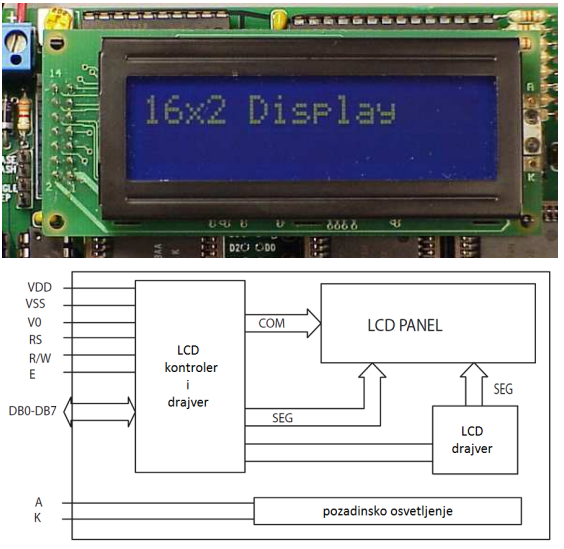
7

Rad sa LCD

Displeji sa tečnim kristalima (eng. *liquid crystal display*, LCD) služe za ispisivanje poruka na minijaturnom ekranu. U ovoj vežbi će biti pi osnovni principi, primeri i zadaci za rad sa LCD koji se realizuju na bazi kontrolera Hitachi HD44780.

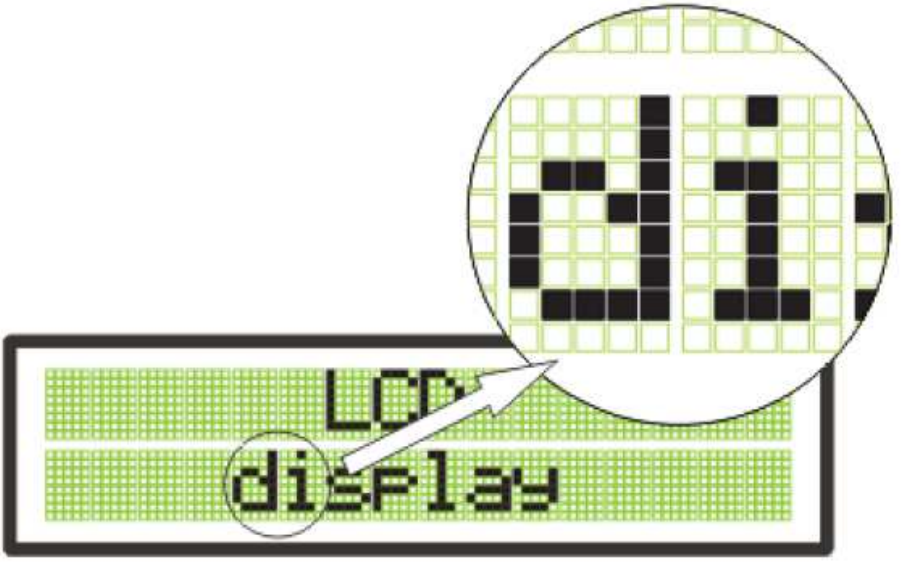
7.1. Uvod o displejima

Fotografija LCD i blok šema su prikazani na slici 7.1.



Slika 7.1. LCD Hitachi HD44780

LCD se sastoji iz 2 reda (može biti i više) sa po 16 polja u kojima se ispisuju karakteri. Svako od ovih polja sastoji se od matrice veličine 5x8 piksela kao što je prikazano na slici 7.2. Broj polja može biti i veći, a i dimenzije karaktera se mogu drugačije podešavati.



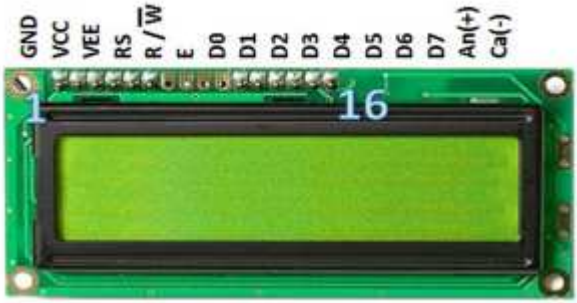
Slika 7.2. Izgled matrice LCD

Displej prikazuje sva slova abecede, grčka slova, znakove interpunkcije, matematičke simbole itd. Pored toga moguće je prikazati i znakove koje korisnik sam projektuje. Takođe poseduje i automatsko pomeranje poruka preko ekrana (pomeranje ulevo i udesno), pojavljivanje kursora, pozadinsko osvetljenje i slično.

Kontrast na ekranu zavisi od napona napajanja i od toga da li se poruke ispisuju u jednom ili dva reda. Zbog toga se na izvod najčešće označen sa Vo ili Vee priključuje promenljivi napon od 0 – Vdd (obično se za ovo koristi trimer potenciometar).

7.1.1. Pinovi displeja

Izgled displeja sa spiskom od 16 pinova se može videti na slici 7.3, a uloga pinova opisana je u narednoj tabeli.



Slika 7.3. Pinovi displeja

Tabela 7.1 Funkcija i opis pinova LCD

Pin	Naziv	Uloga	Logičko stanje	Opis
1.	Vss	Masa	-	0V
2.	Vdd	Napajanje	-	+5V
3.	Vee	Kontrast	-	0 – Vdd
4.	RS	Kontrola rada	0	D0 – D7 se tumače kao komande
			1	D0 – D7 se tumače kao podaci
5.	R/W		0	Upis podataka (iz mikrokontrolera)
			1	Čitanje podataka (u mikrokontroler)
6.	En		0	Onemogućen pristup LCD-u
			1	Normalan rad
			iz 1 u 0	Podaci/komande se prenose u LCD
7.	D0	Podaci / komande	0/1	Bit 0 LSB
8.	D1		0/1	Bit 1
9.	D2		0/1	Bit 2
10.	D3		0/1	Bit 3
11.	D4		0/1	Bit 4
12.	D5		0/1	Bit 5
13.	D6		0/1	Bit 6
14.	D7		0/1	Bit 7 MSB
15.	LED+	Pozadinsko osvetljenje	-	+5V
16.	LED–		-	0V

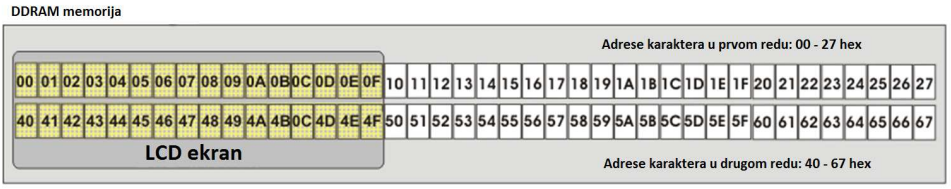
7.1.2. Memorije unutar LCD

Unutar displeja se nalaze 3 memorijska bloka:

1. DDRAM – Display Data RAM,
2. CGROM – Character Generator ROM,
3. CGRAM – Character Generator RAM.

DDRAM memorija

U ovu memoriju smeštaju se karakteri koji treba da budu prikazani na displeju. Veličina ove memorije je dovoljna za smeštanje 80 znakova, a jedan deo ovih lokacija ima direktne veze sa poljima na ekranu. Na slici 7.4 prikazana je adresna mapa DDRAM memorije.



Slika 7.4. Adresna mapa DDRAM memorije

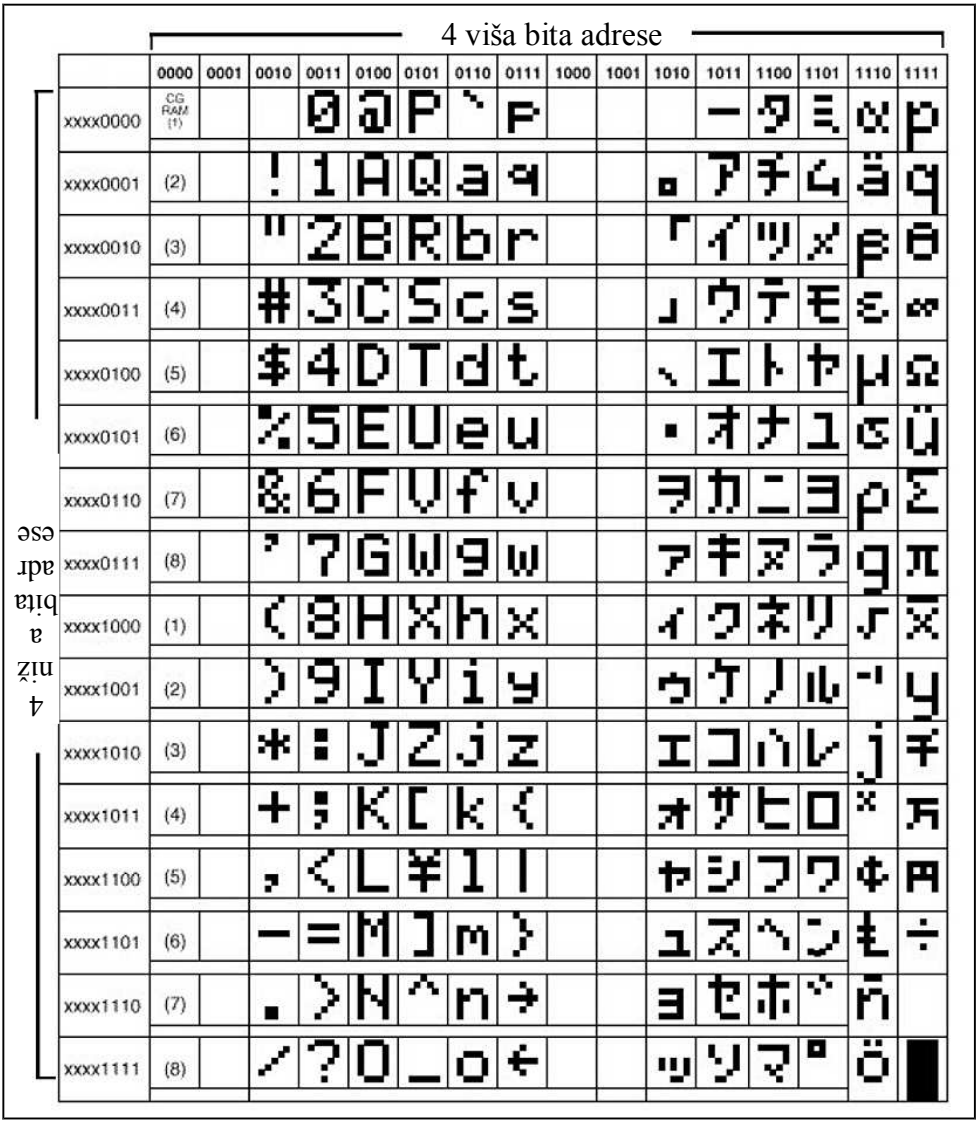
Displej se konfiguriše tako da automatski uvećava adrese (pomeranje udesno) i zatim postavi početnu adresu za poruku koja treba da se ispiše. Nakon toga svi karakteri koji se pošalju linijama D0 – D7 pojaviće se ispisani u vidu poruke s leva u desno. Ako se pošalje više od 16 karaktera oni će i dalje biti memorisani ali neće biti vidljivi. Da bi se prikazali, treba koristiti komandu za pomeranje (*shift*) čime se dobija efekat da poruka „prelazi“ preko ekrana.

Ako se omogući prikazivanje kursora, on će se pojaviti na mestu koje je trenutno adresirano, odnosno, karakteri se pojavljuju na mestu kursora dok se on automatski pomera na sledeće adresirano polje.

Pošto je ova memorija RAM tipa, podaci u nju mogu da se upisuju i da se očitavaju ali se sadržaj nepovratno gubi nestankom napona napajanja.

CGROM memorija

U ovu memoriju fabrički je upisana mapa sa izgledom svih karaktera koje displej može da prikaže, svakom karakteru odgovara jedna memorijska lokacija. Na slici 7.5 prikazana je memorijska mapa CGROM memorije.



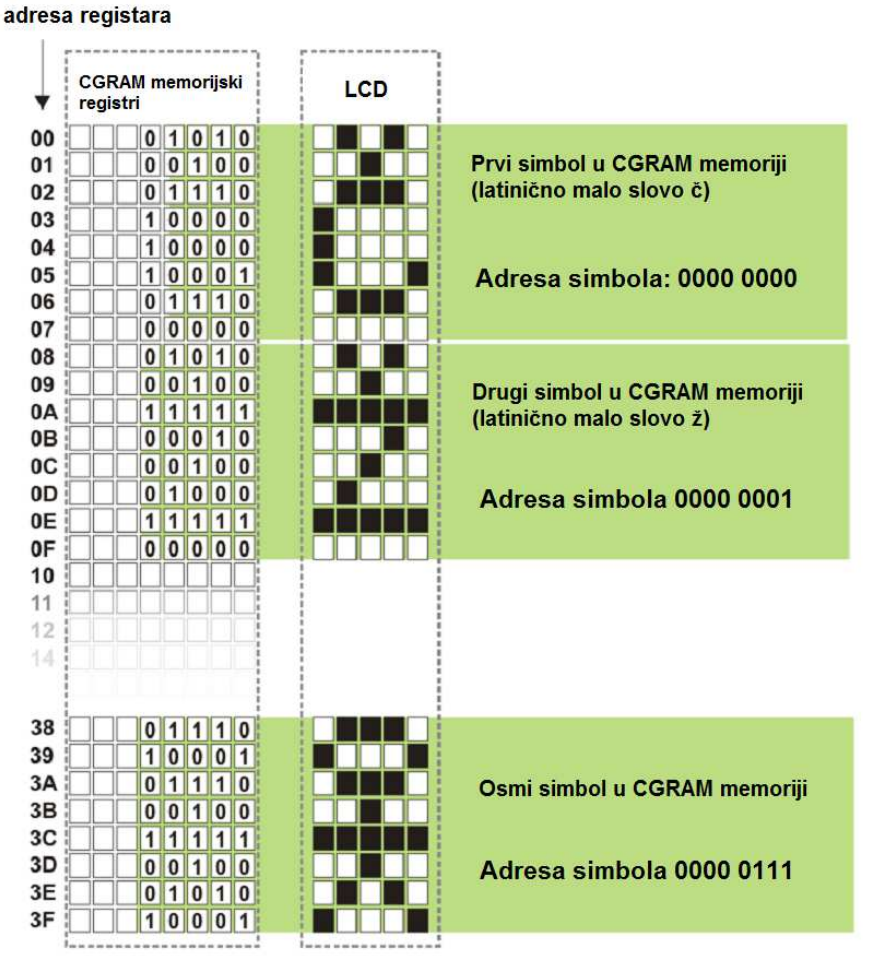
Slika 7.5. Memorijska mapa CGROM memorije

Adrese memorijskih lokacija CGROM-a se poklapaju sa standardnim ASCII vrednostima karaktera. To znači da, ukoliko se u programu koji mikrokontroler izvršava zahteva ispisivanje nekog karaktera, na izlazu će se pojaviti binarna vrednost karaktera. Kada se binarni broj učita u LCD displej, prikazaće se simbol koji se nalazi na memorijskoj lokaciji koja odgovara binarnom broju u CGROMU-u. Ovo važi za sva slova abecede (velika i mala) ali ne i za brojeve.

CGRAM memorija

Pored toga što prikazuje sve standardne karaktere, LCD displej može da prikaže i oznake koje korisnik sâm definiše. To omogućava ispisivanje ćiriličnih fontova kao i drugih simbola koji staju u okvir veličine 5x8 piksela. Sve to omogućava mala RAM memorija (CGRAM) veličine 64 bajta.

Veličina registra u ovoj memoriji je 8 bita ali se koristi samo nižih 5. Logička jedinica u svakom registru predstavlja zatamnjen piksel, dok osam lokacija uzetih zajedno predstavljaju jedan znak. Na slici 7.6 prikazan je primer simbola koje korisnik može sam da definiše.



Slika 7.6. Primer simbola koje korisnik definiše

Simboli se definišu na početku programa, a njihovo prikazivanje se vrši

pozivanjem adrese (prva kolona u CGROM mapi).

7.2. Komande, povezivanje i inicijalizacija LCD

Svi podaci koji se prenose na LCD preko izvoda D0 – D7 biće tumačeni kao komande ili podaci u zavisnosti od logičkog stanja na pinu RS.
RS = 1 – Bitovi D0 – D7 su adrese karaktera koji treba da se prikažu na displeju. Ugrađeni procesor adresira ugrađenu mapu karaktera i prikazuje odgovarajući znak. Mesto pojavljivanja je određeno DDRAM adresom.
RS = 0 – Bitovi D0 – D7 su komande koje određuju način rada displeja.

Spisak svih komandi koje LCD podržava su navedene u tabeli 7.2.

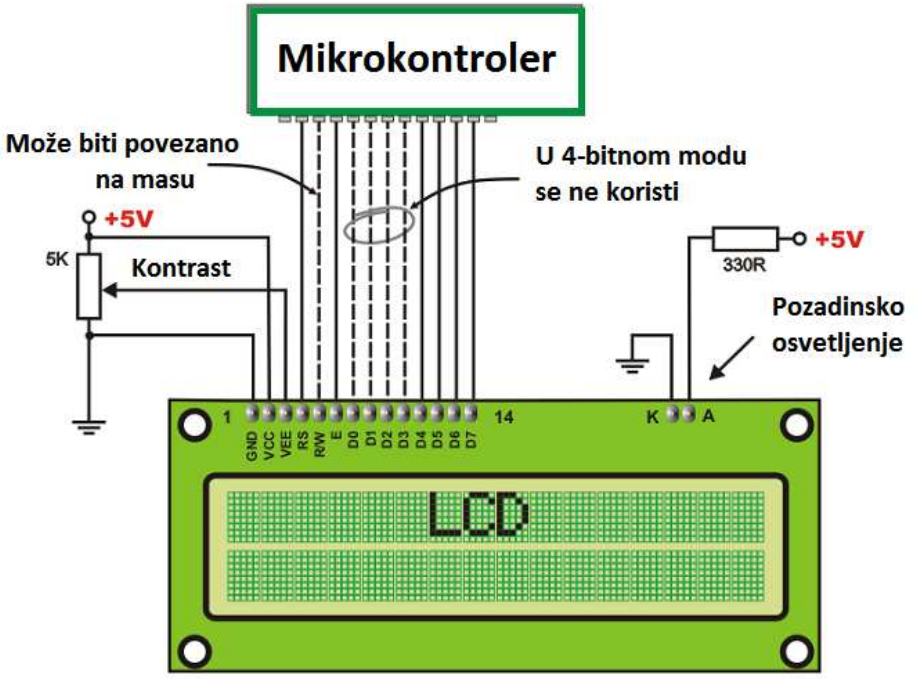
Tabela 7.2. Komande LCD

Komanda	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
Obriši displej	0	0	0	0	0	0	0	0	0	1
Kursor na početak	0	0	0	0	0	0	0	0	1	X
Unos karaktera	0	0	0	0	0	0	0	1	I/D	S
Uključeno/isključeno	0	0	0	0	0	0	1	D	U	B
Pomeranje	0	0	0	0	0	1	D/C	R/L	x	X
Način rada	0	0	0	0	1	DL	N	F	x	x
CGRAM adresa	0	0	0	1	CGRAM adresa					
DDRAM adresa	0	0	1	DDRAM adresa						
Čitanje BUSY flega (BF)	0	1	BF	DDRAM adresa						
Upis u CGRAM ili DDRAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0
Čitanje iz CGRAM-a/DDRAM-a	1	1	D7	D6	D5	D4	D3	D2	D1	D0

I/D 1 = Inkrement
0 = Dekrement
S 1 = Uključen šift registra
0 = Isključen šift registra
D 1 = Uključen displej
0 = Isključen displej
U 1 = Vidljiva linija kursora
0 = Bez linije kursora
B 1 = Kursor treperi
0 = Kursor ne treperi

R/L 1 = Pomeranje udesno
0 = Pomeranje ulevo
DL 1 = 8-bitno povezivanje
0 = 4-bitno povezivanje
N 1 = Ispis u dve linije
0 = Ispis u jednoj liniji
F 1 = Format 5x10
0 = Format 5x8
D/C 1 = Pomeranje displeja
0 = Pomeranje kursora

U zavisnosti od toga koliko se linija koristi za povezivanje sa mikrokontrolerom razlikuju se 8-bitni i 4-bitni način rada LCD. U prvom slučaju podaci se prenose preko izvoda D0 – D7 onako kako je već objašnjeno. U drugom slučaju, za komunikaciju se koriste samo 4 viša bita (D4 – D7) dok se ostali mogu ostaviti nepovezani kao što je prikazano na slici 7.7. U tom slučaju, svaki podatak se LCD-u šalje u dva koraka: prvo se šalju 4 viša bita a zatim se šalju 4 niža bita.



Slika 7.7. Primer povezivanja displeja i mikrokontrolera

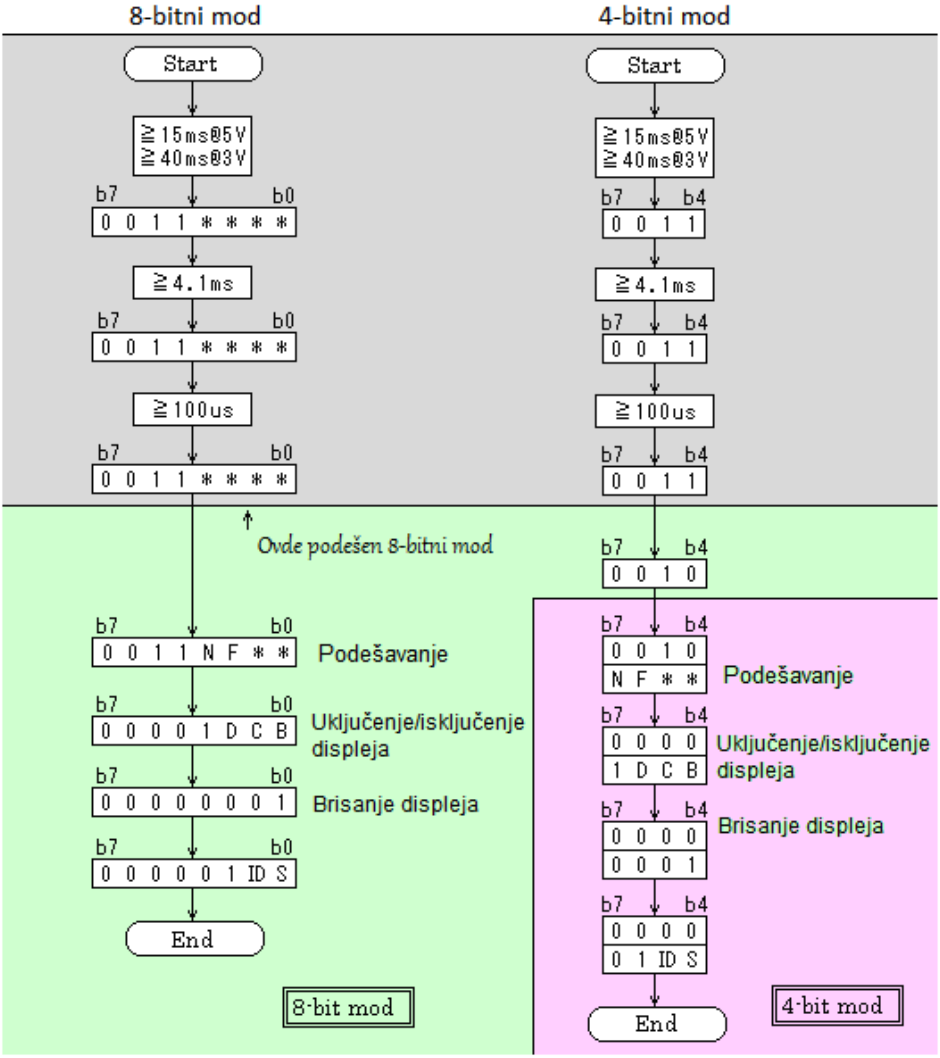
Odmah nakon dovođenja napona napajanja LCD displej se automatski resetuje. Nakon toga displej je spreman za rad i nalazi se u fabrički podešenom načinu rada što znači:

1. Displej je obrisan;
2. Način rada:
- **DL = 1** komunikacija se izvodi 8-bitnom vezom;
- **N = 0** poruke se ispisuju u jednom redu;
- **F = 0** koriste se fontovi formata 5x8 tačaka;

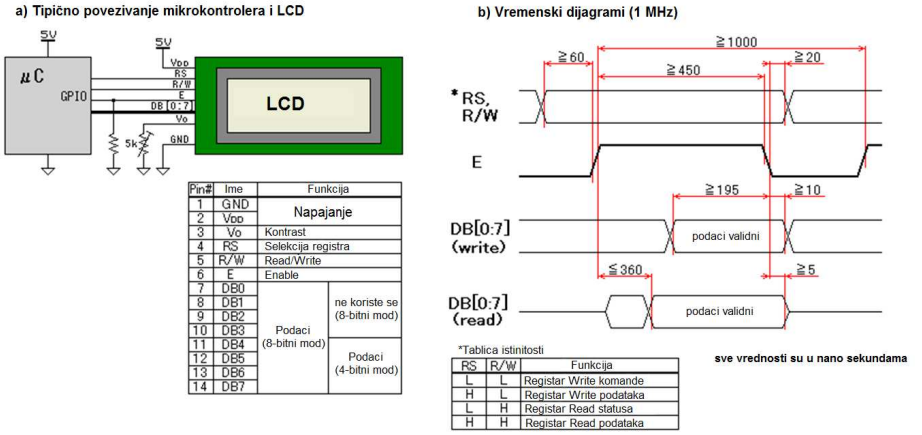
3. Uključeno/isključeno:
- **D = 0** displej je isključen;
- **U = 0** ne vidi se linija kursora;
- **B = 0** isključeno je blinkanje kursora;

4. Unos karaktera:
- **ID = 1** adrese na ekranu se automatski uvećavaju za 1
- **S = 0** isključeno pomeranje (shift)

Algoritam po kome se obavlja inicijalizacija u slučaju 8-bitnog i 4-bitnog povezivanja prikazan je na slici 7.8.



Slika 7.8. Algoritam inicijalizacije LCD



Pin#	Ime	Funkcija	
1	GND	Napajanje	
2	Vdd		
3	Vo	Kontrast	
4	RS	Selekcija registra	
5	R/W	Read/Write	
6	E	Enable	
7	DB0	Podaci (8-bitni mod)	ne koriste se (8-bitni mod)
8	DB1		
9	DB2		
10	DB3		
11	DB4		Podaci (4-bitni mod)
12	DB5		
13	DB6		
14	DB7		

*Tablica istinitosti

RS	R/W	Funkcija
L	L	Registar Write komande
H	L	Registar Write podataka
L	H	Registar Read statusa
H	H	Registar Read podataka

Slika 7.9. Tipični vremenski dijagrami

U ovoj vežbi će biti pokazan rad sa LCD oznake LCD1602 koji se veoma često koristi i spada u red dvorednih LCD sa mogućim prikazom do 16 karaktera po redu. U ovu grupu displeja spadaju mnogi LCD koji su bazirani na Hitachi HD44780 kontroleru ili odgovarajućem ekvivalentu. U našem slučaju je u pitanju SPLC780D kontroler koji radi na 3.3V i ima plavo pozadinsko osvetljenje. Za detalje oko ovog SPLC780D kontrolera pogledati njegovu dokumentaciju na sajtu. Pored 3.3 voltnog napajanja često su u upotrebi i 5 voltni, kao i displeji bez pozadinskog osvetljenja.

Radi lakšeg rada sa displejom koristićemo *wiringPi* biblioteku koja ima već predefinisane funkcije za rad sa LCD displejima. U suprotnom bi trebalo realizovati algoritme za inicijalizaciju, upis i sl. U svakom programu na početku treba dodati:

```
#include <wiringPi.h>
#include <lcd.h>
```

7.2.1. Funkcije za rad sa LCD-ovima

Funkcija za inicijalizaciju displeja **lcdInit**

```
int lcdInit(int rows, int cols, int bits, int rs, int strb,
          int d0, int d1, int d2, int d3, int d4, int d5,
          int d6, int d7)
```

`rows` – broj redova lcd displeja
`cols` – broj kolona
`bits` – 4/8 bitni interfejs
`rs` – lokacija RS pina
`strb` – lokacija strobe pina
`d0-d7` – lokacije pinova za prenos podataka

Funkcija vraća broj (*handle*) koji se koristi posle u ostalim funkcijama. Ako vrati -1 onda je došlo do neke greške.

primer: **lcdInit** (2, 16, 4, RS_PIN, E_PIN, d0_PIN, d1_PIN, d2_PIN, d3_PIN, d4_PIN, d5_PIN, d6_PIN, d7_PIN)

gde su konstante ?_PIN konkretni pinovi koji se koriste.

Funkcija za postavljanje kursora na početak **lcdHome**
lcdHome (int handle)

Funkcija za uključenje/isključenje:

1. displeja: **lcdDisplay** (int fd, int state)
2. kursora: **lcdCursor** (int fd, int state)
3. treptanja kursora: **lcdCursorBlink** (int fd, int state)

Inicijalno je displej uključen, a kursor i njegovo treptanje su isključeni.

Funkacija za brisanje displeja **lcdClear**
lcdClear (int handle)

Funkcija za pozicioniranje kursora na x,y poziciju **lcdPosition**
lcdPosition (int handle, int x, int y)

Funkcije za postavljanje karaktera, stringa i formatiranog stringa na displeju su:
lcdPutchar (int handle, uint8_t data)
lcdPuts (int handle, char *string)
lcdPrintf (int handle, char *message, …)

Funkcija za postavljanje korisničkih karaktera:
lcdCharDef (int handle, int index, unsigned char data [8])

Ova funkcija omogućava da se redefinišu neki od 8 karaktera koje korisnik može da menja. Niz data[8] od 8 bajtova čini skup piksela počevši od vršne linije do donje. Karakteri su u rezoluciji 5x8, tako da se unutar svakog bajta koristi samo nižih 5 bita. Ispis pojedinih karaktera se može vršiti pozivom **lcdPutchar** funkcije.

NAPOMENA

Pin RW na displeju obavezno vezati na GND (na DVK512 je to hardverski rešeno) da se ne može dogoditi da LCD šalje podatke prema RPi koji je 3.3 V, a neki displeji su 5 V! Pri radu sa displejom se koristi i Vo pin koji je povezan na trimer potenciometar i koji služi za regulaciju kontrasta. Kod 5 V displeja Vo pin vezati na GND.

7.3. Primeri

7.3.1. Primer 1

U primeru koji sledi se podrazumeva displej u 4-bitnom modu povezan na kontaktnu letvicu za LCD1602 na DVK512 pločici na sledeći način:

LCD	RS	E	D4	D5	D6	D7
DVK512	P3	P_CLK	P4	P_MOSI	P_MISO	P6
***wiringPi* pin#**	3	14	4	12	13	6

Pogledati šemu DVK512 pločice datu na sajtu radi boljeg razumevanja povezivanja.

```
// lcd1602.c
// kompajlirati sa -lwiringPi -lwiringPiDev

#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <wiringPi.h>
#include <lcd.h>

// dodela vrednosti za konkretne pinove
// prema gornjoj tabeli i semi DVK512
const int RS = 3;
const int EN = 14;
const int D0 = 4;
const int D1 = 12;
const int D2 = 13;
const int D3 = 6;

int main(){

int lcd_h;

     if (wiringPiSetup() < 0){
     fprintf (stderr, "Greška pri inicijalizaciji:
               %s\n", strerror (errno)) ;
     return 1 ;
     }

     lcd_h = lcdInit(2, 16, 4, RS, EN, D0, D1, D2,
                         D3, D0, D1, D2, D3);
```

```
    lcdPosition(lcd_h, 0,0);
    lcdPrintf(lcd_h,"Displej sa 16 ch");
    lcdPosition(lcd_h, 0,1);
    lcdPrintf(lcd_h, "u 2 reda");

    delay(2000);

    lcdClear(lcd_h);
}
```

7.3.2. Primer 2

U ovom primeru će biti dat akcenat na prikaz specijalnih karaktera koje korisnik sam definiše.

```
#include <signal.h>
#include <stdlib.h>
#include <stdio.h>
#include <wiringPi.h>
#include <lcd.h>
#include <string.h>
#include <time.h>
#include <unistd.h>

char level0[8] = { 0b00000, 0b00000, 0b00000, 0b00000,
                   0b00000, 0b00000, 0b00000, 0b11111};
char level1[8] = { 0b00000, 0b00000, 0b00000, 0b00000,
                   0b00000, 0b00000, 0b11111, 0b11111};
char level2[8] = { 0b00000, 0b00000, 0b00000, 0b00000,
                   0b00000, 0b11111, 0b11111, 0b11111};
char level3[8] = { 0b00000, 0b00000, 0b00000, 0b00000,
                   0b11111, 0b11111, 0b11111, 0b11111};
char level4[8] = { 0b00000, 0b00000, 0b00000, 0b11111,
                   0b11111, 0b11111, 0b11111, 0b11111};
char level5[8] = { 0b00000, 0b00000, 0b11111, 0b11111,
                   0b11111, 0b11111, 0b11111, 0b11111};
char level6[8] = { 0b00000, 0b11111, 0b11111, 0b11111,
                   0b11111, 0b11111, 0b11111, 0b11111};
char level7[8] = { 0b11111, 0b11111, 0b11111, 0b11111,
                   0b11111, 0b11111, 0b11111, 0b11111};

#define COLUMNS 16
#define LCD_RS  3
#define LCD_E   14
#define LCD_D4  4
#define LCD_D5  12
#define LCD_D6  13
#define LCD_D7  6
```

```
void memory(void);
void volume(void);
void scrollText(void);

char message[] = "Pozdravče!";
int count =0;
int j = 0;
FILE *mem_file;
char *temp;
int lcd;

int main()
{
     wiringPiSetup();

     if(lcd = lcdInit(2, 16, 4, LCD_RS, LCD_E, LCD_D4,
                    LCD_D5, LCD_D6, LCD_D7, 0, 0, 0, 0)){
     printf ("lcdInit nije uspeo! \n");
     return -1 ;
     }

     int uptimeTimer;

     while(1){
                lcdClear (lcd);
                volume();
                sleep(1);
                memory();
                sleep(4);
                lcdClear (lcd);
                sleep(1);
                scrollText();
     }
}

void memory(void)
{
     char MemTotal[35];
     char MemFree[35];
     char total[35];
     char free[35];

     lcdClear (lcd);

     mem_file=fopen("/proc/meminfo","r");

     if(NULL != mem_file)
     {
     fscanf(mem_file,"%*s%s%*s", MemTotal);
     fscanf(mem_file,"%*s%s%*s", MemFree);
     printf("\x1B[2J");//brisanje ekrana
```

```
      lcdPosition(lcd,0,0);
      lcdPrintf(lcd,"MemTotal-%sk",MemTotal);
      lcdPosition(lcd,0,1);
      lcdPrintf(lcd,"MemFree -%sk",MemFree);
      fclose(mem_file);
      }
      else
      {
      printf("Otvaranje fajla \"/proc/meminfo\" nije
      uspelo!\n");
      }
}

void volume(void)
{
        //Definicija specijalnih karaktera za Volume
        lcdCharDef  (lcd, 0, level0);
        lcdCharDef  (lcd, 1, level1);
        lcdCharDef  (lcd, 2, level2);
        lcdCharDef  (lcd, 3, level3);
        lcdCharDef  (lcd, 4, level4);
        lcdCharDef  (lcd, 5, level5);
        lcdCharDef  (lcd, 6, level6);
        lcdCharDef  (lcd, 7, level7);

        lcdClear (lcd);

        int i;
        lcdPosition (lcd, 9,1);
        lcdPuts (lcd, ":Volume");
        for (i = 0; i < 8; i++){
                lcdPosition (lcd, i, 1);
                lcdPutchar  (lcd, i);
                usleep(400000);
        }
}

void  scrollText(void)
{
        int i,n;
        int h;
        int tempSpace = 0;
        char scrollPadding[] = "                    ";

        int messageLength =  strlen  (scrollPadding) +
                                strlen(message);
        for (n=0;n<messageLength;n++){
                    h = COLUMNS;
                    usleep(300000);
                    printf("\x1B[2J");//brisanje ekrana
                    if ( j > messageLength ) j = 0;
```

```
                    for (i = 0; i < j; i ++){
                        scrollPadding[h-j] = message[ i];
                h++;
                }
                lcdPosition(lcd,0,0);
                lcdClear (lcd);
                lcdPrintf(lcd,"%s",scrollPadding);
                j++;
        }
}
```

7.4. Zadaci

7.4.1. Zadatak 1

Napisati program u C-u za RPi koji očitava temperaturu sa senzora i prikazuje na displeju.

7.4.2. Zadatak 2

Napisati program u C-u za RPi koji omogućava da se korišćenjem daljinskog upravljača pomera kursor na displeju u sva 4 pravca (uz rotaciju kada dođe do kraja).

7.4.3. Zadatak 3

Napisati odgovarajući program u C-u za RPi koji omogućava ispis nekih ćiriličnih karaktera koristeći CGRAM memoriju LCD-a na bazi Primera 2.

7.4.4. Zadatak 4

Napraviti biblioteku za rad sa LCD koja je bazirana na BCM2835. Ovaj zadatak je kompleksan i u nivou je težine projekta.