Versione N° 1.0

LaurTec

ATmega 2560

Corso di programmazione per microcontrollori Atmel

Autore : *Marcello Pinna*

ID: UT0005-IT

Copyright © 2020 Mauro Laurenti

INFORMATIVA

Come prescritto dall'art. 1, comma 1, della legge 21 maggio 2004 n.128, l'autore avvisa di aver assolto, per la seguente opera dell'ingegno, a tutti gli obblighi della legge 22 Aprile del 1941 n. 633, sulla tutela del diritto d'autore.
Tutti i diritti di questa opera sono riservati. Ogni riproduzione ed ogni altra forma di diffusione al pubblico dell'opera, o parte di essa, senza un'autorizzazione scritta dell'autore, rappresenta una violazione della legge che tutela il diritto d'autore, in particolare non ne è consentito un utilizzo per trarne profitto.
La mancata osservanza della legge 22 Aprile del 1941 n. 633 è perseguibile con la reclusione o sanzione pecuniaria, come descritto al Titolo III, Capo III, Sezione II.
A norma dell'art. 70 è comunque consentito, per scopi di critica o discussione, il riassunto e la citazione, accompagnati dalla menzione del titolo dell'opera e dal nome dell'autore.

AVVERTENZE

I progetti presentati non hanno la marcatura CE, quindi non possono essere utilizzati per scopi commerciali nella Comunità Economica Europea.

Chiunque decida di far uso delle nozioni riportate nella seguente opera o decida di realizzare i circuiti proposti, è tenuto pertanto a prestare la massima attenzione in osservanza alle normative in vigore sulla sicurezza.
L'autore declina ogni responsabilità per eventuali danni causati a persone, animali o cose derivante dall'utilizzo diretto o indiretto del materiale, dei dispositivi o del software presentati nella seguente opera.
Si fa inoltre presente che quanto riportato viene fornito così com'è, a solo scopo didattico e formativo, senza garanzia alcuna della sua correttezza.

L'autore ringrazia anticipatamente per la segnalazione di ogni errore.

Tutti i marchi citati in quest'opera sono dei rispettivi proprietari.

Indice

Capitolo VIII....4
Il display alfanumerico LCD....4
Descrizione hardware del display LCD....6
Elenco dei comandi....8
Procedura standard per l'inizializzazione del display....9
Libreria per la gestione dell'LCD....9
Test della libreria....13
Appendice – inclusione dei file....15
History....18

Capitolo VIII

Il display alfanumerico LCD

Una delle periferiche esterne più semplici da utilizzare, tramite la quale è possibile interfacciare l'essere umano con l'hardware che utilizza, è senza dubbio il display LCD (*Liquid Crystal Display*). Esso presenta diversi vantaggi rispetto a una semplice interfaccia a LED o display 7 segmenti. Innanzitutto ha un costo ormai molto contenuto, consente di scrivere e di leggere in modo semplice un set di caratteri previsti e non previsti dalla tabella ASCII standard ed infine permette, secondo il modello utilizzato, di creare, memorizzare e di richiamare all'occorrenza caratteri personalizzati, ovvero ideati dall'utente. In commercio, esistono display alfanumerici di diversi formati, di cui i più famosi sono sicuramente il 16x1, 16x2, 16x4 ed il 20x2. Il primo numero indica il numero di caratteri (colonne) disponibili per la scrittura, mentre il secondo indica il numero di righe disponibili. Ognuno di questi display è gestito da un integrato, chiamato "*controller*", che permette la comunicazione tra lo stesso display e l'hardware a cui esso viene collegato. Tra i più noti c'è senza dubbio il *controller* HD44780, che è diventato ormai anche uno standard. Infatti sono presenti diversi *controller* compatibili come il KS0066. Esistono due versioni di questo *controller*, una giapponese con sigla HD44778-A00 ed una europea con sigla HD44780-A02, di cui riporto il rispettivo set dei caratteri memorizzati in Figura 1 e Figura 2, estratte dal datasheet.

Figura 1: *Tabella dei caratteri HD44780-A00 (giapponese).*

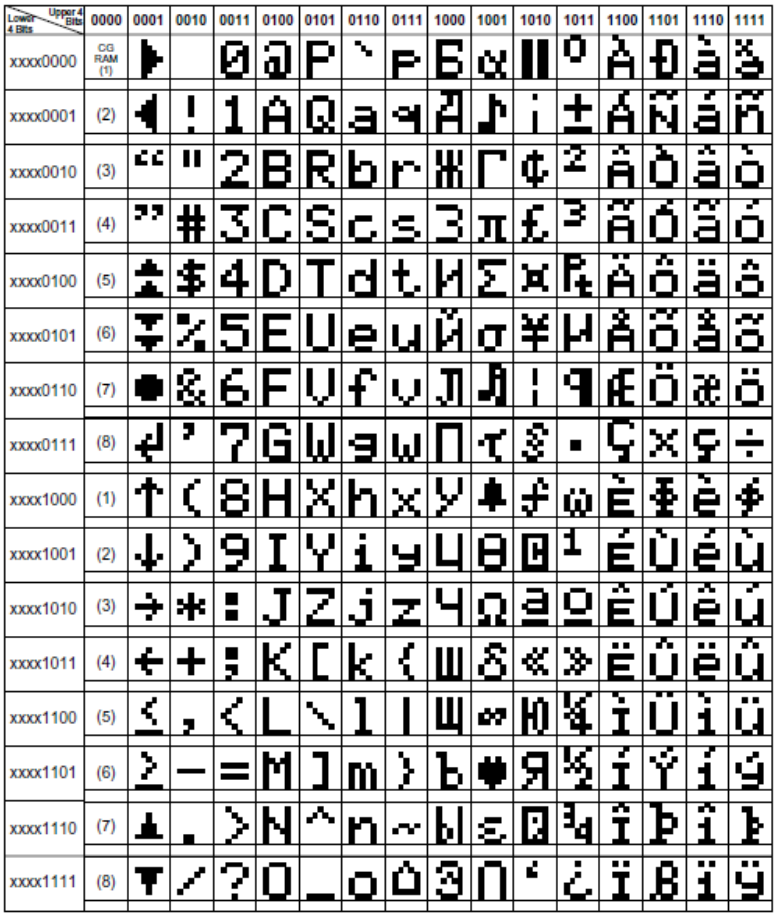

Figura 2: *Tabella dei caratteri HD44780-A02 (Europeo).*

Entrambe le versioni contengono all'interno tre aree (banchi) di memoria chiamati rispettivamente **CGROM** (*Character Generator Read Only Memory*), **CGRAM** (*Character Generator Random Only Memory*) e **DDRAM** (*Display Data Random Access Memory*). Nell'area CGROM è memorizzato in modo permanente il set completo dei caratteri. Nell'area CGRAM **s**ono memorizzati i caratteri non permanenti definiti dall'utente, mentre nell'area DDRAM sono memorizzati i caratteri visualizzati nel display. Dal confronto, si può facilmente vedere che la versione giapponese presenta 33 locazioni libere ma sopratutto è libera l'area CGRAM (colonna 0000), con le quali si potranno creare i caratteri personalizzati. Nel set di caratteri europeo tutte le locazioni sono occupate, compreso CGRAM, per cui non è possibile fare la stessa cosa. Ogni carattere viene scritto nel banco DDRAM in locazioni di memoria non sequenziali, ovvero non contigue tra loro: per esempio, nel caso più semplice di un display 16x2, queste locazioni vanno da 0x00 a 0x0F per la prima riga, e da 0x40 a 0x4F per la seconda riga.
La costruzione di ogni carattere (font) avviene tramite un sistema di coordinate all'interno di una griglia chiamata matrice di punti (pixel), composta da 10 righe x 5 colonne ciascuna. Tuttavia, per la maggior parte del set di caratteri, questa griglia non viene mai rappresentata nelle sue reali dimensioni, ma assume a tutti gli effetti quella corrispondente ad 8 righe x 5 colonne, ad eccezione del cursore, che occupa 10 righe x 5 pixel.

Modalità di funzionamento

Il display LCD può essere programmato con una serie di comandi solo in due modalità: 8 bit e 4 bit. I comandi possono essere sia istruzioni di inizializzazione per il display stesso, oppure comandi per l'invio e ricezione dei dati. Nella modalità ad 8 bit vengono inviati dei comandi composti da 8 bit, nella seconda vengono inviati gli stessi comandi ma divisi in due parti (sequenze) da 4 bit ciascuna, chiamati *nibble* alto e *nibble* basso. Il *nibble* alto è composto dai bit 7-4 ed il secondo composto dai bit 3-0. Prima viene inviato il *nibble* alto e successivamente quello basso. La differenza tra le due modalità è correlata al numero dei pin utilizzati dal microcontrollore per il bus dati; nel primo caso ne occorrono 8, nel secondo caso ne occorrono esattamente la metà. E' quindi evidente che in sistemi embedded, per risparmiare quattro pin della MCU, è spesso più conveniente usare, a parità di risultato, la modalità 4 bit. In base alla modalità di cui si fa uso, il display deve essere collegato in un certo modo all'hardware. Poiché non tutti i display seguono la stessa piedinatura, è necessario consultare il datasheet relativo e prestare molta attenzione ai collegamenti effettuati. Per il resto, i comandi e la tempistica sono generalmente uguali, in particolare in questo capitolo è introdotta la modalità a 4 bit.

Descrizione hardware del display LCD

Un display standard, senza retroilluminazione, presenta 14 pin di collegamento, dal pin 1 al pin 14, mentre quello con retroilluminazione a LED, ne presenta due in più, corrispondenti all'anodo e al catodo, che possono essere anche invertiti secondo il modello utilizzato (quindi si parla di un display a 16 pin). Allo stesso tempo, secondo la marca scelta per il display, questi pin possono essere orientati all'inverso, cioè dal pin 14 al pin 1 senza retroilluminazione, dal pin 16 al pin 1 con retroilluminazione. Per questo è sempre consigliabile leggere il relativo datasheet del display, che seppur basato sullo stesso *controller*, potrebbe avere una piedinatura differente. Questa attenzione eviterà errori di collegamento del modulo LCD, con conseguente danneggiamento del display stesso. In ogni caso, i collegamenti standard da fare per la modalità a 4 bit, a prescindere dall'orientamento dei pin in questione, sono riportati nella seguente Tabella 1.

Pin	Nome	Descrizione
1	**Vss**	GND (massa): livello logico 0.
2	**Vcc**	+5V: livello logico 1.
3	**Vo**	Regolazione del contrasto: prevede un trimmer da 10 - 20K collegato all'alimentazione.
4	**RS**	Bit Register Select: posto a 0 invia una istruzione (comando), posto a 1 invia un dato (es. Carattere).
5	**R/W**	Bit Read/Write: posto a 0 scrive un dato, posto a 1 legge un dato (generalmente usato in scrittura, quindi quasi sempre messo a massa).
6	**E**	Bit Enable: è il segnale di sincronismo con la linea dati; posto a 1 esegue il comando impartito oppure legge il dato, lo interpreta e lo visualizza sul display. Subito dopo queste operazioni, va posto a 0 per permettere la lettura di altri comandi o di flussi di dati in arrivo.
7	**D0**	Bit 0 linea dati: non usato.
8	**D1**	Bit 1 linea dati: non usato.

9	**D2**	Bit 2 linea dati: non usato.
10	**D3**	Bit 3 linea dati: non usato.
11	**D4**	Bit 4 linea dati: usato per la scrittura/lettura dati.
12	**D5**	Bit 5 linea dati: usato per la scrittura/lettura dati.
13	**D6**	Bit 6 linea dati: usato per la scrittura/lettura dati.
14	**D7**	Bit 7 linea dati: usato per la scrittura/lettura dati.
15	**A**	Retroilluminazione display: collegato a Vcc.
16	**K**	Retroilluminazione display: collegato a massa.

Tabella 1: *Collegamenti del display LCD in modalità 4 bit.*

In fase di scrittura, le linee dati servono per inviare un comando da eseguire o per inviare il codice del carattere da visualizzare, in fase di lettura servono per ricevere i dati. La distinzione tra l'invio di un comando oppure di un carattere coinvolge due registri ad 8 bit presenti all'interno del *controller*: Il registro **IR** (*Instruction Register*) e il registro **DR** (*Data Register*). Nel primo registro, in cui si può soltanto scrivere, sono memorizzati solo i comandi, nel secondo le istruzioni che devono essere scritte in uno dei due banchi CGRAM - DDRAM oppure i dati da leggere dagli stessi. Il pin **RS** (*Register Select*) ha un peso determinante in questo caso: se posto a 0, il registro coinvolto nella transazione dati-comandi sarà IR, se posto a 1 il registro coinvolto sarà DR. In fase di lettura, i bit D6-D0 delle linee dati rappresentano la posizione del cursore nelle colonne e righe, tramite l'utilizzo di un contatore chiamato **Address Counter**, mentre il bit D7 rappresenta il bit **Busy Flag**, che informa l'utente quando il *controller* è disponibile o meno durante le operazioni di scrittura/lettura. La posizione del contatore *Address Counter* è molto importante quando si vuole impostarlo sulla colonna e riga desiderata prima di scrivere o leggere un carattere.

Elenco dei comandi

I comandi più usati che si possono inviare al display LCD sono riportati in Figura 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

Figura 3: *Comandi forniti dal controller HD44780.*

Questi comandi offrono una serie di opzioni interessanti che possono essere utilizzate dall'utente per ottenere effetti visivi particolari, in Tabella 2 sono riportati solo quelli da utilizzare per il corretto funzionamento del display in modo standard:

Istruzione (Comando)	Descrizione
Clear display	Pulisce il display ed imposta il cursore nella colonna 1, riga 1.
Return home	Imposta il cursore nella colonna 1, riga 1.
Entry mode set	Incrementa la posizione del cursore.
Display on/off control	Accende il display, cursore visibile e fisso.
Cursor or display shift	Movimento cursore da sinistra a destra.
Function set	Modalità 4 bit.
Set DDRAM address	Imposta il cursore nella colonna e riga in cui leggere o scrivere.
Read Busy Flag & Address	Legge il Busy Flag ed il contenuto di Address Counter.

Tabella 2: *Comandi comuni del controller.*

Procedura standard per l'inizializzazione del display

Come detto prima, affinché il display possa essere utilizzato correttamente deve essere prima inizializzato via software con una serie di istruzioni. Ricordo che ogni volta che si invia un comando, oppure si vuole leggere un dato e visualizzarlo sul display, bisogna sempre portare il pin E (*Enable*) a livello logico 1 per riportarlo subito dopo a livello logico 0 (rispettando i tempi del datasheet). Ad esempio, per impostare l'interfaccia ad 8 bit bisogna portare prima i pin D5 e D4 a 1, poi portare il pin E ad 1 e subito dopo a 0. Prendendo come esempio un display con due righe e font 5x8, la sequenza dei comandi da impartire, è la seguente:

1. Dopo avere dato tensione attendere 15 ms.
2. Portare RS a 0 per informare il *controller* che si vuole inviare dei comandi.
3. Portare D5 e D4 ad 1 (modalità 8 bit).
4. Attendere 4,1 ms.
5. Portare D5 e D4 ad 1.
6. Attendere 100 µs.
7. Portare D5 e D4 ad 1.
8. Attendere 100 µs.
9. Portare D5 ad 1 e D4 a 0 (modalità 4 bit).
10. Inviare il comando 0010 (*nibble* alto) *.
11. Inviare il comando 1000 (*nibble* basso).
12. Spegnere il display inviando il comando 0000 1000.
13. Pulire il display inviando il comando 0000 0001.
14. Inviare il comando 0000 0110 **.
15. Pulire il display inviando il comando 0000 0001.

* Il comando 00101000 imposta il display a 2 righe con font 5x8.
** Il comando 00000110 imposta il cursore da sinistra a destra.

Libreria per la gestione dell'LCD

Stavo iniziando a scrivere una libreria ex-novo per il pilotaggio del display quando ho avuto la fortuna di trovarne una molto completa, ma strutturata per altre MCU AVR. Incuriosito dalla scrittura ed eleganza del codice, dopo averla studiata a fondo, ho deciso comunque di fare una prova sul campo e mi sono accorto che funzionava anche con la MCU 2560, per cui l'ho adottata entusiasticamente. Questa libreria, composta dai file lcd.c e lcd.h è scritta da *Peter Fleury*, prevede due tipi di *controller* e come impostazione base la modalità 4 bit. Le funzioni implementate sono riportate in Tabella 3.

Funzione	Descrizione
void lcd_init (uint8_t dispAttr)	Inizializza il display e seleziona il tipo di cursore.
void lcd_clrscr (void)	Pulisce il display e posiziona il cursore in home.
void lcd_home (void)	Posiziona il cursore nella colonna 1, riga1.
void lcd_gotoxy (uint8_t x, uint8_t y)	Imposta il cursore in una posizione precisa.
void lcd_putc (char c)	Mostra il carattere nella posizione corrente del cursore.

void lcd_puts (const char *s)	Mostra la stringa senza l'avanzamento di riga automatico.
void lcd_puts_p (const char *progmem_s)	Mostra la stringa dalla memoria di programma senza l'avanzamento di riga automatico.
void lcd_command (uint8_t cmd)	Invia al *controller* LCD le istruzioni (comandi).
void lcd_data (uint8_t data)	Invia al *controller* LCD i dati (byte).
#define lcd_puts_P(__s) lcd_puts_p(PSTR(__s))	Macro per memorizzare automaticamente le stringhe come variabili costanti nella memoria di programma.

Tabella 3: *Funzioni della libreria LCD.*

Tutti i parametri modificabili dall'utente si trovano all'interno del file lcd.h. Riporto qui quelli più importanti: come detto prima, sono previsti due tipi di *controller*, l'HD44780 (default) ed il KS0073, codificati rispettivamente con i numeri 0 e 1. Impostando 0 verrà selezionato il primo, impostando 1 verrà selezionato il secondo:

```
/**
 * @name  Definition for LCD controller type
 * Use 0 for HD44780 controller, change to 1 for displays with KS0073
controller.
 */
#ifndef LCD_CONTROLLER_KS0073
#define LCD_CONTROLLER_KS0073 0  /**< Use 0 for HD44780 controller, 1 for
KS0073 controller */
#endif
```

La dimensione del display, ovvero il numero di righe e di colonne, va impostata in questa parte del codice, rispettivamente nei define LCD_LINES e LCD_DISP_LENGHT

```
/**
 * @name  Definitions for Display Size
 * Change these definitions to adapt setting to your display
 *
 * These definitions can be defined in a separate include file \b
lcd_definitions.h instead modifying this file by
 * adding -D_LCD_DEFINITIONS_FILE to the CDEFS section in the Makefile.
 * All definitions added to the file lcd_definitions.h will override the
default definitions from lcd.h
 *
 */
#ifndef LCD_LINES
#define LCD_LINES 2     /**< number of visible lines of the display */
#endif
#ifndef LCD_DISP_LENGTH
#define LCD_DISP_LENGTH 16 /**visible characters per line of the display
*/
#endif
#ifndef LCD_LINE_LENGTH
#define LCD_LINE_LENGTH 0x40 /** internal line length of the display
*/
#endif
#ifndef LCD_START_LINE1
#define LCD_START_LINE1  0x00 /** DDRAM address of first char of line 1
*/
#endif
```

```
#ifndef LCD_START_LINE2
#define LCD_START_LINE2  0x40     /**< DDRAM address of first char of
line 2 */
#endif
#ifndef LCD_START_LINE3
#define LCD_START_LINE3  0x14     /**< DDRAM address of first char of
line 3 */
#endif
#ifndef LCD_START_LINE4
#define LCD_START_LINE4  0x54     /**< DDRAM address of first char of
line 4 */
#endif
#ifndef LCD_WRAP_LINES
#define LCD_WRAP_LINES      0     /**< 0: no wrap, 1: wrap at end of
visibile line */
#endif
```

La configurazione hardware per la modalità 4 bit viene impostata in questa parte di codice, di cui è necessario chiarire alcuni punti fondamentali. La libreria è predisposta per lavorare in due modi diversi: il modo "*Memory Mapped*" ed il modo "*IO Port*", codificati rispettivamente con i numeri 0 e 1. Il primo è utilizzato quando si usa una memoria esterna SRAM, il secondo quando si utilizza una porta qualunque della MCU. Se non si usa la memoria esterna, risulta quindi evidente che si deve obbligatoriamente usare il secondo modo, per cui il define LCD_IO_MODE deve essere impostato a 1 (IO Port). Nel define LCD_PORT va impostata la porta scelta per la comunicazione, ad esempio PORTA, PORTB ecc., mentre nei define LCD_DATA0_PIN - LCD_DATA6_PIN vanno impostati i pin delle linee dati, RS, RW ed E. Anche se questi pin possono essere impostati su porte diverse, consiglio comunque di configurarli tutti su quella scelta.

```
/**
 * @name Definitions for 4-bit IO mode
 *
 * The four LCD data lines and the three control lines RS, RW, E can be
on the
 * same port or on different ports.
 * Change LCD_RS_PORT, LCD_RW_PORT, LCD_E_PORT if you want the control
lines on
 * different ports.
 *
 * Normally the four data lines should be mapped to bit 0..3 on one port,
but it
 * is possible to connect these data lines in different order or even on
different
 * ports by adapting the LCD_DATAx_PORT and LCD_DATAx_PIN definitions.
 *
 * Adjust these definitions to your target.\n
 * These definitions can be defined in a separate include file \b
lcd_definitions.h instead modifying this file by
 * adding \b -D_LCD_DEFINITIONS_FILE to the \b CDEFS section in the
Makefile.
 * All definitions added to the file lcd_definitions.h will override the
default definitions from lcd.h
 *
 */
#define LCD_IO_MODE 1 /**< 0: memory mapped mode, 1: IO port mode */
```

```
#if LCD_IO_MODE

#ifndef LCD_PORT
#define LCD_PORT PORTD /**< port for the LCD lines   */
#endif
#ifndef LCD_DATA0_PORT
#define LCD_DATA0_PORT LCD_PORT     /**< port for 4bit data bit 0 */
#endif
#ifndef LCD_DATA1_PORT
#define LCD_DATA1_PORT   LCD_PORT     /**< port for 4bit data bit 1 */
#endif
#ifndef LCD_DATA2_PORT
#define LCD_DATA2_PORT   LCD_PORT     /**< port for 4bit data bit 2 */
#endif
#ifndef LCD_DATA3_PORT
#define LCD_DATA3_PORT   LCD_PORT     /**< port for 4bit data bit 3 */
#endif
#ifndef LCD_DATA0_PIN
#define LCD_DATA0_PIN    0            /**< pin for 4bit data bit 0  */
#endif
#ifndef LCD_DATA1_PIN
#define LCD_DATA1_PIN    1            /**< pin for 4bit data bit 1  */
#endif
#ifndef LCD_DATA2_PIN
#define LCD_DATA2_PIN    2            /**< pin for 4bit data bit 2  */
#endif
#ifndef LCD_DATA3_PIN
#define LCD_DATA3_PIN    3            /**< pin for 4bit data bit 3  */
#endif
#ifndef LCD_RS_PORT
#define LCD_RS_PORT      LCD_PORT     /**< port for RS line         */
#endif
#ifndef LCD_RS_PIN
#define LCD_RS_PIN       4            /**< pin  for RS line         */
#endif
#ifndef LCD_RW_PORT
#define LCD_RW_PORT      LCD_PORT     /**< port for RW line         */
#endif
#ifndef LCD_RW_PIN
#define LCD_RW_PIN       5            /**< pin  for RW line         */
#endif
#ifndef LCD_E_PORT
#define LCD_E_PORT       LCD_PORT     /**< port for Enable line     */
#endif
#ifndef LCD_E_PIN
#define LCD_E_PIN        6            /**< pin  for Enable line     */
#endif
```

Test della libreria

Per eseguire il test della libreria ho utilizzato un display 16x2 con controller HD44780, collegato alla PORTD, come evidenziato in Figura 4. Si noti in particolare il trimmer RV1 necessario per la regolazione del contrasto.

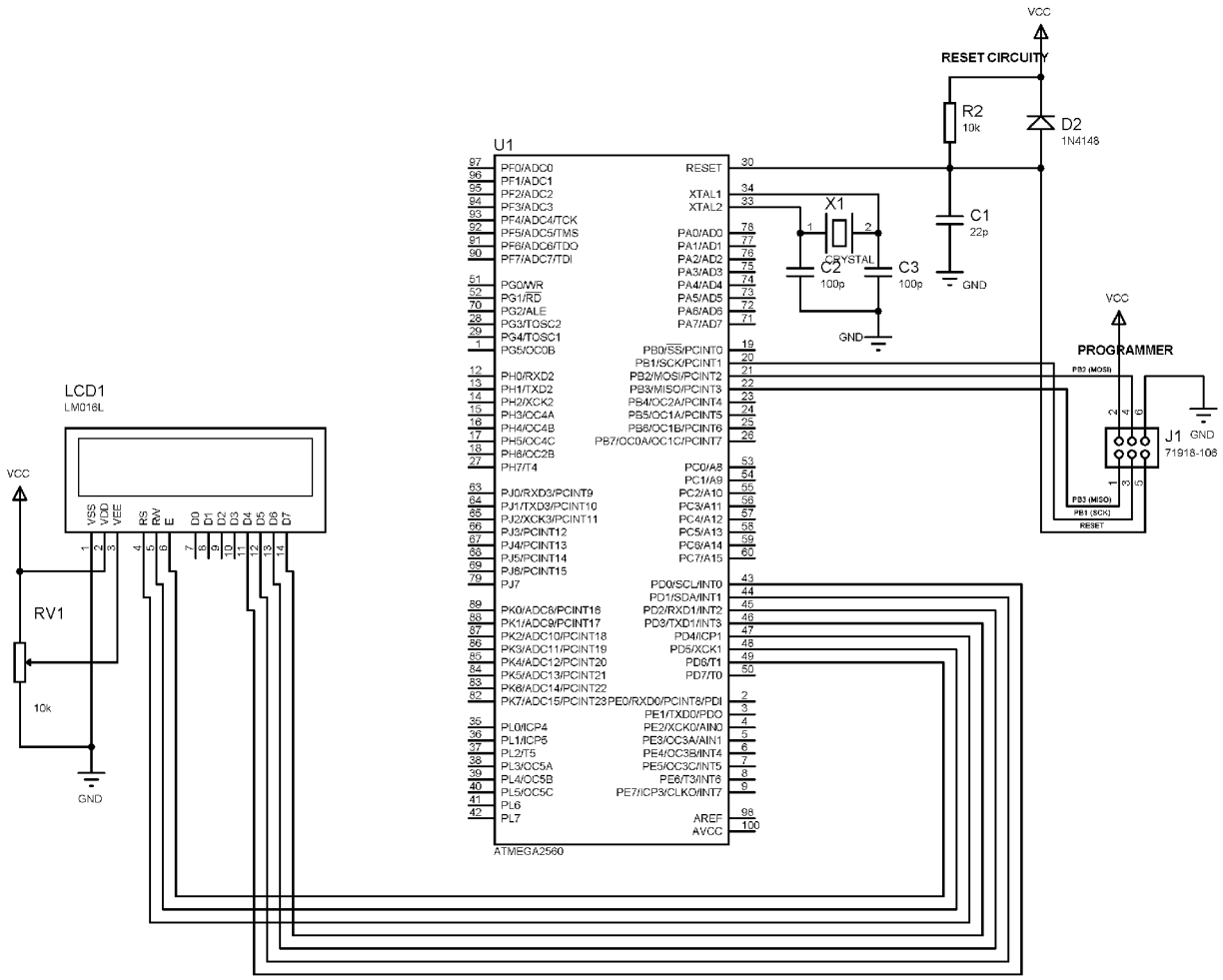

Figura 4: *Schema elettrico dei collegamenti del modulo LCD.*

Ho poi scritto un piccolo programma di prova, includendo manualmente nel progetto i file lcd.c e lcd.h prima della compilazione. L'inclusione di tali file deve essere eseguita ogni volta che nel progetto è previsto l'uso di un display lcd.

```
#pragma config BODLEVEL = BODLEVEL_2V7
#pragma config OCDEN = CLEAR
#pragma config JTAGEN = SET
#pragma config SPIEN = SET
#pragma config WDTON = CLEAR
#pragma config EESAVE = CLEAR
#pragma config BOOTSZ = BOOTSZ_4096W_1F000
#pragma config BOOTRST = SET
#pragma config CKDIV8 = CLEAR
#pragma config CKOUT = CLEAR
#pragma config SUT_CKSEL = SUT_CKSEL_EXTXOSC_8MHZ_XX_16KCK_65MS

// Definisco la frequenza della CPU
#define F_CPU   16000000UL

// Includo i file per la gestione I/O, dei ritardi e dell'lcd
#include <avr/io.h>
#include <util/delay.h>
#include "lcd.h"
#include "lcd.c"

int main(void){

 // COLLEGARE IL DISPLAY ALLA PORTC DELL'ATMEGA !!!!
 // Inizializzo il display lcd, display acceso, cursore presente. (Per
ulteriori funzioni di inizializzazione vedere il file lcd.h)
 lcd_init(LCD_DISP_ON_CURSOR);

 while(1){

    // Pulisco lo schermo
    lcd_clrscr();

    // Posiziono il cursore sulla riga 1, colonna 1
    lcd_home();

    // Scrivo qualcosa
    lcd_puts("test lcd");

    // Posiziono il cursore sulla riga 2, colonna 1
    lcd_gotoxy(0,1);

    // Scrivo qualcosa
    lcd_puts("superato !!!");

    // Piccolo ritardo
    _delay_ms(50);
 }
}
```

In questa libreria, la codifica delle colonne e delle righe inizia sempre con il numero 0. Per esempio, con un display 16x4 la numerazione delle colonne va da 0 a 15, quella delle righe da 0 a 3. Ad esempio, volendo posizionare il cursore in quella che l'essere umano chiama, iniziando dal numero 1, colonna 10, riga 3, il comando da inviare è lcd_gotoxy (9, 2).

Appendice – inclusione dei file

Se si fa uso dell'IDE Atmel Studio, l'inclusione manuale va fatta in questo modo: nella finestra *Solution Explorer* (in alto a destra) cliccare con il tasto destro sul nome del progetto, selezionare:

> `Add → Existing Item.`

Cliccare su quest'ultimo, come mostrato in Figura 5.

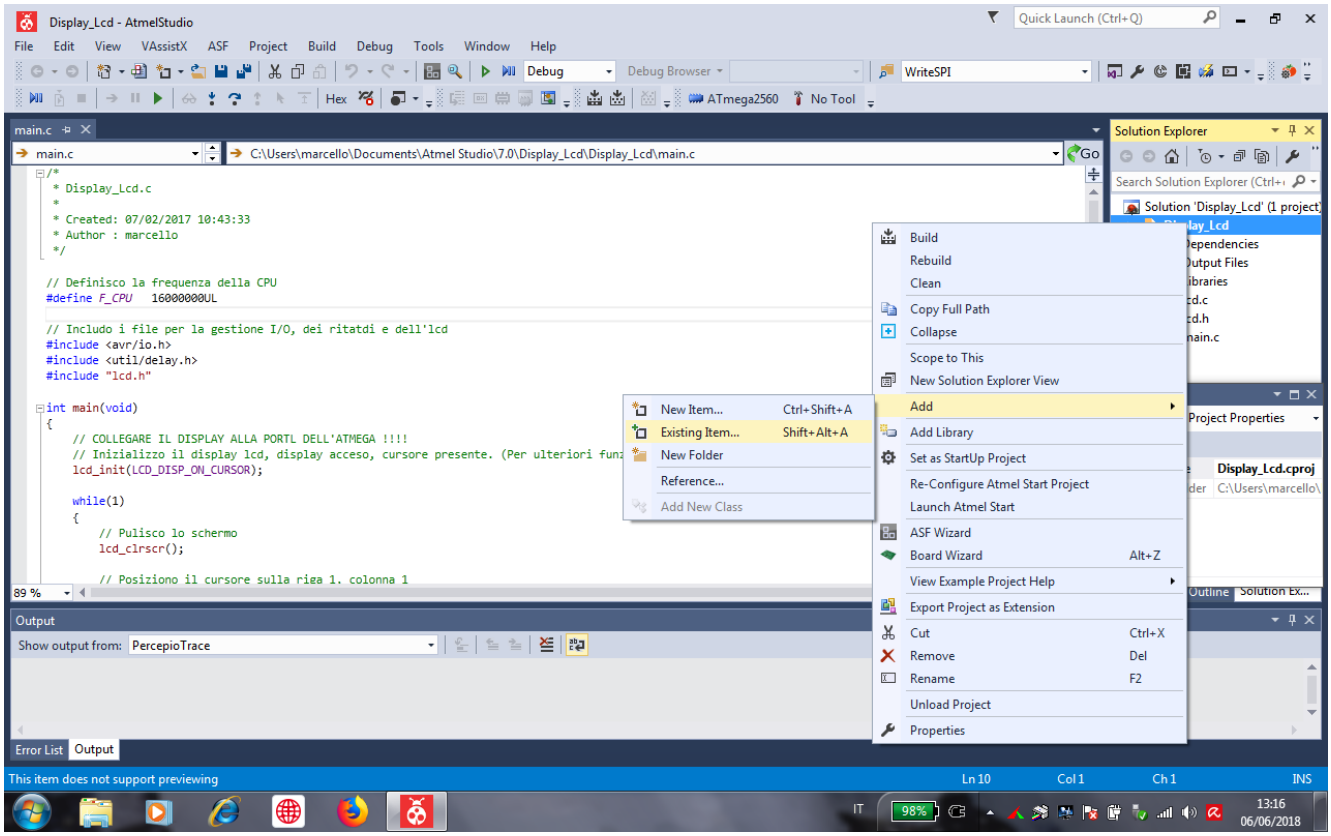

Figura 5: *Aggiungere file di libreria manualmente.*

Si aprirà una finestra che permette all'utente di raggiungere la cartella in cui sono archiviati i due file della libreria (Figura 6)

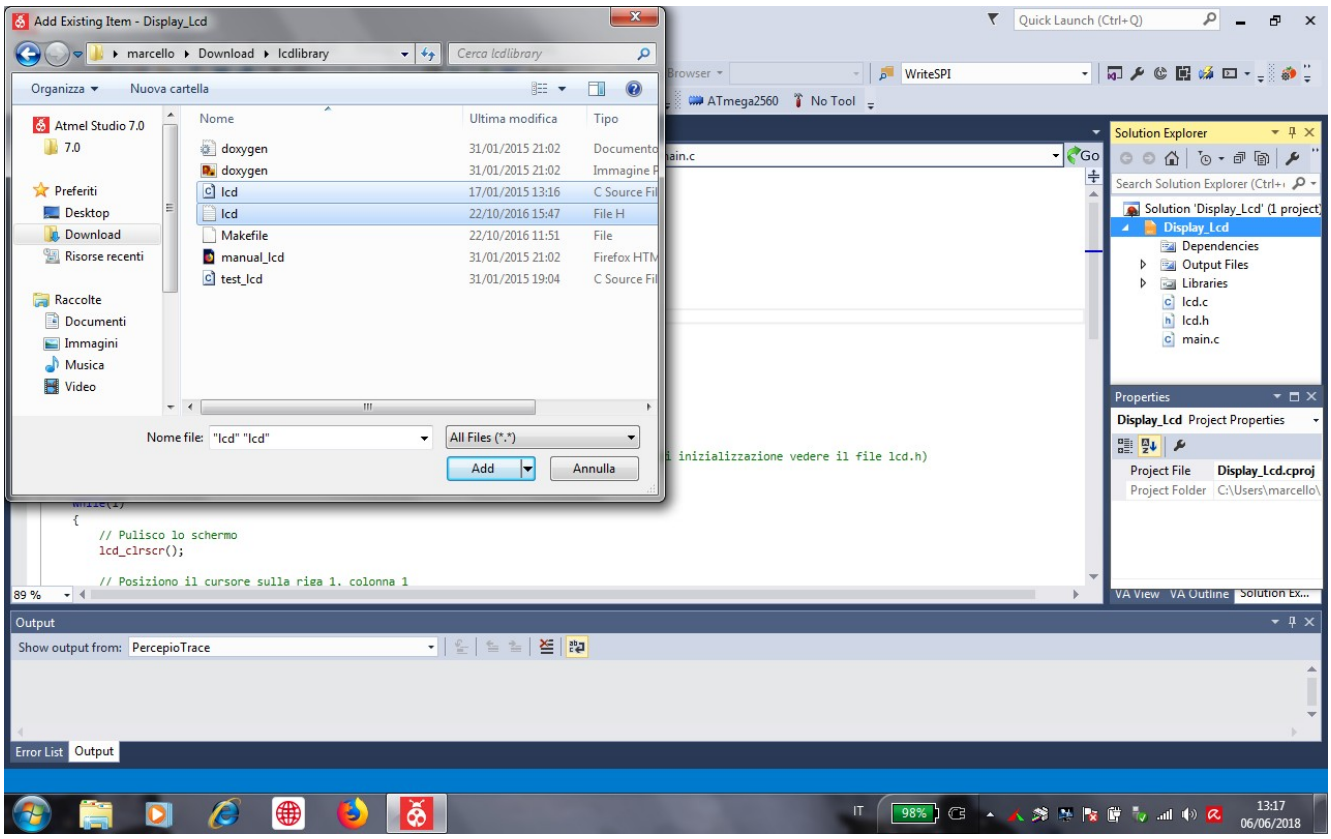

Figura 6: *Selezione dei file.*

Evidenziare i due file e premere il tasto *Add*. A quel punto, sempre nella finestra *Solution Explorer*, sotto la voce *Libraries*, compariranno i file selezionati. Ora si può compilare il progetto che fa uso della libreria.

Indice Alfabetico

A
Address Counter....7
B
Busy Flag....7
C
CGRAM....5, 7
CGROM....5
controller....4
D
Data Register....7
DDRAM....5, 7
DR....7
H
HD44780....4
I
Instruction Register....7
IR....7
K
KS0066....4
L
LCD....4
P
Peter Fleury....9
R
Register Select....7
retroilluminazione....6
RS....7
S
Solution Explorer....15

History

Data	Versione	Autore	Revisione	Descrizione Cambiamento
22/09/20	1.0	Marcello Pinna	Mauro Laurenti	Versione Originale.