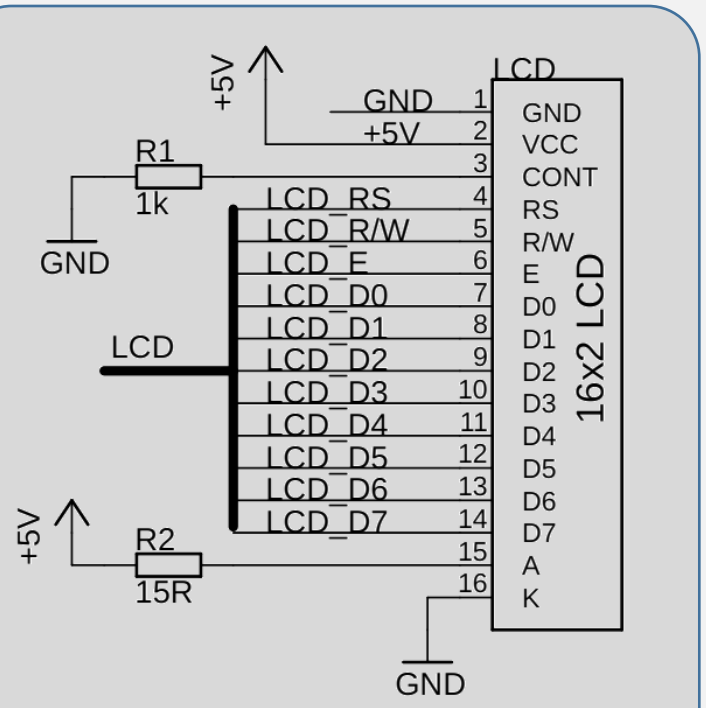
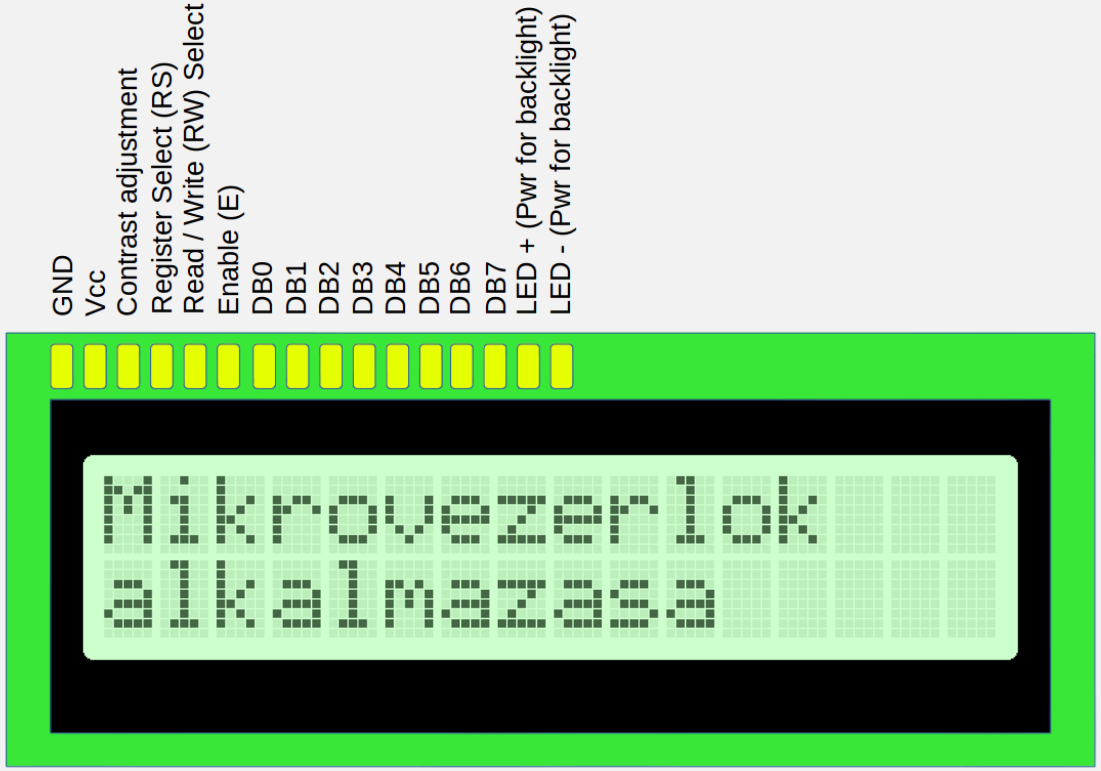
LCD kijelző alkalmazása

LCD modul bekötése

- µMogi2 panelon egy Hitachi HD44780 vezérlővel egybeintegrált 2 soros, 16 karakteres alfanumerikus LCD kijelző található



HD44780 kompatibilis LCD modul

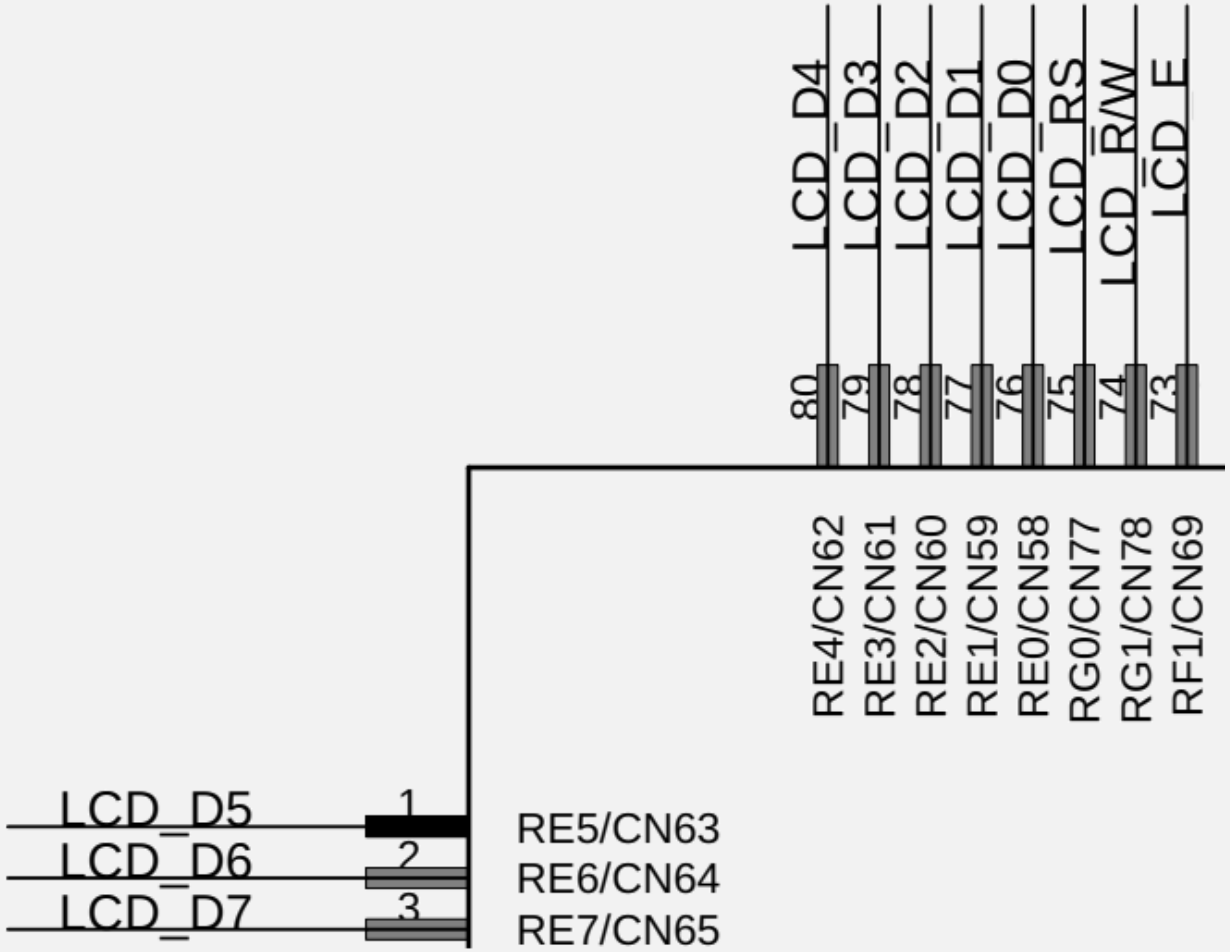


LCD modul bekötése

- Az LCD vezérlő IC párhuzamos porton kommunikál a mikrovezérlővel:
  - D0 - D7 ⇨ RE0 – RE7
  - RS ⇨ RG0
  - R/W ⇨ RG1
  - E ⇨ RF1

- uMOGI2 panel bekötése
  - teljes 8 bites üzemmód
  - adatok megérkezésének visszaolvasása (D7)

```
#define LCD_DATA    LATE // bit 0..7
#define LCD_RS      _LATG0
#define LCD_RW      _LATG1
#define LCD_E       _LATF1
#define LCD_BF      _RE7
```



Az LCD kijelző lábainak értelmezése:

Jelvezeték	Bitek száma	Adatirány (LCD felöl)	Funkció
RS	1	bemenet	Regiszter kiválasztás: 0: utasításregiszter íráskor, cím számláló olvasáskor 1: adatregiszter (íráskor/olvasáskor)
R/W	1	bemenet	Művelet kiválasztás: 0: Írás 1: Olvasás
E	1	bemenet	Az írási/olvasási folyamat indítása
D4..D7	4	kimenet / bemenet	Az adat felső négy bitje. Az adatátvitelben használt ki-, vagy bemenet. A DB7 láb lehet foglaltság jelző (Busy Flag) is.
D3..D0	4	kimenet/ bemenet	Az adat alsó négy bitje. Négy bites kommunikációban nem használt.

Az LCD modul használata

- A modult 4 vagy 8 bites adatszélességgel lehet használni.
  - A 4 bites üzemmódban a lábak száma csökkenthető, de két adatkiküldés/adatbeolvasás szükséges.
- A modult a bekötésének megfelelően kell inicializálni.
- A működtetés lehetséges időzítéssel, vagy a Busy Flag állapotának visszaolvasásával.
- Ha az R/W láb GND-re van kötve, akkor csak írni lehet a modult és időzítéssel működtethető, viszont 1 lábbal kevesebb szükséges.
- Az adatokat kiküldjük az adatvezetékre és egy Strobe jellel jelezzük a modulnak, hogy új adat vagy utasítás érkezett.
- A Strobe jel: az E lábat először magas, majd kis várakoztatás után alacsony állapotba vezéreljük

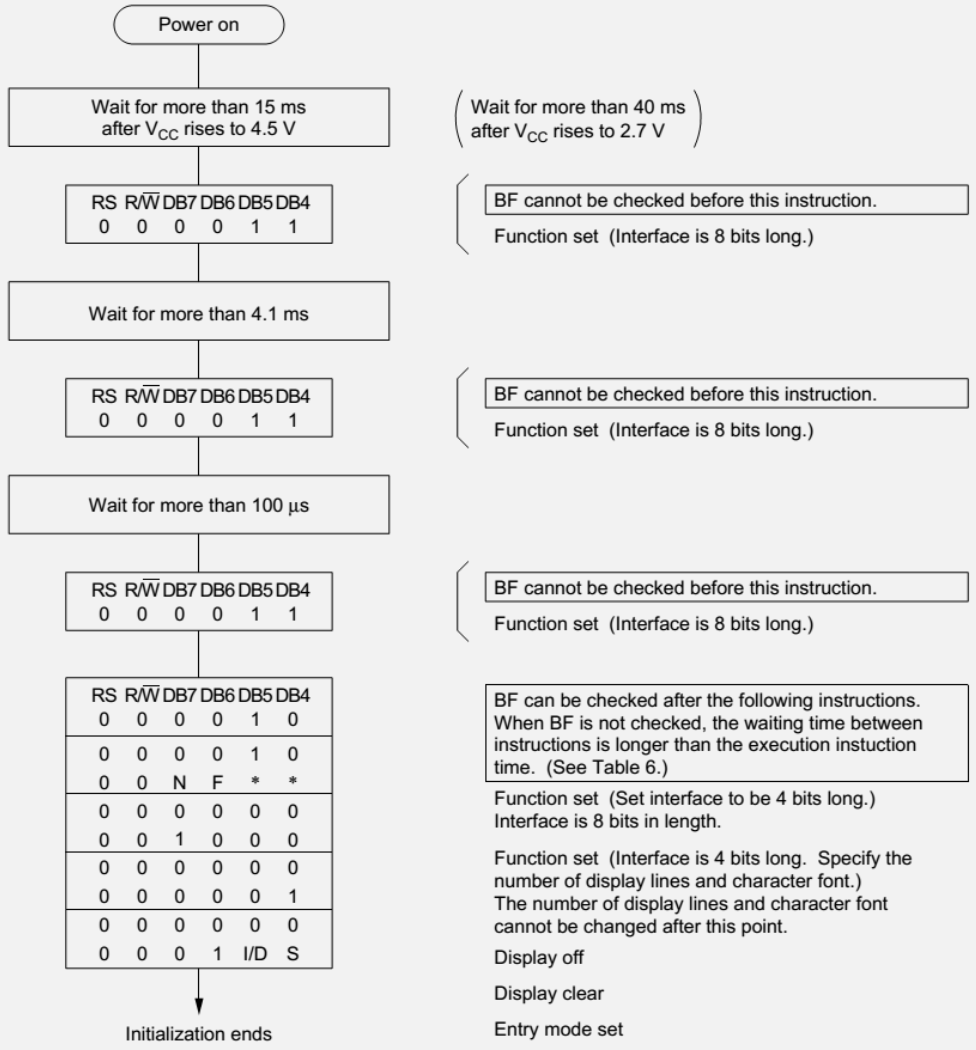
HD44780 utasításkészlet

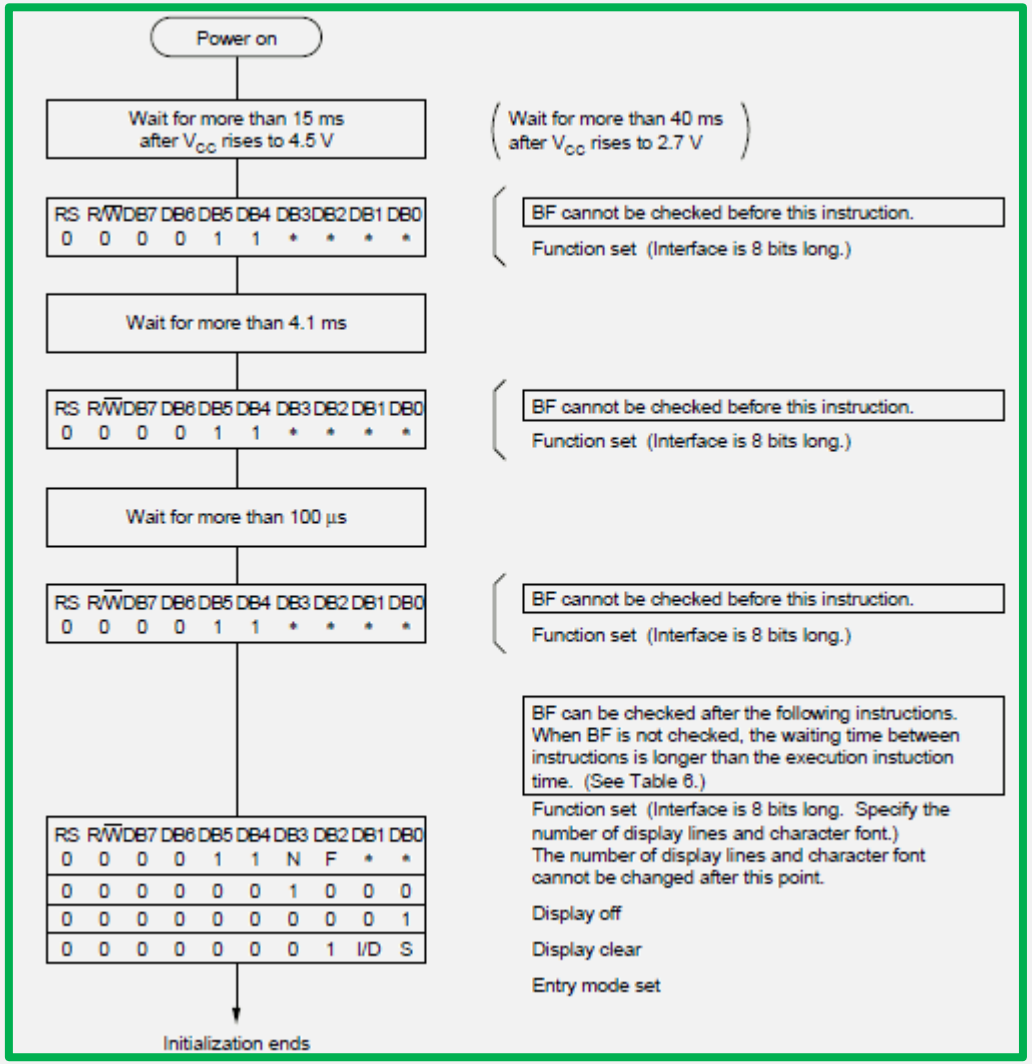
Instruction	Code										Description	Execution time (max) (when f_{cp} = 270 kHz)
	RS	R/W	B7	B6	B5	B4	B3	B2	B1	B0		
Clear display	0	0	0	0	0	0	0	0	0	1	Clears display and returns cursor to the home position (address 0).	1.52 ms
Cursor home	0	0	0	0	0	0	0	0	1	*	Returns cursor to home position. Also returns display being shifted to the original position. DDRAM content remains unchanged.	1.52 ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction (I/D); specifies to shift the display (S). These operations are performed during data read/write.	37 µs
Display on/off control	0	0	0	0	0	0	1	D	C	B	Sets on/off of all display (D), cursor on/off (C), and blink of cursor position character (B).	37 µs
Cursor/display shift	0	0	0	0	0	1	S/C	R/L	*	*	Sets cursor-move or display-shift (S/C), shift direction (R/L). DDRAM content remains unchanged.	37 µs
Function set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL), number of display line (N), and character font (F).	37 µs
Set CGRAM address	0	0	0	1	CGRAM address						Sets the CGRAM address. CGRAM data are sent and received after this setting.	37 µs
Set DDRAM address	0	0	1	DDRAM address							Sets the DDRAM address. DDRAM data are sent and received after this setting.	37 µs
Read busy flag & address counter	0	1	BF	CGRAM/DDRAM address							Reads busy flag (BF) indicating internal operation being performed and reads CGRAM or DDRAM address counter contents (depending on previous instruction).	0 µs
Write CGRAM or DDRAM	1	0	Write Data								Write data to CGRAM or DDRAM.	37 µs
Read from CG/DDRAM	1	1	Read Data								Read data from CGRAM or DDRAM.	37 µs

Instruction bit names —

I/D - 0 = decrement cursor position, 1 = increment cursor position; **S** - 0 = no display shift, 1 = display shift; **D** - 0 = display off, 1 = display on; **C** - 0 = cursor off, 1 = cursor on; **B** - 0 = cursor blink off, 1 = cursor blink on ; **S/C** - 0 = move cursor, 1 = shift display; **R/L** - 0 = shift left, 1 = shift right; **DL** - 0 = 4-bit interface, 1 = 8-bit interface; **N** - 0 = 1/8 or 1/11 duty (1 line), 1 = 1/16 duty (2 lines); **F** - 0 = 5×8 dots, 1 = 5×10 dots; **BF** - 0 = can accept instruction, 1 = internal operation in progress.

Inicializálás 4 vagy 8 bites üzemmódban





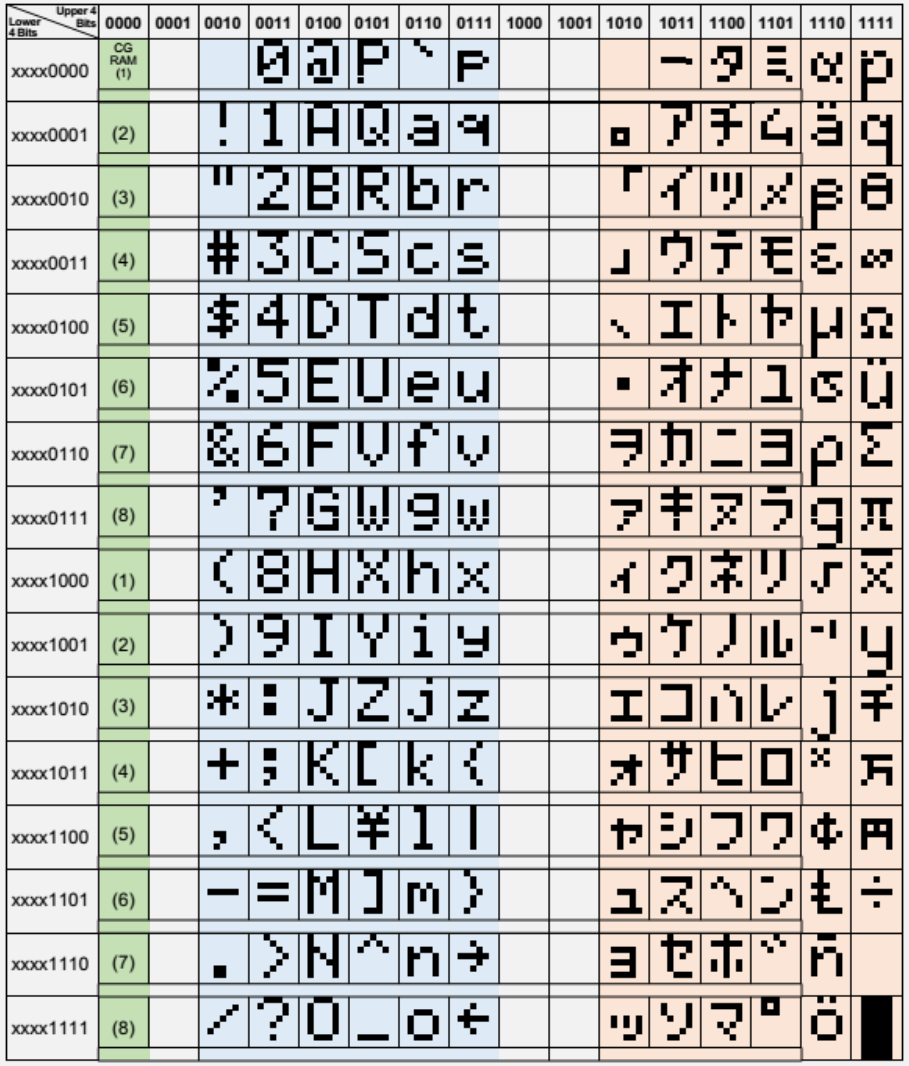
Inicializálás 4 vagy 8 bites üzemmódban

A gyakorlaton az alábbi beállításokat használjuk:

- Function Set:
  - Két sor, 4 bites üzemmód, 5x8 pont — 0x28
  - Két sor, 8 bites üzemmód, 5x8 pont — 0x38
- Entry Mode Set
  - Automatikus inkrementálás, nem lépteti a kijelzőt — 0x06
- Display control
  - Kijelző bekapcsolása, kurzor és villogás kikapcsolva — 0x0C
- Cursor/Display shift
  - Display automatikus mozgatása balra — 0x18
- Clear Display
  - Törli a kijelzőt — 0x01

Az LCD modul karaktertáblája

- Soronként 40 karakter tárolható a Display Data RAM (DDRAM)–ban
- Character Generator ROM (CGROM)-ban találhatóak a felhasználható rajzolatok (5x7, vagy 5x8)
- Az általános karakterek az ASCII kódjuk alapján kerültek be a ROM-ba
- 8 db karakter szabadon definiálható a Character Generator RAM (CGRAM)-ban



Inicializálás 8 bites üzemmódban

```
void lcdInit(){
    // az LCD vezérlő vonalainak beállítása
    TRISE &= 0xff00; // D0-D7 kimenet
    _TRISG0 = 0;     // RS kimenet
    _TRISG1 = 0;     // RW kimenet
    _TRISF1 = 0;     // E kimenet
    LCD_RS = 0;      // RS alacsony
    LCD_RW = 0;      // RW alacsony
    LCD_E = 0;       // E alacsony

    DELAY_MS(50);    // vár az eszköz beállására

    lcdPutCmd(0x38); // 2 soros display, 5x8 karakter
    lcdPutCmd(0x08); // display off
    lcdPutCmd(0x01); // képernyő törlése, kurzor alaphelyzetbe állítás
    lcdPutCmd(0x06); // automatikus inkrementálás, nem lépteti a kijelzőt
    lcdPutCmd(0x0C); // a display bekapcsolása; kurzor és villogás kikapcsolva

    DELAY_MS(3);
}
```

Adat vagy utasítás kiküldése foglaltság figyelésével

```
// Adat vagy utasítás küldése, c adat , rs 0 - utasítás, 1 - adat
void lcdWrite(uint8_t c, uint8_t rs){
    uint8_t BF;       // BusyFlag (foglaltság) figyelése
    TRISE |= 0x00ff; // lábak bemenetek
    LCD_RW = 1;       // beolvasás
    LCD_RS = 0;       // utasítás
    LCD_DATA &= 0xff00;
    do {
        LCD_E = 1;    // E magas
        Nop(); Nop(); Nop(); BF = LCD_BF;
        LCD_E = 0;    // E alacsony
    } while (BF);

    TRISE &= 0xff00; // lábak kimenetek
    LCD_RW = 0;       // írás
    LCD_RS = rs ? 1 : 0;       // adat vagy parancs
    LCD_DATA = (LCD_DATA & 0xff00) | c;    // 8 bit küldése
    LCD_E = 1;        // E magas
    Nop(); Nop(); Nop();
    LCD_E = 0;        // E alacsony
}
```

LCD makrók és karakterlánc kiküldése

```
// LCD makrók
#define lcdPutChar(c) lcdWrite(c, 1)    // karakter küldése
#define lcdPutCmd(d)  lcdWrite(d, 0)    // utasítás küldése
#define lcdClear()    lcdWrite(0x01,0)  // LCD törlése
#define lcdGoHome()   lcdWrite(0x02,0)  // LCD küldése az 1. sorba
#define lcdGo2Row()   lcdWrite(0xC0,0)  // LCD küldése a 2. sorba
```

```
// kiír egy karakterfüzért az LCD-re
void lcdPutStr(char *s){
  while (*s) {
    char c = *s;
    if (c == '\n')
      lcdGo2Row();        // kurzor mozgatása a második sor elejére
    else
      lcdPutChar(c);      // karakter kiíratása
    s++;
  }
}
```

LCD használata

```
#include <stdio.h>  // sprintf miatt
#include <stdlib.h> // malloc
#include <string.h> // memset
```

```
lcdInit();                      // LCD modul inicializálása
lcdPutStr("HELLO\nWORLD");      // szöveg kiíratása
DELAY_MS(2000);                 // várakozás
```

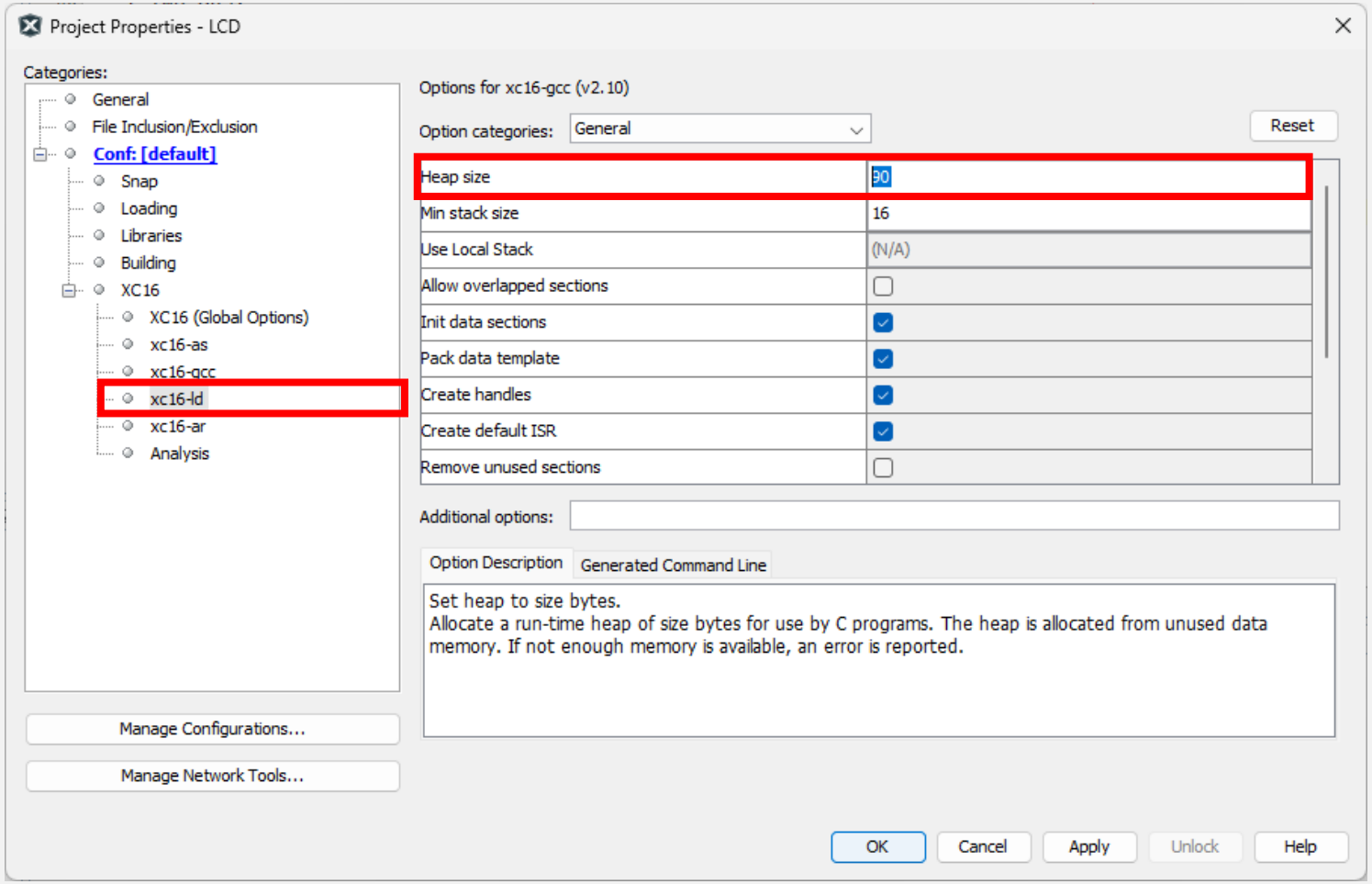
```
char LCD[80];                   // statikus karaktertömb
```

```
char* LCD;                      // dinamikus karaktertömb
LCD = (char *)malloc(80);       // a heap size méretét be kell állítani
LEDR = !LCD;                    // teszt – ha nincs elég hely a foglaláshoz, akkor a piros led kigyullad
```

```
int nr = 124;
double volt = 3.3;
memset(LCD, 0x00, 80);          // LCD kiürítése
sprintf(LCD, "Nr:%i\nVolt:%.2f", nr, volt);  // számok konvertálása karaktertömbbe
lcdClear();                     // LCD törlése
lcdPutStr(LCD);                 // szöveg kiküldése az LCD-re
```

Heap size beállítása a dinamikus memóriafoglaláshoz

- File -> Project Properties

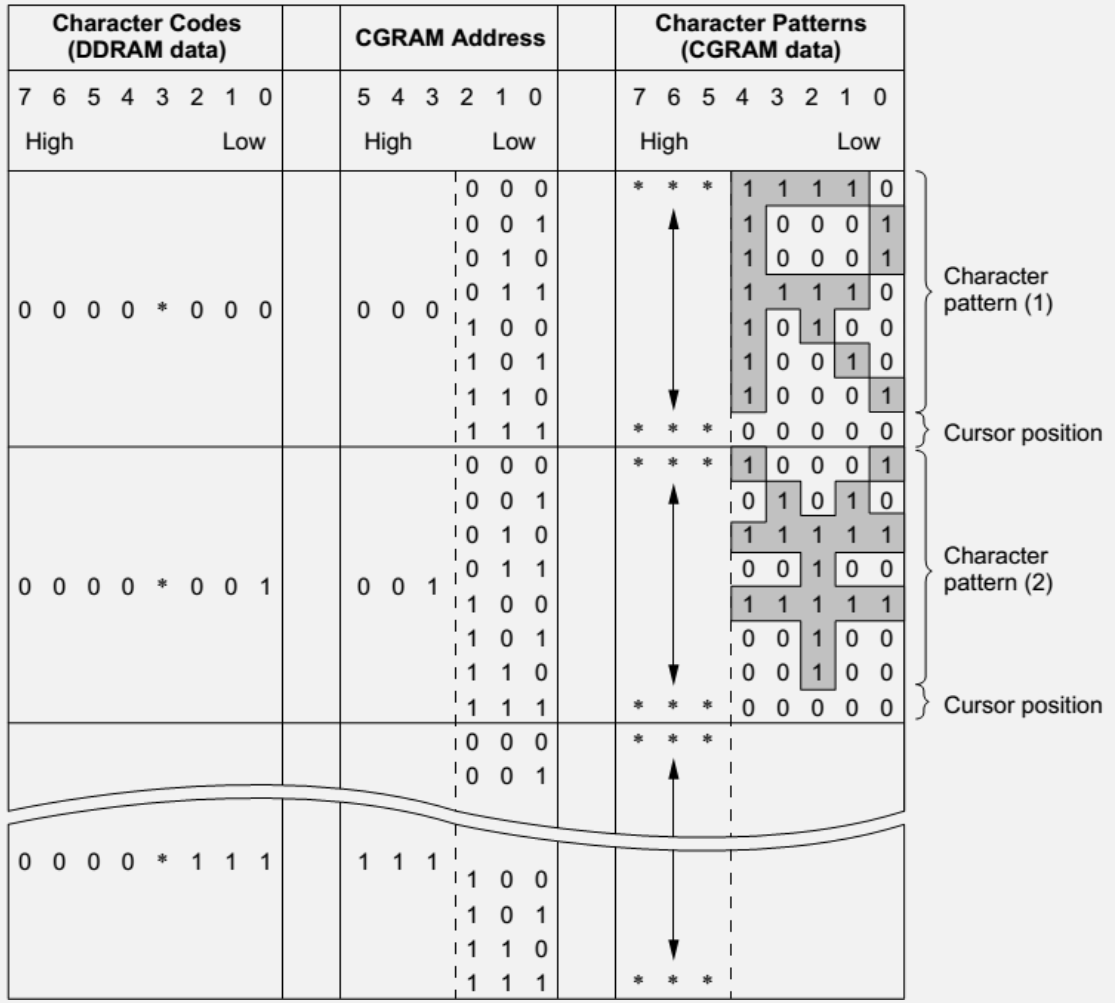


Saját karakter készítése

- Egy rajzolatot kell betölteni a CGRAM-ba
  http://www.quinapalus.com/hd44780udg.html

```
// magyar ékezetes karakterek
const unsigned char hu_char[] = {
0x02,0x04,0x0E,0x01,0x0F,0x11,0x0F,0x00,     // á
0x02,0x04,0x0E,0x11,0x1F,0x10,0x0E,0x00,     // é
0x02,0x04,0x0C,0x04,0x04,0x04,0x0E,0x00,     // í
0x02,0x04,0x0E,0x11,0x11,0x11,0x0E,0x00,     // ó
0x02,0x04,0x11,0x11,0x11,0x13,0x0D,0x00,     // ú
0x0A,0x00,0x11,0x11,0x11,0x13,0x0D,0x00,     // ü
0x05,0x0A,0x11,0x11,0x11,0x13,0x0D,0x00,     // ű
0x05,0x0A,0x0E,0x11,0x11,0x11,0x0E,0x00      // ő
};
```

A Projectet a magyar ékezetes betűk miatt windows-1250-es karakterkódulásúra kell átállítani!



Saját karakterek betöltése

```
// magyar ékezetes karakterek feltöltése a CGRAM-ba
void lcdLoadHuChars(void) {
    int i;
    lcdPutCmd(0x40);              // kurzor a CGRAM elejére (0. char)
    for(i=0; i<64; i++) {
      lcdPutChar(hu_char[i]);     // definiálható karakterek feltöltése
    }                             // ékezetes karakterekkel
    lcdPutCmd(0x80);              // kurzor vissza, a DDRAM elejére
}
```

```
lcdInit();                        // LCD modul inicializálása
lcdLoadHuChars();                 // magyar karakterek betöltése
lcdPutStr("ékezetes karakterek: árvíztükörfúrógép ");    // szöveg kiíratása

while(1)
{
     DELAY_MS(1000);
     lcdPutCmd(0x18);             // LCD képernyő léptetése balra
}
```

Saját karakterek használata

```
// kiír egy karakterfüzért az LCD-re
void lcdPutStr(char *s){
  while (*s) {
    char c = *s;
    switch(c) {           // magyar karakterek cseréje a feltöltöttre
      case 'á':  c = 0x00; break;
      case 'é':  c = 0x01; break;
      case 'í':  c = 0x02; break;
      case 'ó':  c = 0x03; break;
      case 'ú':  c = 0x04; break;
      case 'ü':  c = 0x05; break;
      case 'ű':  c = 0x06; break;
      case 'ő':  c = 0x07; break;
      case 'ö':  c = 0xEF; break;    }
    if (c == '\n')
      lcdGo2Row();        // kurzor mozgatása a második sor elejére
    else
      lcdPutChar(c);      // karakter kiíratása
    s++;
  }
}
```