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

Prezentarea afișorului cu cristale lichide.

Orice afișor LCD este alcătuit din două componente:

- Interfața propriu zisă Revedeți prelegerea 2, figura 6 și textul corespunzător.
- Matricea de puncte din cristale lichide (dot-matrix liquid crystal display)

Controlerul IO are codul HD44780. Circuitul integrat HD44780 este cel mai popular controler pentru afișoare cu cristale lichide, 99% din controlerele prezente astăzi pe piață fiind compatibile cu acesta. Documentația pentru circuitului HD44780 este disponibilă în fișierul **HD44780U.pdf**. În figura 1 este prezentată schema bloc a interfeței:

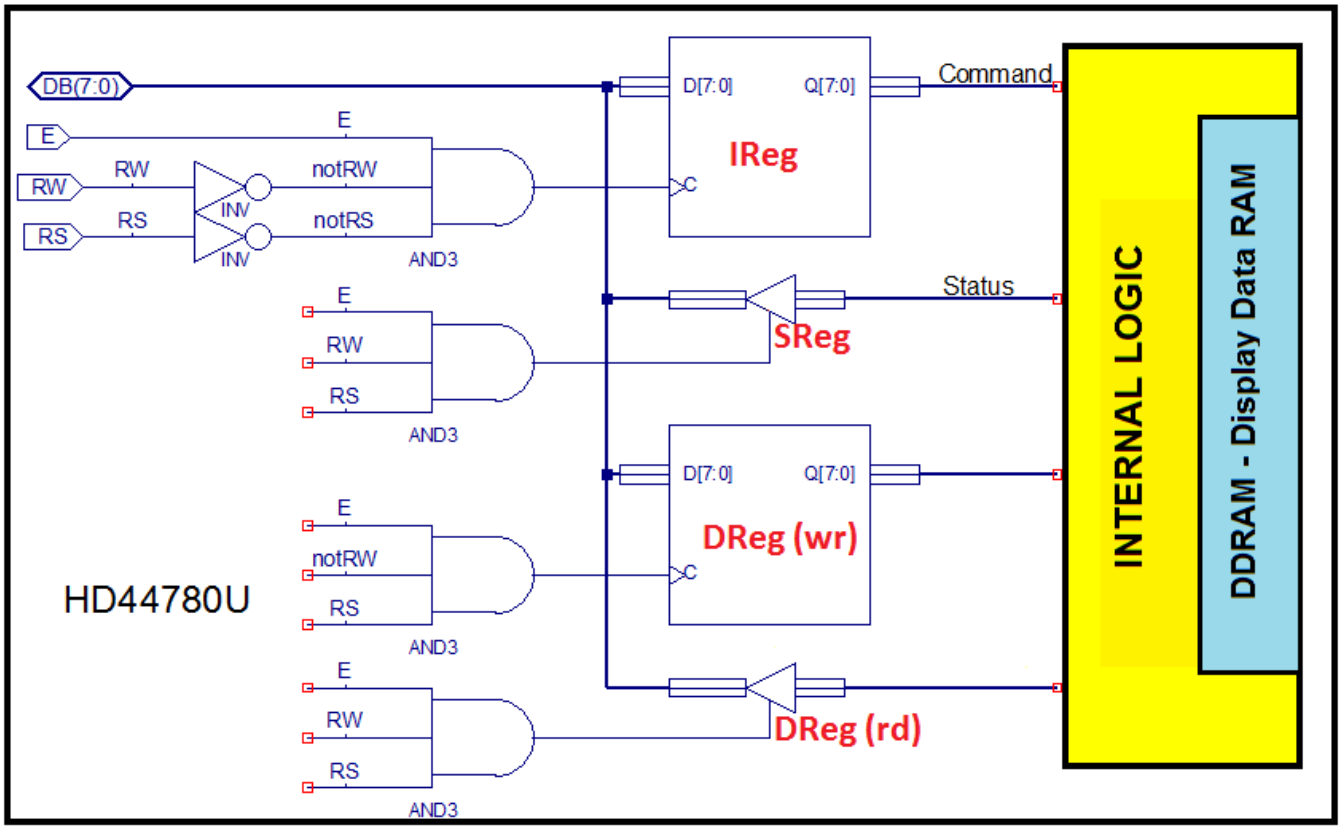


figura 1

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 - **IR**eg, registrul de stare își păstrează numele - **SR**eg iar registrul de date este de fapt alcătuit din două registre, unul care poate fi numai scris - **DR**egWr și altul care poate fi numai citit - **DR**egRd. 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 două 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
	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

Caracterele din tabelul anterior sunt **codificate ASCII.** De exemplu, caracterul ‚H' are valoarea 48h (0x48).

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/$\bar{W}$** – read/write. Dacă E='0' nici un registru nu este selectat. Dacă **E='1'**, selecția se face în funcție de starea lui RS şi R/W. Rezultă tabelului următor:

tabelul 2

RS	**R/$\bar{W}$**	**Semnificația din documentație**
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 afișa (Data Register Write)
1	**1**	**DRegRd** – registru pentru citirea datelor ce se afișează, adică au fost scrise anterior (Data Register Read).

IReg stochează codul instrucțiunii trimise către controler, ca de exemplu ştergerea afişajului, 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 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 **DR**egRd 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 specifică 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
	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 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
	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 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 orice operație cu interfața LCD, BF trebuie să fie zero! Această condiție se poate poate fi îndeplinită în două moduri: fie se citește BF și se execută operația după ce BF este ‚0', fie se așteaptă un timp mai mare decât durata de execuție a operației. De exemplu, pentru afișarea unui caracter trebuie așteptat mai mult de 37µs deoarece aceasta este durata de execuție a acestei operații.

Deoarece **simulatorul nu funcționează corect in cazul citirii din interfața LCD**, după fiecare operație cu interfața LCD **se va aștepta** un timp mai mare decât durata de execuție a operației.

În mod normal pentru a aștepta o anumită perioadă de timp se folosește una din funcțiile *_delay_ms*(...) sau *_delay_us*(...). In laboratorul doi s-a folosit funcțis *_delay_ms*(...). Din păcate în simulare funcțiile de delay **nu funcționează întotdeauna corect**. Din acest motiv se va folosi o versiune echivalenta de funcție *delay*. Această funcție se numește *wait(unsigned long int delay)* și este implementată după cum urmează:

```
volatile unsigned long int delay;
void wait(unsigned long int val){
   for(delay=0;delay<val;delay++){}
```

Pentru a aștepta aproximativ o secundă valoarea parametrului *delay* este 250000. Creșterea clarității codului se face cu ajutorul următoarelor definiții:

```
//S = secundă
#define S  250000UL

//MS= milisecundă
#define MS 250UL

//US = microsecundă
#define US 25UL/100UL
```

În loc de

```
wait(250000);
```

se scrie

```
wait(1*S);
```

și codul devine mai clar.

Funcția *wait* definită anterior nu ne permite să așteptăm intervale de timp mai mici de 4 microsecunde. Pentru a aștepta intervale mai mici de 4 us, vom folosi o instrucțiune are nevoie de o singură perioadă de ceas procesor pentru a fi executată. O astfel de instrucțiune este:

```
volatile unsigned char cnt8;
...
#define WAIT125   cnt8++
```

Timpul de execuție pentru această instrucțiune este de 125 ns când frecvența ceasului procesor este 8 MHz. Prin utilizarea repetată a acestei instrucțiuni putem aștepta *n**125ns, *n*>0:

```
WAIT125;   așteaptă 125 ns
…

WAIT125;   așteaptă 250 ns
WAIT125;
…

WAIT125;   așteaptă 375 ns
WAIT125;
WAIT125;
…
```

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 funcție 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	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 t_{ADD} = 4 µs*
Read data from CG or DDRAM	1	1	Read data								Reads data from DDRAM or CGRAM.	37 µs 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,
$37\ \mu s \times \frac{270}{250} = 40\ \mu s$

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$_2$=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:

void wr_LCDreg(unsigned char vRS, unsigned char data)	scrie registru LCD
void initLCD()	inițializare interfața LCD
void sysinit()	inițializare sistem

Deoarece aceste funcții 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 LCD

Scheletul programului principal se află în fișierul LCDtest.c

Desfășurarea lucrării

Pasul 1: Conectarea afișorului cu microcontrolerul

Afișorul DEM16216 se conectează la microcontroler prin intermediul a 11 pini: 8 pentru DB(7:0) și încă 2 pentru RS și E.

Deoarece operația de citire nu funcționează în simulare, semnalul R/W# se conectează la masă. Ca urmare semnalul R/W# est tot timpul ‚0'. Deoarece R/W# este tot timpul ‚0', singurul tip de transfer care se poate face cu interfață LCD este transferul de scriere.

Conectarea interfeței LCD la microcontroler se face conform schemei din figura următoare:

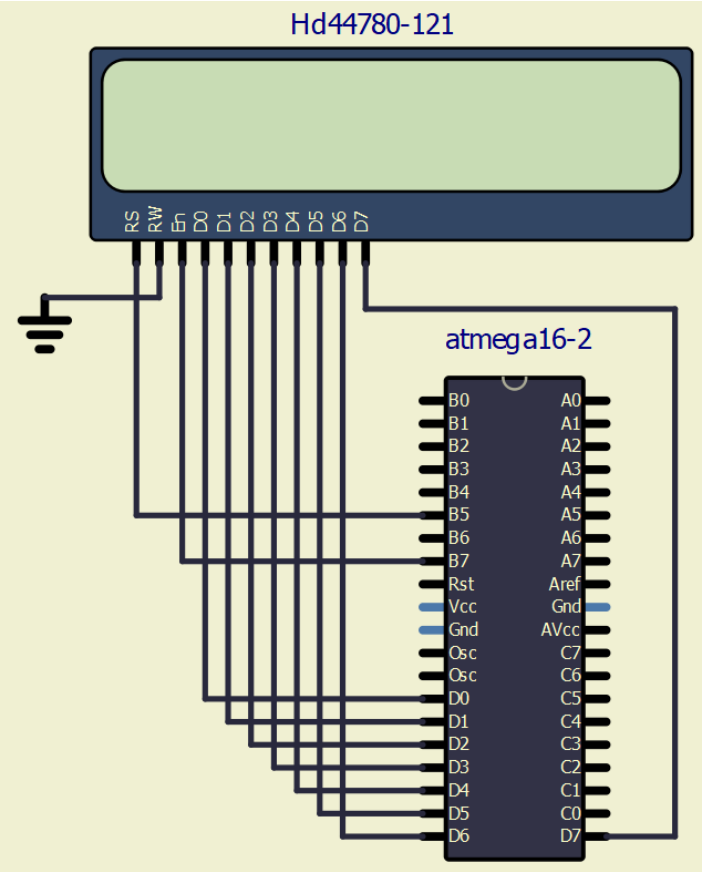
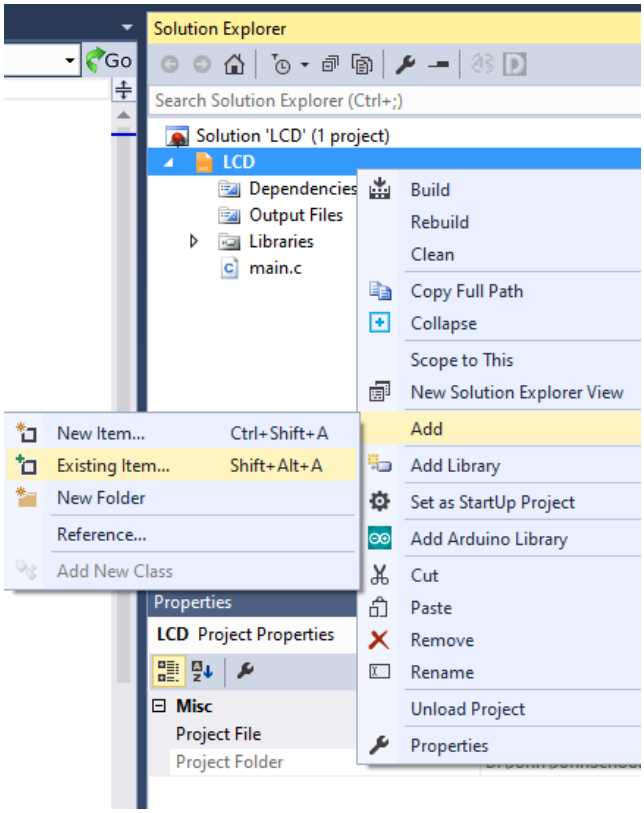


figura 2

Schema este foarte asemănătoare cu schema din prelegerea 3, figura 19. 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 și PB7 în loc de PA0-PA3. Principiul de funcționare este însă același.



Pasul 2: Crearea proiectului

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

- **Creați** proiectul după procedura explicată în laboratoarele precedente. Proiectul se va numi LCD.
- Sursele defs.h, LCDtest.c și IOfn.c se găsesc în arhiva platformei de laborator. **Copiați aceste surse** în subfolderul LCD al folderului LCD.

LCD este folderul soluției iar subfolderul LCD este folderul proiectului. Adăugați aceste surse la proiect.

Pentru a adăuga o sursă/surse la proiect faceți clic dreapta pe numele proiectului (LCD) în fereastra soluției și din meniul contextual care va apare selectați **Add,** iar apoi selectați **Existing Item... .** Vezi figura alăturată.

- **Eliminați** din proiect fișierul **main.c** creat și adăugat automat la proiect de Atmel Studio. Pentru a elimina o sursă din proiect, în fereastra

proiectului, faceți clic dreapta pe acea sursă și din meniul contextual care va apare selectați **Remove** și apoi alegeți **Delete.**

Ca și în laboratorul 3, codul care alcătuiește o funcție **este scris cu verde**; restul sunt explicații, comentarii și exemple. Nu le adăugați!

Pasul 3: Scrierea registrelor interfeţei

Prima funcție care se va scrie în fișierul IOfn.c este funcția este funcția

void wr_LCDreg(unsigned char vRS, unsigned char data)

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:

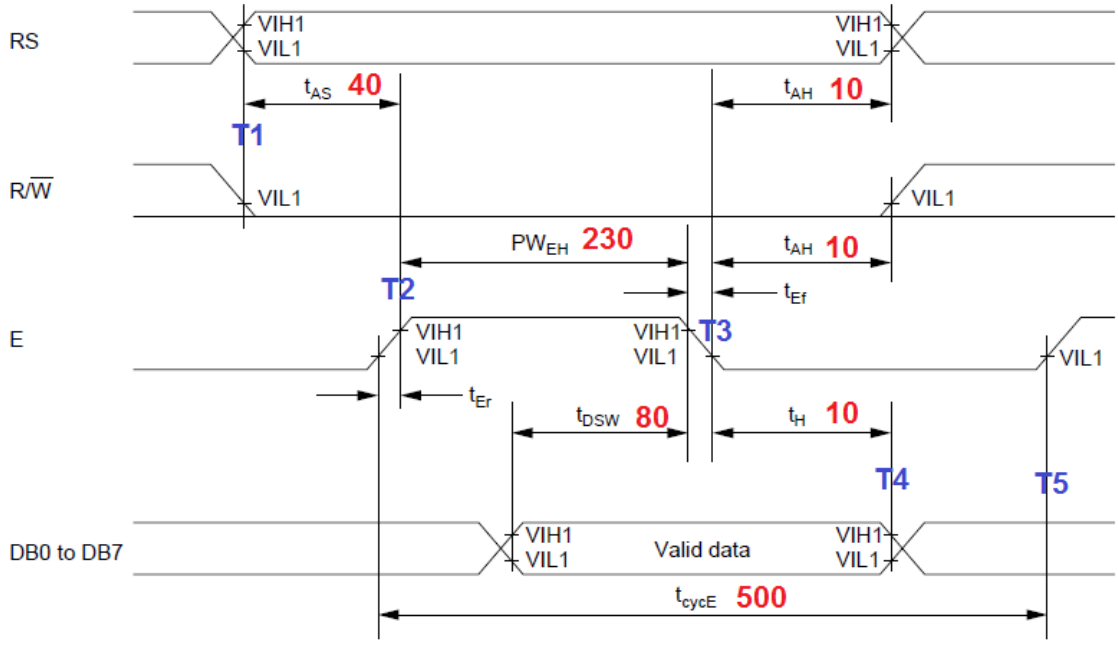


figura 3

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

Pentru a genera timingul din figura 3 trebuie stabilită valoarea logică a semnalelor E și RS. Pentru a activa sau inactiva semnalele E și RS folosiți macro definițiile *setbit* sau *clrbit* din fișierul defs.h

Pașii care trebuie urmați în scrierea codului pentru funcției

void wr_LCDreg(unsigned char vRS, unsigned char data)

sunt:

1. Stabiliți valoarea semnalului RS. Ați ajuns la T1.
2. 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.
3. Așteptați t_{AS}. Valoarea lui t_{AS} se găsește în figura 3. Folosiți WAIT125 de câte ori este nevoie.
4. Activați E.

5. Așteptați t_{PWEH}. Valoarea lui t_{PWEH} se găsește în figura 3. . Folosiți WAIT125 de câte ori este nevoie.
6. Inactivați E.
7. Așteptați T_H. Valoarea lui T_H se găsește în figura 3. . Folosiți WAIT125 de câte ori este nevoie.
8. Ați terminat!

Atenție! La pașii următori accesați registrele controlerului LCD prin intermediul funcției *wr_LCDreg*.

De exemplu, pentru a inițializa LCD-ul la un moment dat 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 DReg 1
```

Folosind aceste declarații, comanda „Clear dispay" se va trimite cu:

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

Astfel codul devine mult mai clar.

Obligatoriu: Scrieți registrele LCD în stilul de mai sus!

Pasul 4: *initLCD*()

A doua 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.

Se scrie funcția **initLCD**. Se scriu în controlerul LCD următoarele instrucțiuni:

- Function set: DL- 8 biţi, N - 2 linii, F - 5x8 dots.
- 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.
- Entry mode set: I/D - increment, S - no shift
- Clear display

Documentația pentru aceste setări se găsește în tabelul 3 și în **HD44780U.pdf**

După fiecare comandă așteptați ca interfața LCD să execute comanda. Timpul de execuție a comenzii depinde de timpul de tipul comenzii și este precizat în tabelul 3, coloana „**Execution Time**". Așteptați cu funcția *wait* un interval de timp cu 25% - 30% mai mare decât timpul din tabel deoarece funcția *wait* nu este foarte exactă. De exemplu, în loc de 37 us folosiți 50 us.

Este obligatoriu să se documenteze valorile biților din fiecare instrucțiune. Fără documentare este imposibil să modificăm codul după o lună sau un an. La orice firmă modificarea ulterioară a codului se face de persoana care a scris codul inițial. Așa că documentarea codului este în primul rând un mesaj de la dumneavoastră din prezent pentru dumneavoastră din viitor!

De exemplu, pentru instrucțiunea „Function set" trebuie să scriem următoarele trei comentarii:

```
//7 6 5  4 3 2 1 0   numar bit
//0 0 1 DL N F – –   semnificație bit  copiată din documantație
//0 0 1  1  ...      valoare programată
```

Pe prima linie va apare numărul bitului din instrucțiune. Pe a doua linie copiem din documentație semnificația fiecărui bit. Pe a treia linie apar valorile alese de cel care scrie codul. Pentru „Function set" biții 7, 6, 5 au întotdeauna valorile ‚0', '0', ‚1', așa că le copiem pe linia a treia. Urmează bitul DL care setează numărul de conexiuni de date (interface data length). Deoarece am folosit 8 pini de date, DL va fi ‚1'. Completați pe a treia linie valorile pentru N și F și apoi folosiți toate valorile de pe linia a treia pentru a scrie registrul (Care registru?) LCD. Vezi ca model „Function set" în IOfn.c.

Procedați la fel pentru celelalte trei instrucțiuni.

Obs. Nu uitați să așteptați după fiecare comandă trimisă către interfața LCD.

Pasul 5: *sysinit* - Inițializarea sistemului

Prima funcție care trebuie apelată este funcția care face inițializarea sistemului. Vom numi această funcție *sysinit*. Funcția este apelată o singură data la începutul lui *main*(). Completați corpul funcției.

Setările care trebuie făcută se referă la direcția și valorile inițiale pentru pin porturile B și D. deoarece nu este posibilă citirea, portul D este tot timpul port de ieșire.

```
DDRD=....
```

Pinii PB7 și PB5 din portul B sunt folosiți pentru a genera semnalele de control ale interfeței LCD. 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.

```
PORTB=...
DDRB=...
```

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 (dar în simulare nu contează).

```
wait(...
initLCD(..
```

Pasul 6: Testare HW+SW

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

Dacă funcțiile au fost scrise corect trebuie să apară cursorul pe linia 1 coloana 1.

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 alimentarea simulatorului. Trebuie procedat în acest fel deoarece controlerul LCD nu are pin de RESET.

Dacă apare cursorul pe prima poziție, apelați profesorul.