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

## Prezentarea afişorului cu cristale lichide.

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

- Controler IO cu codul KS0070B. Revedeţi prelegerea 2, figura 6 şi textul corespunzător.
- Afişorul propriu-zis.

Circuitul integrat KS0070B este compatibil cu circuitul HD44780. Circuitul integrat HD44780 este cel mai popular controler pentru afişoare cu cristale lichide, 99% din controlerele prezente astăzi pe piaţă fiind compatibile cu acesta. Deoarece documentaţia pentru afişorul DEM16216 conţine greşeli şi este neclară în multe aspecte vom folosi documentaţiei originală a circuitului HD44780 disponibilă în fişierul **HD44780U.pdf**. În figura 1 este prezentată schema bloc a interfeţei:

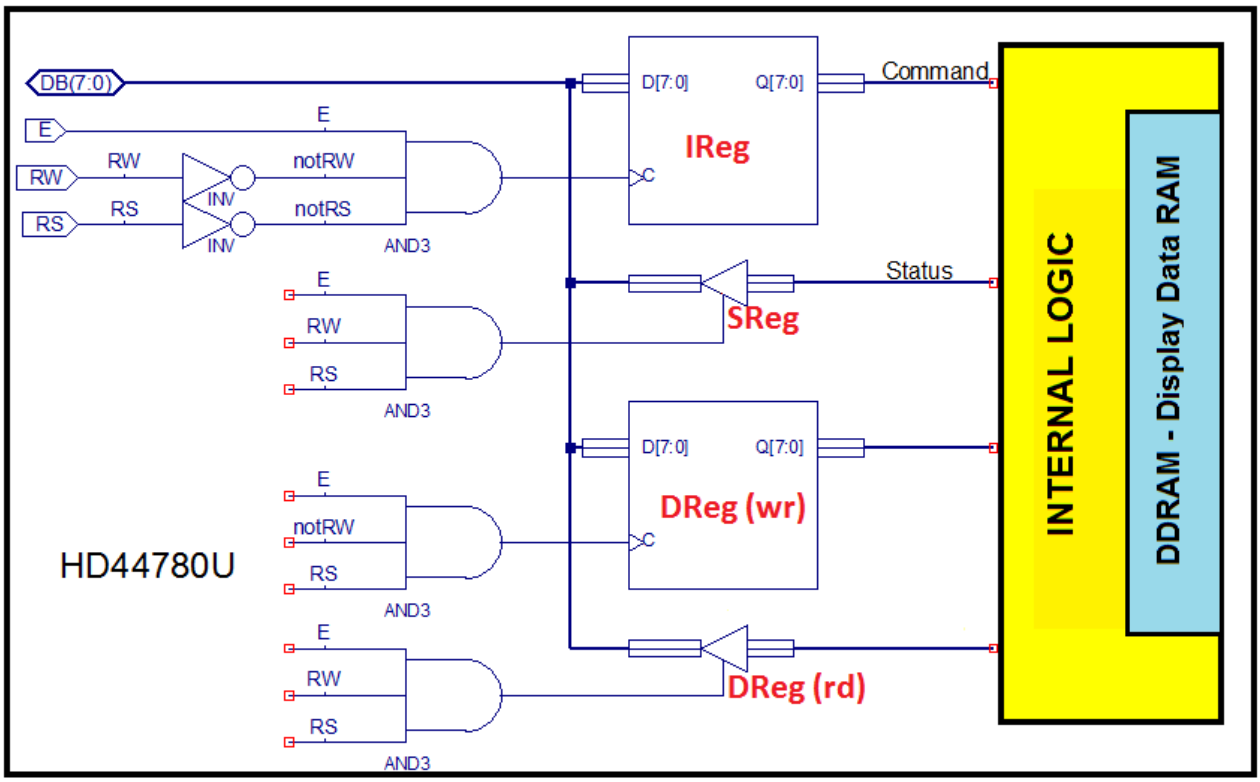



**figura 1**

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

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

Datele afişate pe LCD sunt memorate în memoria internă a controlerului. Această memorie se numeşte DDRAM - Display Data RAM şi are capacitatea de 80 de octeţi. **Atenţie**: DDRAM este accesibilă numai prin intermediul registrelor interfeţei. DDRAM nu poate fi mapată în spațiul de adrese

al procesorului. Un octet care se doreşte afişat este mai întâi scris de procesor în registrul DRegWr şi 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. Afişorul LCD de care dispunem la acest laborator poate afişa două linii, fiecare linie având lungimea de 16 caractere. Când controlerul este configurat astfel încât să se utilizeze o singură linie, se va afişa numai o linie, cea de a două nefiind utilizată.

Modul în care informaţia din DDRAM este afişată 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 |
| | Conţinut 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** | **!** | **_** | | | |
| | Conţinut 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 componenţa controlerului LCD:**

**Registre.**

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

**tabelul 2**

| **RS** | **R/$\overline{W}$** | **Semnificaţia din documentaţie** |
|---|---|---|
| 0 | 0 | **IReg** – Instruction Register |
| 0 | 1 | **SReg** – registru de stare: busy flag (DB7) and AC (DB0 to DB6) - status |
| 1 | 0 | **DRegWr** – registru pentru înscrierea datelor ce se vor afişa (Data Register Write) |
| 1 | 1 | **DRegRd** – registru pentru citirea datelor (Data Register Read). |

**IR**eg stochează codul instrucţiunii trimise către controler, ca de exemplu ştergerea afişajului, sau deplasarea cursorului. IReg poate fi doar scris de către microprocesorul extern.

**DR**egWr 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 operaţii interne.

**DR**egRd 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 operaţii interne iar apoi procesorul preia respectiva data prin citirea acestui registru. În plus, după citirea registrului DRegRd de procesor, tot printr-o operaţie internă, se citeşte data de la următoarea adresă şi se scrie în DRegRd pentru a fi disponibilă în cazul ca următoarea citire se va face la adresa următoare.

**SR**eg (status) este folosit pentru citirea informaţiilor de stare. Informaţia de stare este compusă din BF – Busy Flag şi AC – Address Counter. BF şi 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 specifică adresa locaţiei din memoria DDRAM ce se va scrie sau citi. După fiecare operaţie AC este incrementat sau decrementat în funcţie de starea bitului intern I/D, bit setat prin intermediul IReg.

Să presupunem că starea afişorului 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 |
| | Conţinut 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 | | | | | | | | | | | | | | | | |
| | Conţinut 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 operaţie este să scriem în registrul de date DReg (write) caracterului '**o**' = 0x6f, bitul I/D fiind setat pe incrementare. După această scriere se obţine:

| | **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 |
| | Conţinut 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 | | | | | | | | | | | | | | | | |
| | Conţinut 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 poziţia indicată de AC înainte de scriere. După scriere AC se incrementează.

**BF - Busy Flag**

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

**Regulă: înainte de lansare oricărei operaţii, aşteptaţi ca BF să fie zero!**

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

Semnificaţia informaţiilor scrise sau citite în/din registrele controlerului se găseşte în fişierul **HD44780U.pdf**, de la de la pagina 23 la pagina 29. Mai jos este reprodus tabelul cu comenzile acceptate de interfaţă:

**tabelul 3**

| Instruction | Code RS | R/W | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 | Description | Execution Time (max) (when $f_{cp}$ or $f_{OSC}$ is 270 kHz) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 µs |
| 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 µs |
| Cursor or display shift | 0 | 0 | 0 | 0 | 0 | 1 | S/C | R/L | — | — | Moves cursor and shifts display without changing DDRAM contents. | 37 µs |
| 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 µs |
| 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 µs |
| 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 µs |
| 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 µs |
| Write data to CG or DDRAM | 1 | 0 | Write data | | | | | | | | Writes data into DDRAM or CGRAM. | 37 µs<br>$t_{ADD}$ = 4 µs* |
| Read data from CG or DDRAM | 1 | 1 | Read data | | | | | | | | Reads data from DDRAM or CGRAM. | 37 µs<br>$t_{ADD}$ = 4 µs* |

IReg

SReg

DReg Wr

DReg Rd

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 × 10 dots, F = 0: 5 × 8 dots
BF = 1: Internally operating
BF = 0: Instructions acceptable

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

Execution time changes when frequency changes
Example:
When $f_{cp}$ or $f_{OSC}$ is 250 kHz,
$37\ \mu s \times \frac{270}{250} = 40\ \mu s$

**Atenţie!** 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 selecţie al acestuia. De exemplu, în loc de IReg, cum ar fi natural, în tabel apare

RS=’0’ şi R/$\overline{W}$ =’0’. Astfel, pentru comanda „Clear display” în tabel apare RS=’0’, R/$\overline{W}$ =’0’ şi DB(7:0)=0000_0001$_2$=01h.

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

Citiţi documentaţia înainte de a scrie codul!

## Scopul lucrării

Se va cupla modulul LCD la microcontrolerul AT Mega16 şi se vor implementa şi testa următoarele funcţii:

| | |
|---|---|
| ***void sysinit()*** | iniţializare sistem |
| ***unsigned char rd_LCDreg(unsigned char vRS)*** | citește registru LCD. |
| ***void wr_LCDreg(unsigned char vRS, unsigned char data)*** | scrie registru LCD |
| ***void ckbf()*** | așteaptă până când BF este 0 |
| ***void initLCD()*** | inițializare interfața LCD |

Deoarece aceste funcţii vor fi necesare în laboratoarele ce vor urma, proiectul va conține trei fișiere:

| | |
|---|---|
| ***IOfn.c*** | conține toate funcțiile de mai sus |
| ***defs.h*** | conține definiții, prototipurile funcțiilor din IOfn.c care sunt folosite în LCDtest.c etc. de exemplu initLCD() este folosit numai în ***sysinit()*** și de aceea prototipul sau nu va fi scris în defs.h |
| ***LCDtest.c*** | conține codul care testează funcțiile de LCD |

Scheletul programului principal se află în fişierul LCDtest.c

## Desfăşurarea lucrării

### Pasul 1: Conectarea afişorului cu microcontrolerul

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

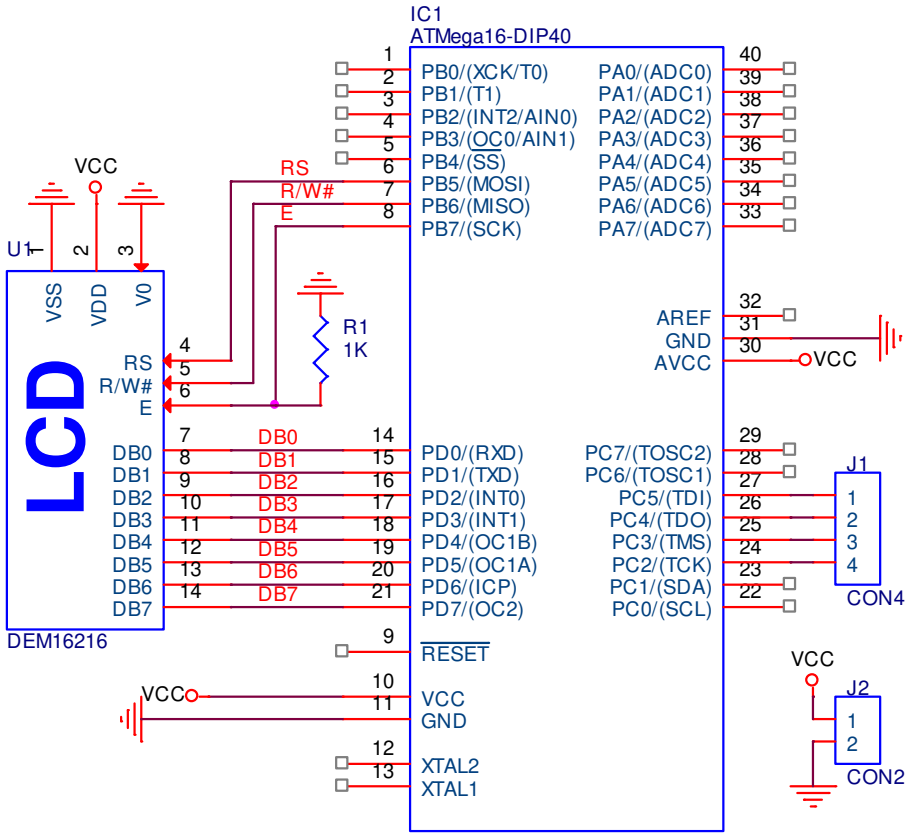
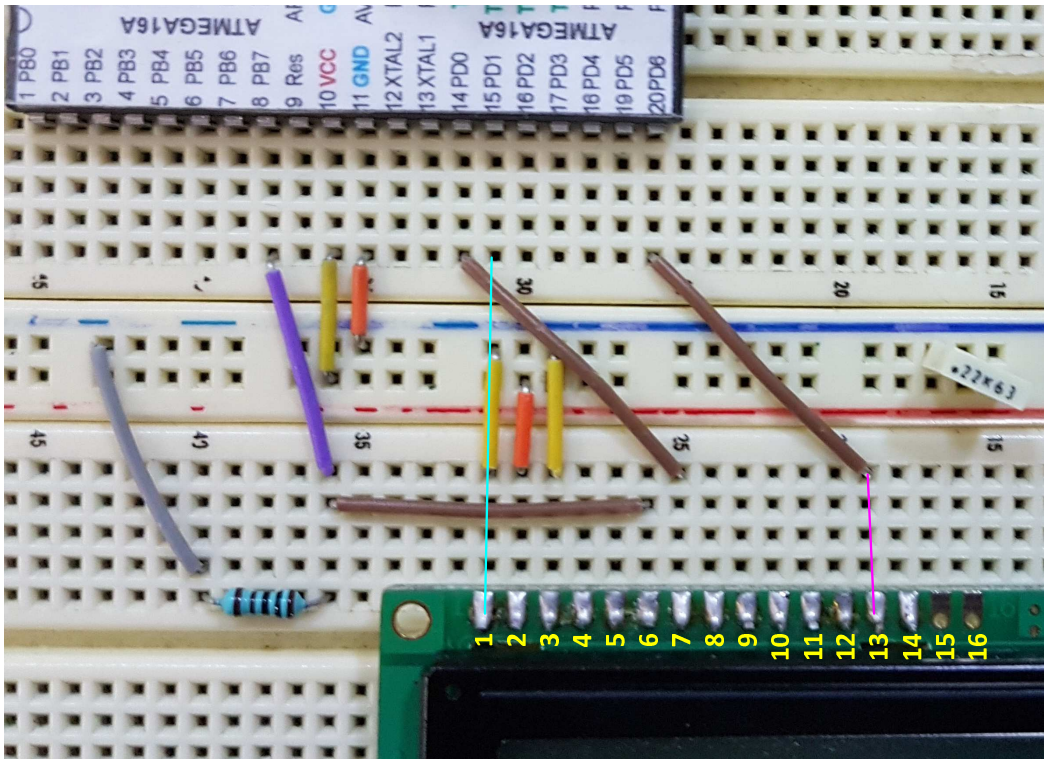



**figura 2**

În figură sunt făcute şi conexiunile pentru alimentarea LCD.

Schema este foarte asemănătoare cu schema din prelegerea 3, figura 19. Pentru a rezulta o conectare cât mai simplă, fără încălecări de fire, faţa 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 şi PB7 în loc de PA0-PA3. Principiul de funcţionare este însă acelaşi.

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 şi conexiunile de date DB0 şi DB6.

**Indicaţii de montaj:**

1. Montaţi LCD-ul. Pinul 1 LCD se montează în linia 31. Urmăriţi linia de ghidaj **bleu**.
2. Conectaţi pinii 1, 2 şi 3 ale LCD-ului.
3. Conectaţi pinul 7 LCD (DB0) la pinul PD0 al microcontrolerului.
4. Conectaţi 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 roşu lung**, chiar dacă îl montaţi arcuit (dar lipit de placă).
7. Montaţi **restul firelor** şi rezistenţa, conform schemei din figura 2.

**NU ALIMENTATI!**

## Pasul 2: *sysinit* - Inițializarea sistemului

În proiect vor exista cele trei fișiere discutate anterior: defs.h, LCDtest.c şi IOfn.c. Creați proiectul după cum urmează:

- **Creați** proiectul după procedura explicată în laboratoarele precedente. Proiectul se va numi LCD. **Întotdeauna bifați** opțiunea „Create initial file". Altfel proiectul ar fi gol iar proiectele goale se comport eronat.
- Sursele defs.h, LCDtest.c şi IOfn.c se găsesc în arhiva platformei de laborator. **Copiați aceste surse** în folderul proiectului și apoi adăugaţi-le la proiect. Pentru a adăuga o sursă/surse la proiect faceți clic dreapta pe **Source files** în fereastra proiectului şi din meniul contextual care va apare selectați **Add Exixting Source File(s)**.
- **Eliminați** din proiect fișierul LCD.c creat și adăugat automat la proiect deoarece ați bifat „Create initial file". Acum avem deja 3 surse în proiect și putem sa eliminăm pe LCD.c care a fost creat numai ca proiectul să nu fie gol. Pentru a elimina o sursă din proiect faceți clic dreapta pe **Source files** în fereastra proiectului și din meniul contextual care va apare selectați **Delete File**.

Prima funcție care se va scrie este funcţia care iniţializează sistemul. Este nevoie această funcție deoarece în laboratorul LCD şi mai ales următoarele laboratoare sunt multe iniţializări de făcut.

Ca şi în laboratorul precedent, codul care alcătuieşte o funcţie **este scris cu verde**; restul sunt explicaţii, comentarii şi exemple. Nu le adăugaţi!

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

Pinii PB7, PB6 şi PB5 din portul B sunt folosiţi pentru a genera semnalele de control ale LCD-ului. Ceilalţi 5 pini setaţi-i ca pini de intrare. Valoarea iniţială pentru E (PB7) trebuie să fie ‚0'. Stabiliţi valoarea iniţială şi direcţia pentru pinii portului B.

În final se va aştepta 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 interfeţei

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

- 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# din curs sunt active pe ‚0’.

Vom trata pentru început operaţia 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:

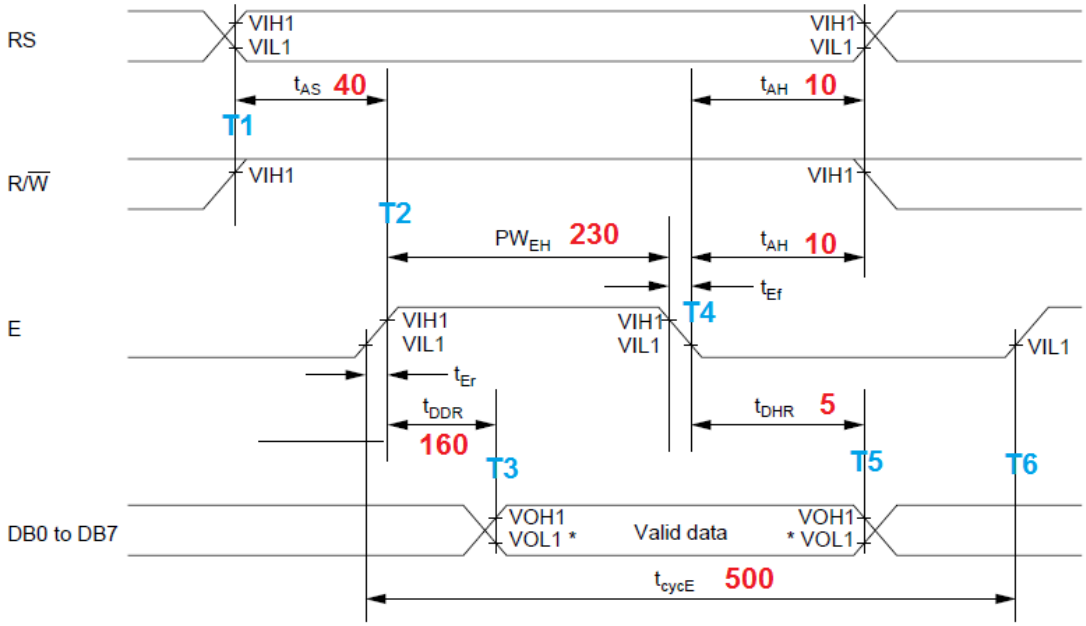



**figura 3**

**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 funcţia 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 alcătuieşte funcţia **este scris cu verde**; restul sunt explicaţii, comentarii şi exemple. Nu le adăugaţi! **Citiți cu atenție** explicaţiile următoare deoarece funcţia de scriere registru o veţi scrie voi fără indicaţii!

1. Funcţia este declarată cu:

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

Funcţia primeşte ca parametru pe vRS. Acesta are exact semnificația din tabelul 2: ‚1' = Date, ‚0'=Control sau Stare. Funcţia 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 interfaţă LCD datele sunt generate de interfaţă şi citite de microcontroler, portul D trebui 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 intenţia
```

Întotdeauna folosiţi 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 iofn.c în fișierul :

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

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

`clrbit` şi `setbit` sunt două macro definite în fișierului defs.h. Dacă vă este neclar modul în care funcţionează aceste două macro, recitiţi setarea, resetare şi testarea biților unui cuvânt din laboratorul 3.

5. 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
```

6. În acest moment suntem la momentul **T1** în diagrama din figura 3. Următoarea acţiune este activarea semnalului E (Enable). Din figură se observă ca între T1 şi activarea lui E la **T2** trebuie să treacă cel puţin 40 ns. Deci trebuie să aşteptăm 40 ns.

   Cea mai simplă modalitate de **a aștepta o perioadă scurtă de timp** este să executăm un număr de instrucţiuni. O instrucţiune se execută de ATmega într-unul sau în două cicluri de ceas. Cum în cazul nostru frecvenţa ceasului procesor a fost setată la 8MHZ, execuţia unei instrucţiuni necesită 125ns pentru instrucţiunile care se execută într-un ciclu de ceas sau 250 ns pentru cele care se execută în două cicluri.

   Deoarece nu avem cum să aşteptăm exact 40 ns, singura posibilitate este să executăm o singură instrucţiune cu timpul de execuţie de 125 ns. În acest sens există două variante: să declarăm o variabilă de tip **unsigned char** cu specificatorul **volatile** şi 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 instrucţiunea în limbaj de asamblare.

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

```
nop;
```

**nop** este un macro definit în fișierului defs.h. Definiţia este suficient de sugestivă şi nu mai necesită comentarii.

7. 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 sa executăm 2 nop-uri pentru a aștepta cele 160 ns.

   Mai există însă un fapt de care trebuie să ţinem seama: pentru a micşora 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, aşa că trebuie să executăm încă două **nop**-uri înainte de a citi datele.

   Încă un nop este necesar pentru întârzierile datorate capacităţilor 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 = 125x5=625 ns am respectat şi timpul cât E trebuie să fie ‚1' ($PW_{EH}$ în figura 3) :

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

    Suntem la **T4**.

11. Nu schimbăm valorile lui RS şi R/$\overline{W}$ până la următorul acces la LCD, adică cel puţin 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 şi a lui RW – o instrucțiune, nu ar mai fi necesare **nop**-uri. Vom adăugă totuşi un **nop** pentru a compensa întârzierile datorate capacităţilor parazite ale plăcii de dezvoltare:

```
nop;
```

    Este posibil şi 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 trebui să fie de tip ieşire pentru a evita ca semnalele DB să fluctueze.

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

14. Şi sfârşitul!

```
return data;
}
```

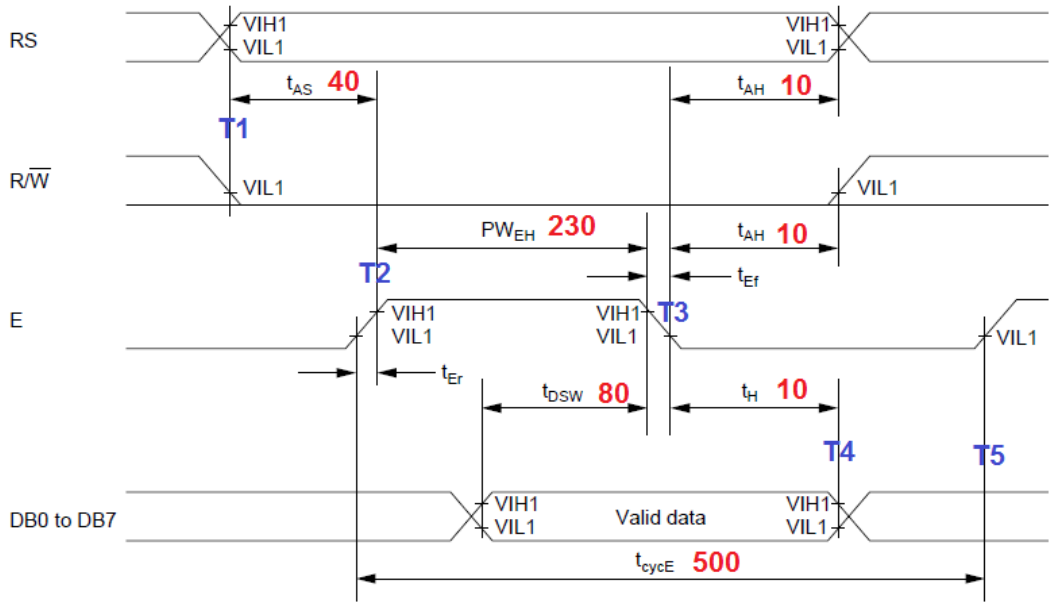
## Pasul 4: Scrierea registrelor interfeţei

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

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

în fişierul 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, funcţia care implementează **scrierea** registrului va avea un parametru numit vRS. vRS se va folosi pentru a stabili valoarea semnalului RS.

Paşii care trebuie urmaţi în scrierea codului pentru funcţiei

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

sunt:

1. Stabiliţi valoarea semnalului RS.
2. Stabiliţi valoarea semnalului RW. Aţi ajuns la T1.
3. Scrieți datele în portul D. Chiar dacă datele ar putea fi scrise şi mai târziu nu se întâmplă nimic dacă se scriu înainte de activarea lui E.
4. Aşteptaţi până la T2.
5. Activaţi E.
6. Aşteptaţi până la T3.
7. Inactivaţi E.
8. Aşteptaţi până la T5.
9. Aţi terminat!

**Atenție! La paşii următori accesaţi registrele controlerului LCD prin intermediul celor două funţii deja scrise: *rd_LCDreg* şi *wr_LCDreg*.**

De exemplu, pentru a iniţializa 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 şi 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 dispay” se va trimite cu:

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

Astfel codul devine mult mai clar.

**Obligatoriu:** Citiţi sau scrieţi 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 aşteaptă atâta timp cât BF – Busy Flag – este ‚1’. Pentru a folosi registrul corect, folosiți tabelul 3. Dacă acesta nu este suficient, folosiţi şi documentaţia controlerului din **HD44780U.pdf**

### Pasul 6: *initLCD*()

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

- Function set: DL- 8 biţi, N - 2 linii, F - 5x8 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 biţii C şi B trebuie să fie obligatoriu setat, pentru ca să apară cursorul.
- Clear display

Documentaţia 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.

Obs. Nu uitaţi să verificaţi BF înainte de fiecare comandă trimisă către interfaţa LCD.

### Pasul 7: Testare HW+SW

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

**Chemaţi profesorul pentru validarea codului şi a montajului.**

Dacă funcţiile au fost scrise corect trebuie să apară cursorul pe prima poziţie.

**Atenţie:** în cazul în care ceva nu funcţionează corespunzător, probabil că aţi trimis comenzi greşite către controler. În consecinţă starea acestuia este necunoscută. După ce modificaţi sursele, controlerul ar trebui resetat  înainte de a trimite noile comenzi (presupus corecte). Această operaţie se face oprind şi apoi repornind JTAG ICE şi sursa de alimentare Hameg. Trebuie procedat în acest fel deoarece controlerul LCD nu are pin de RESET.

**Dacă apare cursorul pe prima poziţie, chemaţi profesorul.**

## Pasul 8: Deconectare

La terminare executați următoarele operaţii:

1. Se opreşte execuţia programului.
2. Se opreşte depanatorul.
3. **Se şterge memoria de program.**
4. Se închide AVR Studio.
5. Se opreşte alimentarea JTAG ICE.
6. Se oprește sursa Hameg
7. Se desface montajul, numai componentele și firele adăugate în acest laborator.