu cristale lichide (LCD), partea a I-a

## Prezentarea afișorului cu cristale lichide.

Afișorul cu cristale lichide ce va fi folosit în continuare are codul DEM16216. Modulul afișor este alcătuit din două componente:

- Controler IO cu codul KS0070B. **Revedeți prelegerea 2, figura 6 și textul corespunzător.**
- Afișorul propriu-zis.

Circuitul integrat KS0070B este compatibil cu circuitul HD44780. Circuitul integrat HD44780 este cel mai popular controler pentru afișoare cu cristale lichide, 99% din controlerile prezente astăzi pe piață fiind compatibile cu acesta. Deoarece documentația pentru afișorul DEM16216 **conține greșeli** și este neclară în multe aspecte vom folosi documentației originală a circuitului HD44780 disponibilă în fișierul **HD44780U.pdf**. În figura 1 este prezentată schema bloc a interfeței:

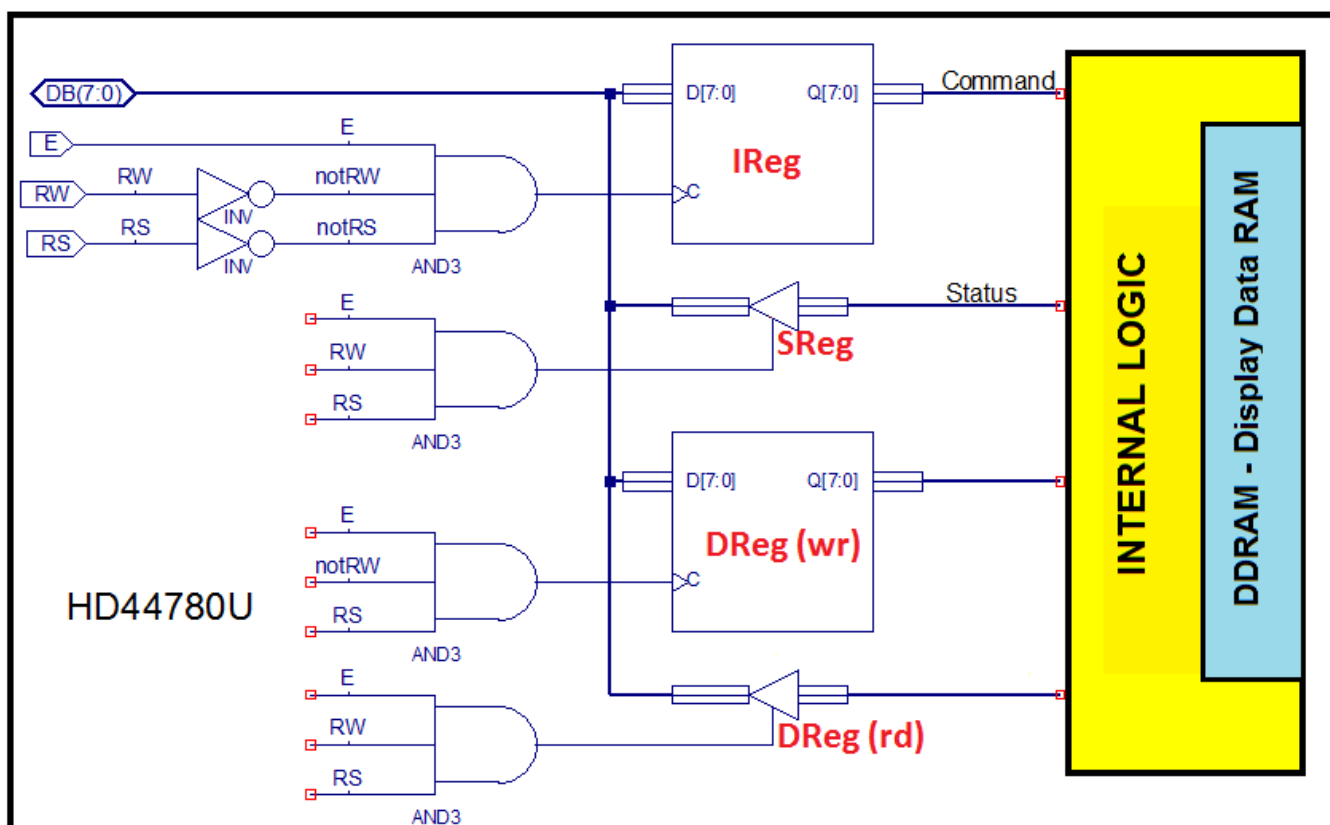


figura 1

Controlerul LCD dispune de **registrele IO** (porturi) discutate la curs: registru de control, registru de stare și registru de date. Vom folosi termenul de registru deoarece acesta este termenul folosit în documentația circuitului.

În datele de catalog registrul de control se numește registru de instrucțiuni - **IReg**, registrul de stare își păstrează numele - **SReg** iar registrul de date este de fapt alcătuit din două registre, unul care poate fi numai scris - **DRegWr** și altul care poate fi numai citit - **DRegRd**. Aceste registre vor fi discutate în detaliu în curând.

Datele afișate pe LCD sunt memorate în memoria internă a controlerului. Această memorie se numește DDRAM - Display Data RAM și are capacitatea de 80 de octeți. **Atenție:** DDRAM este accesibilă numai prin intermediul registrelor interfeței. DDRAM nu poate fi mapată în spațiul de adrese

al procesorului. Un octet care se dorește afișat este mai întâi scris de procesor în registrul DRegWr și apoi logica internă a controlerului mută acest octet în DDRAM. Mai multe detalii în continuare. În concluzie memoria internă NU poate fi accesată direct.

Controlerul poate comanda un LCD cu una sau două linii. Lungimea maximă a unei linii este de 40 de caractere. Afișorul LCD de care dispunem la acest laborator poate afișa două linii, fiecare linie având lungimea de 16 caractere. Când controlerul este configurat astfel încât să se utilizeze o singură linie, se va afișa numai o linie, cea de a doua nefiind utilizată.

Modul în care informația din DDRAM este afișată pe LCD în cazul configurării pe două linii este:

tabelul 1

	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
	Conținut DDRAM	48h	65h	6ch	6ch	6fh	20h	20h	20h	20h	20h	20h	20h	20h	20h	20h	20h
	Ce se vede	H	e	l	l	o											
Linia 2	Ce se vede							w	o	r	l	d	!	_			
	Conținut DDRAM	20h	20h	20h	20h	20h	20h	77h	6fh	72h	6ch	64h	21h	20h	20h	20h	20h
	Adresa DDRAM	40h	41h	42h	43h	44h	45h	46h	47h	48h	49h	4Ah	4Bh	4Ch	4Dh	4Ef	4Fh

## Blocuri din componența controlerului LCD:

### Registre.

Din figura 1 rezultă ca selecția registrului cu care se va executa transferul se face în funcție de starea a trei pini: **E** – Enable, **RS** – register select și **R/W** – read/write. Dacă E='0' nici un registru nu este selectat. Dacă E='1', selecția se face în funcție de starea lui RS și R/W. Rezultă tabelului următor:

tabelul 2

RS	R/W	Semnificația din documentație
0	0	<b>IReg</b> – Instruction Register
0	1	<b>SReg</b> – registru de stare: busy flag (DB7) and AC (DB0 to DB6) - status
1	0	<b>DRegWr</b> – registru pentru înscrierea datelor ce se vor afișa (Data Register Write)
1	1	<b>DRegRd</b> – registru pentru citirea datelor (Data Register Read).

**IReg** stochează codul instrucțiunii trimise către controler, ca de exemplu ștergerea afișajului, sau deplasarea cursorului. **IReg** poate fi doar scris de către microprocesorul extern.

**DRegWr** este folosit pentru memorarea temporară a datelor care trebuie scrise în DDRAM. Data scrisă în **DRegWr** de către procesor este apoi scrisă în DDRAM prin intermediul unor operații interne.

**DRegRd** este folosit pentru memorarea temporară a datelor care trebuie citite din DDRAM. Pentru a citi o dată din DDRAM se specifică mai întâi adresa acesteia printr-o comandă scrisă în **IReg**. Data de la adresa specificată este transferată în **DReg (read)** prin intermediul unei operații interne iar apoi procesorul preia respectiva data prin citirea acestui registru. În plus, după citirea registrului **DRegRd** de procesor, tot printr-o operație internă, se citește data de la următoarea adresă și se scrie în **DRegRd** pentru a fi disponibilă în cazul ca următoarea citire se va face la adresa următoare.

**SReg** (status) este folosit pentru citirea informațiilor de stare. Informația de stare este compusă din BF – Busy Flag și AC – Address Counter. BF și AC se vor detalia în continuare.

### Numărătorul de adrese AC (Address Counter) :

Numărătorul de adrese se află în interiorul blocului „Internal logic” din figura 1. **Rolul numărătorului de adrese AC** este de a specifica adresa locației din memoria DDRAM ce se va scrie sau citi. După fiecare operație AC este incrementat sau decrementat în funcție de starea bitului intern **I/D**, bit setat prin intermediul IReg.

Să presupunem că starea afișorului este cea din figura următoare:

	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
	Conținut DDRAM	48h	65h	6ch	6ch	20h	20h	20h	20h	20h	20h	20h	20h	20h	20h	20h	20h
	Ce se vede	H	e	l	l	_											
Linia 2	Ce se vede																
	Conținut DDRAM	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

AC=4

Prima operație este să scriem în registrul de date DReg (write) caracterului 'o' = 0x6f, bitul I/D fiind setat pe incrementare. După această scriere se obține:

	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
	Conținut DDRAM	48h	65h	6ch	6ch	6fh	20h	20h	20h	20h	20h	20h	20h	20h	20h	20h	20h
	Ce se vede	H	e	l	l	o	_										
Linia 2	Ce se vede																
	Conținut DDRAM	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

AC=5

Caracterul scris, adică ,o', apare în poziția indicată de AC înainte de scriere. După scriere AC se incrementează.

### BF - Busy Flag

Este important de reținut modul de executare a operațiilor cu interfața LCD: orice operație necesită timp pentru a se executa. De exemplu, afișarea unui caracter înseamnă mai întâi scrierea caracterului în **DRegWr**, apoi transferul din **DRegWr** în DDRAM la adresa indicată de AC și în final incrementarea sau decrementarea AC. Astfel întreaga operație necesită 37μS. Atâta timp cât o operație internă este în curs de desfășurare este interzisă lansarea unei noi operații în afară de citirea registrului de stare. Bitul 7 din registrul de stare este BF (Busy Flag): dacă o operație internă este în curs de executare atunci BF=1.

**Regulă: înainte de lansare oricărei operații, așteptați ca BF să fie zero!**

**Cursorul** ( \_ ) indică poziția în care se va face următoarea scriere. Poziția în care se va face următoarea scriere se va numi în continuare poziția curentă. Cursorul este vizibil sau nu, în funcția de modul în care a fost setat bitul **C**, prin intermediul instrucțiunii **Display on/off control**.

Semnificația informațiilor scrise sau citite în/din registrele controlerului se găsește în fișierul **HD44780U.pdf**, de la de la pagina 23 la pagina 29. Mai jos este reprodus tabelul cu comenzile acceptate de interfață:

tabelul 3

Instruction	Code										Description	Execution Time (max) (when $f_{cp}$ or $f_{osc}$ is 270 kHz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DDRAM address 0 in address counter.	
Return home	0	0	0	0	0	0	0	0	1	—	Sets DDRAM address 0 in address counter. Also returns display from being shifted to original position. DDRAM contents remain unchanged.	1.52 ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.	37 $\mu$ s
Display on/off control	0	0	0	0	0	0	1	D	C	B	Sets entire display (D) on/off, cursor on/off (C), and blinking of cursor position character (B).	37 $\mu$ s
Cursor or display shift	0	0	0	0	0	1	S/C	R/L	—	—	Moves cursor and shifts display without changing DDRAM contents.	37 $\mu$ s
Function set	0	0	0	0	1	DL	N	F	—	—	Sets interface data length (DL), number of display lines (N), and character font (F).	37 $\mu$ s
Set CGRAM address	0	0	0	1	ACG	ACG	ACG	ACG	ACG	ACG	Sets CGRAM address. CGRAM data is sent and received after this setting.	37 $\mu$ s
Set DDRAM address	0	0	1	ADD	ADD	ADD	ADD	ADD	ADD	ADD	Sets DDRAM address. DDRAM data is sent and received after this setting.	37 $\mu$ s
Read busy flag & address	0	1	BF	AC	AC	AC	AC	AC	AC	AC	Reads busy flag (BF) indicating internal operation is being performed and reads address counter contents.	0 $\mu$ s
Instruction	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description	
Write data to CG or DDRAM	1	0	Write data								Writes data into DDRAM or CGRAM.	37 $\mu$ s $t_{ADD} = 4 \mu s^*$
Read data from CG or DDRAM	1	1	Read data								Reads data from DDRAM or CGRAM.	37 $\mu$ s $t_{ADD} = 4 \mu s^*$
<div><div>I/D = 1: Increment I/D = 0: Decrement S = 1: Accompanies display shift S/C = 1: Display shift S/C = 0: Cursor move R/L = 1: Shift to the right R/L = 0: Shift to the left DL = 1: 8 bits, DL = 0: 4 bits N = 1: 2 lines, N = 0: 1 line F = 1: 5 <math>\times</math> 10 dots, F = 0: 5 <math>\times</math> 8 dots BF = 1: Internally operating BF = 0: Instructions acceptable</div><div>DDRAM: Display data RAM CGRAM: Character generator RAM ACG: CGRAM address ADD: DDRAM address (corresponds to cursor address) AC: Address counter used for both DD and CGRAM addresses</div><div>Execution time changes when frequency changes Example: When <math>f_{cp}</math> or <math>f_{osc}</math> is 250 kHz, <math>37 \mu s \times \frac{270}{250} = 40 \mu s</math></div></div>												

**Atenție!** În tabelul de mai sus, unde sunt centralizate comenzile controlerului LCD, registrul corespunzător comenzii este specificat neintuitiv. În loc să se precizeze numele registrului, se specifică modul hardware de selecție al acestuia. De exemplu, în loc de IReg, cum ar fi natural, în tabel apare

RS='0' și  $R/\overline{W}$  = '0'. Astfel, pentru comanda „Clear display” în tabel apare RS='0',  $R/\overline{W}$ ='0' și DB(7:0)=0000\_0001<sub>2</sub>=01h.

Similar, în loc de registrul de stare SReg - **Read busy flag and address** - în tabel apare RS='0' și ,  $R/\overline{W}$ ='1', în loc de registrul **DReg (write)** apare RS='1' și  $R/\overline{W}$ ='0' iar în loc de **DReg (read)** apare RS='1' și  $R/\overline{W}$ ='1'.

Citiți documentația înainte de a scrie codul!

## Scopul lucrării

Se va cupla modulul LCD la microcontrolerul AT Mega16 și se vor implementa și testa următoarele funcții:

<i>void sysinit()</i>	inițializare system
<i>unsigned char rd_LCDreg(unsigned char vRS)</i>	citește registru LCD.
<i>void wr_LCDreg(unsigned char vRS, unsigned char data)</i>	scrie registru LCD
<i>void ckbfb()</i>	așteaptă până când BF este 0
<i>void initLCD()</i>	inițializare interfața LCD

Deoarece aceste funcții vor fi necesare în laboratoarele ce vor urma, proiectul va conține două fișiere:

***IOfn.c*** va conține toate funcțiile de mai sus

***LCDtest.c*** conține codul care apelează funcțiile.

Scheletul programului principal se află în fișierul ***LCDtest.c***

## Desfășurarea lucrării

### Pasul 1: Conectarea afișorului cu microcontrolerul

Afișorul DEM16216 se conectează prin intermediul a 16 pini. Pentru conectarea la microcontroler trebuie alocăți 11 pini: 8 pentru DB(7:0) și încă 3 pentru RS,  $R/\overline{W}$  și E. Afișorul se conectează la ATmega16 conform schemei din figura următoare:



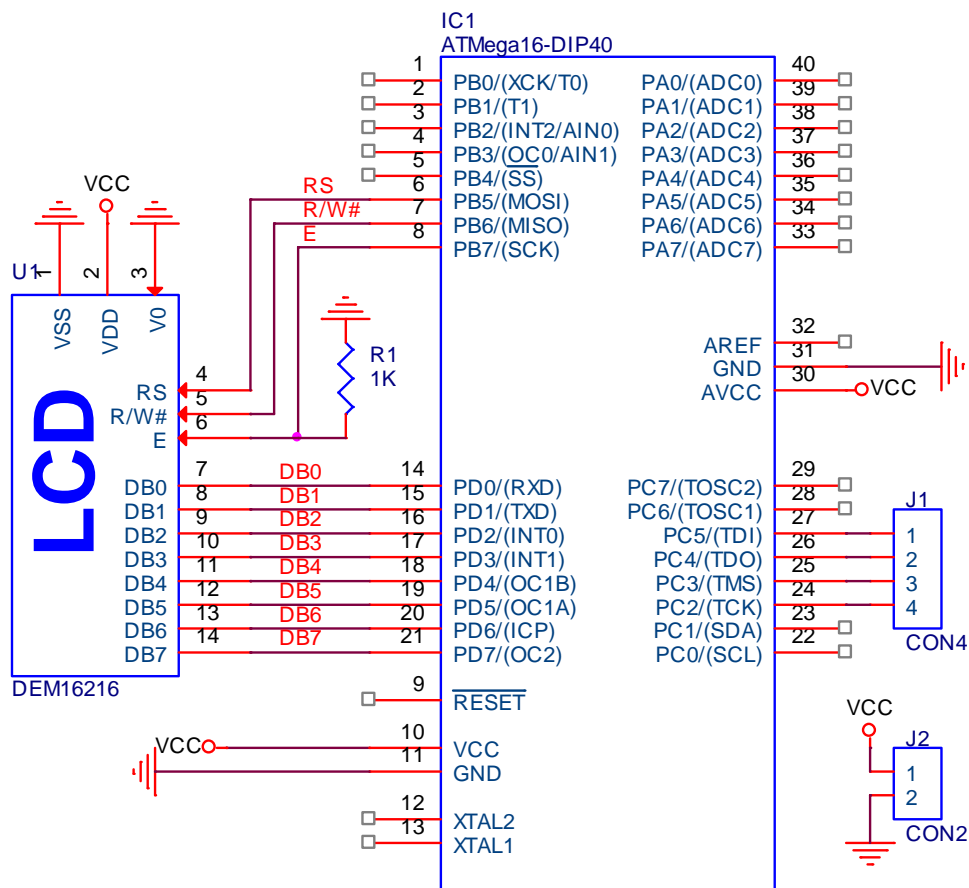
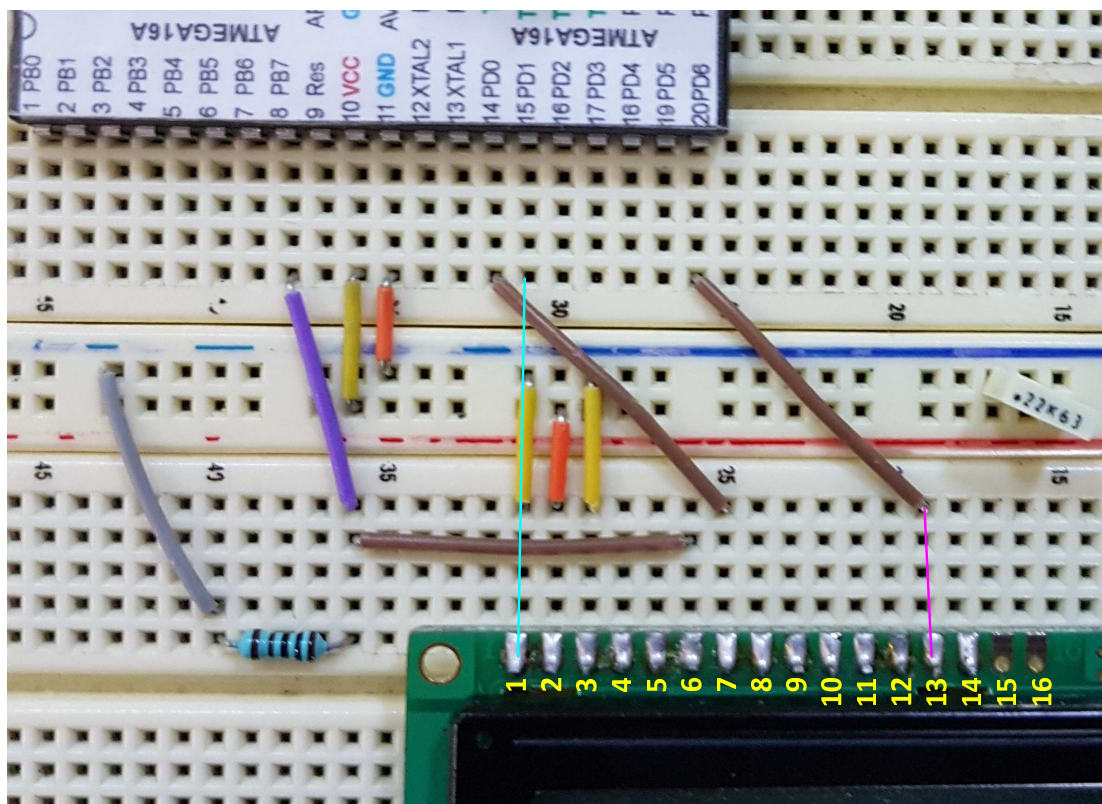


figura 2

În figură sunt făcute și conexiunile pentru alimentarea LCD.

Schema este foarte asemănătoare cu schema din prelegerea 3, figura 15. Pentru a rezulta o conectare cât mai simplă, fără încălecări de fire, fața de curs s-au făcut câteva modificări. portul de date este portul D (în curs este portul C) iar semnalele de control sunt PB5, PB6 și PB7 în loc de PA0-PA3. Principiul de funcționare este însă același.

Amplasarea LCD pe placa de dezvoltare se face conform figurii următoare:



În figură sunt făcute doar conexiunile de alimentare Vss, V0, conexiunea de control E și conexiunile de date DB0 și DB6.

### Indicații de montaj:

1. Montați LCD-ul. Pinul 1 LCD se montează în linia 31. Urmăriți linia de ghidaj **bleu**.
2. Conectați pinii 1, 2 și 3 ale LCD-ului.
3. Conectați pinul 7 LCD (DB0) la pinul PD0 al microcontrolerului.
4. Conectați pinii LCD 8-12 la microcontroler.
5. Conectați pinul 13 LCD (DB6) la pinul PD6 al microcontrolerului. Verificați alinierea cu linia de ghidaj **mov**.
6. Conectați pinul 14 LCD (DB7) la pinul PD7 al microcontrolerului. Pinul PD7 este pe partea cealaltă a microcontrolerului. **Pentru această conexiune folosiți un SINGUR fir roșu lung**, chiar dacă îl montați arcuit (dar lipit de placă).
7. Montați **restul firelor** și rezistența, conform schemei din figura 2.

### NU ALIMENTATI!

## Pasul 2: *sysinit* - Inițializarea sistemului

Proiectul se va numi LCD. La crearea proiectului **NU** bifați căsuța **Create initial file** deoarece sursele sunt furnizate în arhiva lucrării de laborator. Sursele se numesc **LCDtest.c** și **IOfn.c**. **După crearea proiectului copiați aceste surse în folderul proiectului și apoi adăugați-le la proiect.** Pentru a adăuga o sursă/surse la proiect faceți clic dreapta pe **Source files** în fereastra proiectului și din meniul contextual care va apare selectați **Add Existing Source File(s)**.

Prima funcție care se va scrie este funcția care inițializează sistemul. Este nevoie această funcție deoarece în laboratorul LCD și mai ales următoarele laboratoare sunt multe inițializări de făcut.

Ca și în laboratorul precedent, codul care alcătuiește o funcție **este scris cu verde**; restul sunt explicații, comentarii și exemple. Nu le adăugați!

Prima setare care trebuie făcută se referă la direcția și valorile inițiale pentru porturile B și D. Portul D este port de date și va fi alternativ port de ieșire sau port de intrare. Portul D va fi port de intrare numai în funcția de citire registru. În rest va fi port de ieșire pentru a evita fluctuațiile (intrări lăsate „în aer”).

Pinii PB7, PB6 și PB5 din portul B sunt folosiți pentru a genera semnalele de control ale LCD-ului. Ceilalți 5 pini setați-i ca pini de intrare. Valoarea inițială pentru E (PB7) trebuie să fie ,0'. Stabiliți valoarea inițială și direcția pentru pinii portului B.

În final se va aștepta aproximativ o secundă pentru ca LCD-ul să iese din reset. Deoarece LCD-ul nu are pin dedicat de reset trebuie să ne asigurăm că LCD-ul nu mai este în reset când îl programăm.

## Pasul 3: Citirea registrelor interfeței

Următoarele două funcții care trebuie scrise pentru a putea programa modulul LCD sunt citire registru LCD și scriere registru LCD. Pe lângă valorile logice ale semnalelor E, RS, R/ $\bar{W}$  și DB7:0, scriere sau citirea unui registru LCD necesită respectarea unui anumit timing. **Acest timing este foarte semănător cu timingul prezentat în prelegerea 3 capitolul 4, „Conectarea cu interfețe IO”.**

- Semnalul RS este de fapt adresa A0.
- $R/\overline{W}$  specifică tipul transferului și are timing de adresă.
- Semnalul E se comportă ca RD# sau WR# dar E este activ pe ,1' pe când RD# sau WR sunt active pe ,0'.

Vom trata pentru început operația de citire. Timingul operației de citire este prezentat în documentul [HD44780U.pdf](#) la pagina 58, figura 26. Valorile timpilor din această figură se găsesc la pagina 52, secțiunea **Read operation**. În figura următoare este prezentat timingul de citire adnotat cu valorile din tabel:

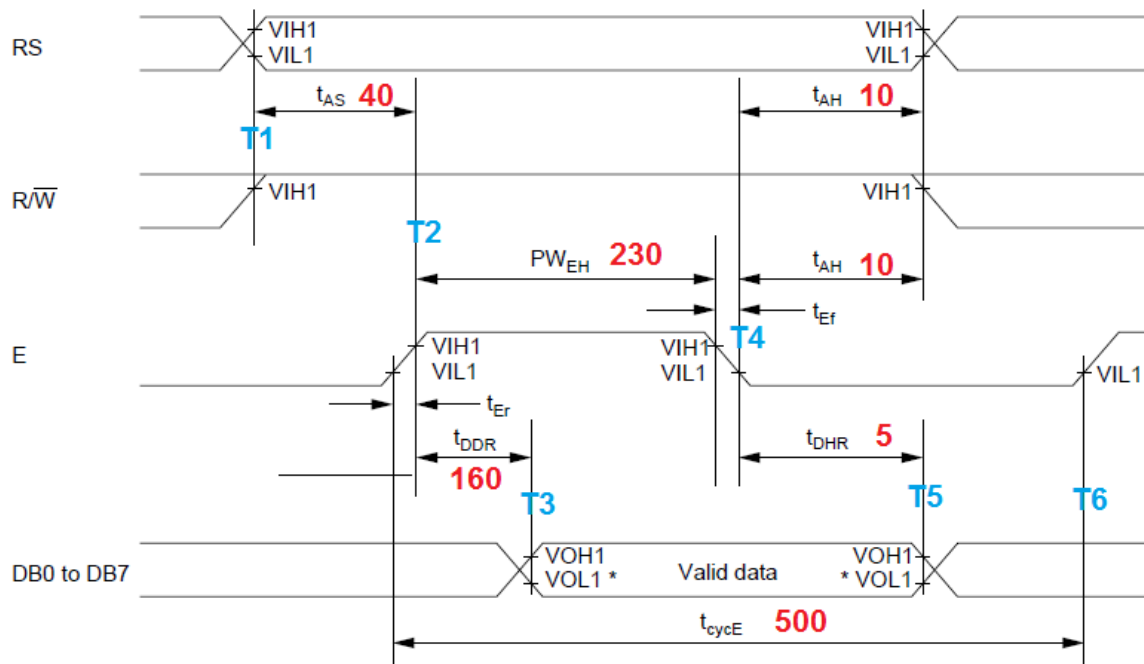


figura 3

**Citirea are același timing indiferent de registrul care se va citi: SReg sau DRegRd. Registrul care se va citi se specifică prin intermediul semnalului RS.** Din acest motiv funcția care implementează citirea registrului va avea un parametru numit vRS (v de la variabilă). vRS se va folosi pentru a stabili valoarea semnalului RS.

Modul în care schema din figura 2 și diagrama de timp din figura 3 este folosită pentru a scrie codul funcției **rd\_LCDreg** este detaliat în continuare. Codul care alcătuiește funcția **este scris cu verde**; restul sunt explicații, comentarii și exemple. Nu le adăugați! **Citiți cu atenție** explicațiile următoare deoarece funcția de scriere registru o veți scrie voi fără indicații!

1. Funcția este declarată cu:

```
unsigned char rd_LCDreg(unsigned char vRS){
```

Funcția primește ca parametru pe vRS. Acesta are exact semnificația din tabelul 2: ,1' = Date, ,0' = Control sau Stare. Funcția întoarce octetul citit din registrul selectat de vRS.

2. În variabila data se va memora temporar octetul citit din registru:

```
unsigned char data;
```

3. Deoarece în cazul citirii din interfață LCD datele sunt generate de interfață și citite de microcontroler, portul D trebuie să fie de tip input.

```
DDRD=???;
```

4. Apoi trebuie stabilită valoarea semnalului RS, generat prin intermediul portului B, bitul 5:



```

if(vRS==0)
    clrbit(PORTB,RS); //RS='0'
else
    setbit(PORTB,RS); //RS='1'

```

Pentru a nu lucra cu valori numerice, în loc

```
clrbit(PORTB,5); //foarte neclar
```

s-a folosit

```
clrbit(PORTB,RS); //devine evidenta intentia
```

Întotdeauna folosiți nume în loc de numere!

Deoarece RS este conectat la bitul 5 din portul B, R/W la bitul 6 iar E la bitul 7 s-au făcut următoarele declarații:

```

#define E 7
#define RW 6
#define RS 5

```

Folosire numelor E, RW, RS în loc de numerele 5, 6 și 7 face codul mult mai clar.

`clrbit` și `setbit` sunt două macro definite la începutul fișierului `IOfn.c`. Dacă vă este neclar modul în care funcționează aceste două macro, recitiți setarea, resetare și testarea biților unui cuvânt din laboratorul 3.

- Deoarece transferul de date cu LCD-ul este de tip citire, semnalul R/W generat pe bitul 6 din portul B trebuie să fie ,1', conform tabelul 2. Setarea lui R/W se face cu:

```
...; //RW=1
```

- În acest moment suntem la momentul **T1** în diagrama din figura 3. Următoarea acțiune este activarea semnalului E (Enable). Din figură se observă ca între T1 și activarea lui E la **T2** trebuie să treacă cel puțin 40 ns. Deci trebuie să așteptăm 40 ns.

Cea mai simplă modalitate de a aștepta o perioadă SCURTA de timp este să executăm un număr de instrucțiuni. O instrucțiune se execută de ATmega într-unul sau în două cicluri de ceas. Cum în cazul nostru frecvența ceasului procesor a fost setată la 8MHZ, execuția unei instrucțiuni necesită 125ns pentru instrucțiunile care se execută într-un ciclu de ceas sau 250 ns pentru cele care se execută în două cicluri.

Deoarece nu avem cum să așteptăm exact 40 ns, singura posibilitate este să executăm o singură instrucțiune cu timpul de execuție de 125 ns. În acest sens există două variante: să declarăm o variabilă de tip **unsigned char** cu specificatorul **volatile** și să o incrementăm sau să executăm o instrucțiune în limbaj de asamblare. Cum varianta cu variabila consumă un octet din memoria RAM, alegem varianta cu instrucțiunea în limbaj de asamblare.

Cea mai folosită instrucțiunea în asamblare pentru a aștepta un ciclu de ceas este **nop** (no operation) deoarece execuția acesteia nu afectează starea uC (registre, indicatori de condiție, etc.). Pentru a executa un **nop** scriem:

```
nop;
```

**nop** este un macro definit la începutul fișierului `iofn.c`. Definiția este suficient de sugestivă și nu mai necesită comentarii.

- În acest moment suntem în diagrama din figura 3 la momentul **T2**. Acum activăm pe E:

```
... //E=1;
```

8. Activarea lui E face ca LCD-ul să genereze pe pinii de date DB7:0 conținutul registrului selectat de RS. Din figură se observă că această operație necesită timp, mai exact 160 ns ( $t_{DDR}$  – timp de propagare). Rezultă că ar trebui să executăm 2 nop-uri pentru a aștepta cele 160 ns.

Mai există însă un fapt de care trebuie să ținem seama: pentru a micșora probabilitatea ca citirile din exterior să genereze metastabilitate, porturile de intrare din ATmega sunt dublu bufferate. Acest fapt introduce o întârziere de maxim două cicluri de ceas, așa că trebuie să executăm încă două **nop**-uri înainte de a citi datele.

Încă un nop este necesar pentru întârzierile datorate capacităților parazite ale plăcii de dezvoltare.

```
nop; nop; nop; nop; nop;
```

9. Suntem la momentul **T3**. Citim datele generate de LCD prin intermediul portului D:

```
data=...;
```

10. După ce am citit datele inactivăm pe E. Deoarece am așteptat 5 nop-uri =  $125 \times 5 = 625$  ns am respectat și timpul cât E trebuie să fie ,1' ( $PW_{EH}$  în figura 3) :

```
... //E=0
```

Suntem la **T4**.

11. Nu schimbăm valorile lui RS și  $R/\bar{W}$  până la următorul acces la LCD, adică cel puțin o instrucțiune = 125 ns. Astfel vom respecta  $t_{ah} = 10$  ns.

12. Ultima acțiune este să așteptăm  $t_{cycE} - PW_{EH} = 500 - 230 = 270$  pentru a atinge T6. Pentru 270 ns am avea nevoie de 3 **nop**-uri. Cum un eventual nou acces la LCD începe cu setarea lui RS – 2 instrucțiuni și a lui RW – o instrucțiune, nu ar mai fi necesare **nop**-uri. Vom adăuga totuși un **nop** pentru a compensa întârzierile datorate capacităților parazite ale plăcii de dezvoltare:

```
nop;
```

Este posibil și cu mai puține **nop**-uri dacă ne asigurăm că funcția *rd\_LCDreg* nu este transformată de vreo optimizare a compilatorului în macro. Dacă *rd\_LCDreg* este funcție mai avem întârzieri cu transferul parametrilor, apelul de funcție, transferul rezultatului și return.

13. Între transferuri portul D trebuie să fie de tip ieșire pentru a evita ca semnalele DB să fluctueze.

```
DDRD=???;
```

14. Și sfârșitul!

```
return data;
}
```

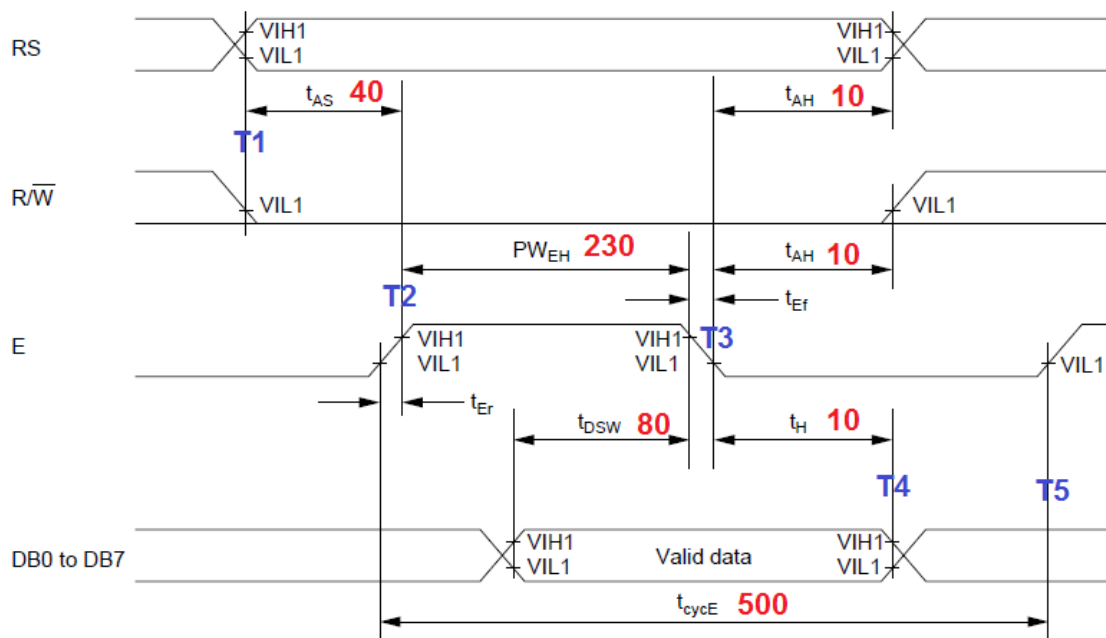
#### Pasul 4: Scrierea registrelor interfeței

Folosind ca model citirea registrului LCD, **se cere scrierea** funcției

```
void wr_LCDreg(unsigned char vRS, unsigned char data)
```

în fișierul IOfn.c.

Timingul operației de scriere este prezentat în documentul [HD44780U.pdf](#) la pagina 58, figura 25. Valorile timpilor din această figură se găsesc la pagina 52, secțiunea **Write operation**. În figura următoare este prezentat timingul scrierii adnotat cu valorile din tabel:



La fel ca în cazul citirii, **scrierea are același timing indiferent de registrul care se va scrie: IReg sau DRegWr. Registrul care se va scrie se specifică prin intermediul semnalului RS.** Din acest motiv, la fel ca în cazul citirii, funcția care implementează **scrierea** registrului va avea un parametru numit vRS. vRS se va folosi pentru a stabili valoarea semnalului RS.

Pașii care trebuie urmați în scrierea codului pentru funcției

**void wr\_LCDreg(unsigned char vRS, unsigned char data)**

sunt:

1. Stabiliți valoarea semnalului RS.
2. Stabiliți valoarea semnalului RW. Ați ajuns la T1.
3. Scrieți datele în portul D. Chiar dacă datele ar putea fi scrise și mai târziu nu se întâmplă nimic dacă se scriu înainte de activarea lui E.
4. Așteptați până la T2.
5. Activați E.
6. Așteptați până la T3.
7. Inactivați E.
8. Așteptați până la T5.
9. Ați terminat!

**Atenție! La pașii următori accesați registrele controlerului LCD prin intermediul celor două funcții deja scrise: rd\_LCDreg și wr\_LCDreg.**

De exemplu, pentru a inițializa LCD-ul trebuie scrisă în IReg comanda „Clear display”. Codul acestei comenzi este 0x01 (vezi tabelul 3). Deci va trebui să scriem:

```
wr_LCDreg(0, 0x01);
```

primul parametru este 0 și reprezintă valoarea lui RS. În ideea unui cod clar s-au făcut următoarele declarații:

```
#define IReg 0
#define SReg 0
#define DReg 1
```

Folosind aceste declarații, comanda „Clear display” se va trimite cu:

```
wr_LCDreg( IReg, 0x01 );
```

Astfel codul devine mult mai clar.

**Obligatoriu:** Citiți sau scrieți registrele LCD în stilul de pe linia de mai sus!

### Pasul 5: *ckbf()*

Se scrie în fișierul IOfn.c funcția **ckbf** (check Busy Flag). În funcție se așteaptă atâta timp cât BF – Busy Flag – este ,1’. Pentru a folosi registrul corect, folosiți tabelul 3. Dacă acesta nu este suficient, folosiți și documentația controlerului din [HD44780U.pdf](#)

### Pasul 6: *initLCD()*

Se scrie funcția **initLCD**. Se fac următoarele setări:

- Function set: DL- 8 biți, N - 2 linii, F - 5x10 dots.
- Entry mode set: I/D - increment, S - no shift
- Display on/off control: D - display on, C - la latitudinea voastră, B - la latitudinea voastră. Unul din biții C și B trebuie să fie obligatoriu setat, pentru ca să apară cursorul.
- Clear display

Documentația pentru aceste setări se găsește în tabelul 3 și în [HD44780U.pdf](#)

**Este obligatoriu să se documenteze modul în care se fac TOATE setările de mai sus** – vezi ca model „Function set” în IOfn.c.

**Decomentați apelul lui initLCD() din sysinit();**

**Obs. Nu uitați să verificați BF înainte de fiecare comandă trimisă către interfața LCD.**

### Pasul 7: Testare HW+SW

Compilați proiectul cu **Build →Rebuild All**.

**Chemați profesorul pentru validarea codului și a montajului.**

Dacă funcțiile au fost scrise corect trebuie să apară cursorul pe prima poziție.

**Atenție:** în cazul în care ceva nu funcționează corespunzător, probabil că ați trimis comenzi greșite către controler. În consecință starea acestuia este necunoscută. După ce modificați sursele, controlerul ar trebui resetat înainte de a trimite noile comenzi (presupus corecte). Această operație se face oprind și apoi repornind JTAG ICE și sursa de alimentare Hameg. Trebuie procedat în acest fel deoarece controlerul LCD nu are pin de RESET.

**Dacă apare cursorul pe prima poziție, chemați profesorul.**

### Pasul 8: Deconectare

La terminare executați următoarele operații:

1. Se oprește execuția programului.

2. Se oprește depanatorul.
3. **Se șterge memoria de program.**
4. Se închide AVR Studio.
5. Se oprește alimentarea JTAG ICE.
6. Se oprește sursa Hameg
7. Se desface montajul, numai componentele adăugate în acest laborator.