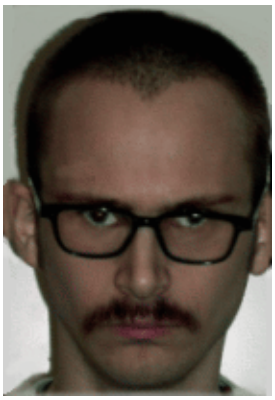
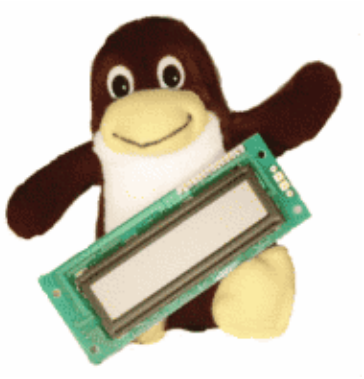
Capire i display LCD compatibili HD44780



by Jan Svenungson
<jan.svenungson(at)linux.nu>

About the author:

Jan usa GNU/Linux dal 1996 e ha avuto 2 riavvii non intenzionali da allora (non contando i riavvii dovuti a cadute di tensione).



Abstract:

Questo articolo prova ad insegnarvi una o due cose a proposito dei display LCD compatibili HD44780.
Vedremo come collegarlo ad una porta parallela e come progrmmarlo con un piccolo programma chimato LCDInfo.
Non dovete solamente collegare il display, lanciare un programma e avere ciò che vi serve sul display ma anche capire come l'hardware può fare ciò che volete.

Introduzione

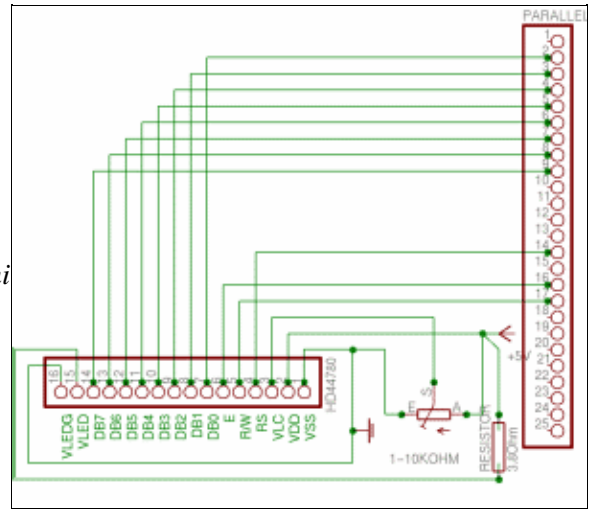
Primo, dovete avere dell'hardware e del software, suppongo che abbiate già un computer con una porta parallela (stampante) standard su cui potete usare GNU/Linux con gcc e glibc.
Avete anche bisogno di un display LCD che sia compatibile HD44780, cavi per collegarlo alla porta parallela e anche un potenziometro se volete cambiare il contrasto. Per alimentare il display avete bisogno di più energia di quanta la vostra porta parallela possa darvi e potreste avere bisogno di prenderla da qualche altra parte nel vostro computer. Il modo migliore per farlo e di usare una connessione +5V standard (quelle usate per alimentare i floppy, i dischi fissi, ecc.).

Quando avete connesso il display LCD dovete sapere come funziona. Questo è quello che normalmente viene tralasciato in altri articoli riguardanti l'oggetto ma io proverò a spiegare alcune funzioni interne del display che vi aiuteranno a programmarlo.

L'ultima cosa da fare è attualmente far stampare qualcosa di utile al display. Come riferimento userò un piccolo programma chiamato LCDInfo che supporta la maggior parte delle caratteristiche del HD44780 ma al momento non stampa molto. Questo programma è a livello alpha e ci lavoro a tempo perso.
Se non avete mai programmato in C prima potete considerare di leggere qualcosa sul C, suppongo che voi

siate principianti in C dato che questo è il mio attuale livello.

Come collegarlo



Primo guardiamo i differenti pin disponibili sul LCD e spieghiamo cosa fanno.
Pin 1 è chiamato VSS e suppongo vada a terra (GND).
Pin 2 è chiamato VDD ed è il pin che fornisce l'alimentazione da +5V.
Pin 3 è chiamato VLC ed è connesso al potenziometro che definisce il contrasto del display.
Pin 4 è il pin RS da cui dipende il display per avere *istruzioni* o *dati*.
Pin 5 è il pin R/W che controlla quando il display sta *inviando* o *ricevendo*.
Pin 6 è il pin di Abilitazione. Quando cambia da basso a alto e poi nuovamente basso l'LCD legge i pin 4,5 e 7−14.
Pin 7−14 sono la *linea bus dati* chiamati DB0−DB7, questi sono i principali bit dati inviati all'LCD e dove e come devono essere scritti sul display.
Pin 15 e 16 sono presenti solo sui display retroilluminati e sono semplicemente +5V e GND con una resistenza di 3.8 Ohm tra il pin 15 e +5V.

Per trovare dove connettere questi pin sulla porta parallela potete guardare lo schema sulla destra dove ho provato a renderlo il più chiaro possibile. Click sul diagramma per un'immagine più grande.
Questo schema si applica solamente se volete cambiare il contrasto sul display. Io ho semplicemente collegato il pin 3 e il pin 1 a GND che funziona bene per me, se avete una strana illuminazione nella stanza potete considerare di aggiungere il potenziometro.

Quando prendete l'alimentazione dal PC fate attenzione. Se la prendete dal cavo sbagliato avrete +12V che friggerà il vostro LCD. Il cavo che vi serve è quello rosso. Il giallo è +12V e il nero è la terra (GND).
Se avete fatto questo correttamente l'LCD avrà la prima (e la terza se esiste) riga nera quando accenderete il PC.

Come funziona l'LCD

L'LCD non fa nulla autonomamente, semplicemente aspetta finché riceve un attivazione valida alto−basso (quando mettiamo il pin "enable" alto, aspettiamo un'attimo e poi lo riabbassiamo). A questo punto il display legge se sono istruzioni o dati da processare, poi se riceverà o spedirà informazioni e alla fine i bit dati vengono spediti o ricevuti.
In questo articolo non riceveremo mai informazioni dell'LCD così il pin R/W sarà sempre basso, che significa scrivi.
Il pin RS sarà basso eccetto quando stamperemo caratteri, qualunque altra cosa è considerata instruzioni. Questo rende molto semplice programmare il display.

Sapendo questo vogliamo partire accendendo il display e prepararlo per ricevere informazioni. Questo è fatto nella sequanza di inizializzazione dove diciamo al display di accendersi, quale "set di funzioni" usare, ecc.
L'alimentazione deve essere già accesa se prendete l'energia da un cavo del PC, altrimenti è la prima cosa da

fare.
La prossima è il "set di funzioni" che dipende dal display che avete.
Per renderlo più facile da capire vi spiegherò esattamente cosa facciamo quando usiamo un set di funzioni.
DB2 è il bit del Font caratteri e può essere *basso* che significa una matrice dot 5x7.
BD3 è il bit delle Linee Display e può essere *alto* che significa 2 linee. Cosa fare se avete 4 linee sul display? Non preoccupatevi, la prima e la terza linea sono le stesse nella memoria del display così potete usere alto lo stesso.
DB4 è il bit Lunghezza Dati e decide se avete 4 o 8 DB, se collegate il display secondo il mio schema avrete DB *alto*.
Mettere *DB5* alto per dire al display che questo è un "Set di Funzioni", poi assicuratevi che *RS* e *R/W* siano *basso* e date un alto−basso a enable. Per i tempi controllate il manuale, suppongo che aspettare qualche microsecondo sia più che sufficiente.

E il codice?

Qui vediamo le parti del programma LCDInfo che vi servono per capire come lavora l'interfaccia del HD44780. Potete scaricare il programma LCDInfo alla fine dell'articolo o guardare direttamente i file con il codice C iolcd.c and lcdinfo.c cliccando qui.

Quello di cui abbiamo bisogno adesso sono le istruzioni C e credetemi quando vi dico che è facile. Mi sposterò nel codice passo dopo passo e anche se siete dei principianti del C capirete.
Prima di tutto includiamo alcuni file header e definiamo le funzioni (controllate i sorgenti per le informazioni). Poi iniziamo la parte divertente.

```
#define D_REGISTER 0
#define I_REGISTER 2
#define WRITE_DATA 8
#define BASE 0x378

int main(void)
{
  ioperm(BASE,3,1);
  [CUT]
}
```

Questa è la prima istruzione nella funzione main che ci permette di usare la porta parallela. BASE deve essere 0x378 o simile e il "3" significa che abbiamo accesso a 0x378, 0x379 e 0x380 che normalmente sono tutte le porte stampanti.
La ragione per cui ci sono 3 indirizzi è che la porta è divisa tra dati, status e controllo. Per noi significa che dobbiamo impostare il pin dati per primo, il pin controllo per secondo e non possiamo farlo in un solo comando.
La prossima cosa da fare è il set di funzioni descritto sopra.

```
void function_set(void)
{
  outb(56, BASE);
```

Questo imposta il pin DB su una matrice dot 5x7, 2 linee ecc.

```
  outb(I_REGISTER + WRITE_DATA, BASE + 2);
```

Questo imposta il pin RS e R/W su istruzioni e scrivi. Ho creato delle variabili globali di I_REGISTER e WRITE_DATA e sono uguali a 2 e 8.
Il prossimo e alto−basso su enable

```
  outb(ENABLE + I_REGISTER + WRITE_DATA, BASE + 2);
  usleep(0);
  outb(I_REGISTER + WRITE_DATA, BASE + 2);
}
```

Quello che fa questo codice è impostare enable su alto, aspettare e impostarlo su basso. Il comando usleep(0); non è l'ideale ma non ho finito il codice temporizzatore per il display.
Qualcuno di voi potrebbe chiedersi perchè imposto RS e R/W su *on* nel codice quando ho detto che devono essere impostati su basso sulle istruzioni. Questo perchè i pin 1, 14 e 17 sono "invertiti a livello hardware" e significa che se il pin 14 è "off" per quanto riguarda la porta stampante, il pin è alto!
Bene, vi ho detto quanto era semplice, no?

Come visualizzare i caratteri

Forse volete avere un esempio pratico del vostro display, come visualizzare del testo? Nessun problema.
Il codice (codice come comandi) è identico per stampare un carattere e per impostare una funzione. L'unica cosa che dobbiamo fare è cambiare alcune variabili. Non vogliamo impostare RS su istruzioni ma sui dati da cui partire. Questo rende la funzione print_character() come:

```
void print_character(int character)
{
  outb(D_REGISTER + WRITE_DATA, BASE + 2);
  outb(character, BASE);
  outb(ENABLE + D_REGISTER + WRITE_DATA, BASE + 2);
  usleep(0);
  outb(D_REGISTER + WRITE_DATA, BASE + 2);
}
```

Come potete vedere combiamo "I_REGISTER" con "D_REGISTER" e "56" con "character" ma cosa significa? Se guardate i codici carattere sul vostro manuale capirete.
L'unica cosa da fare è fornire un carattere alla funzione (dato che usiamo il C non dobbiamo prima definire un intero) e poi il carattere comparirà nel display.

Con questo codice avete uno scheletro di un programma LCD, usatelo per le vostre necessità, visualizzare la memoria libera, visualizzare le connessioni http attive o qualunque altra cosa. Alcuni esempi si trovano sul programma LCDInfo che mostra alcune cose disponibili nel filesystem proc di un computer GNU/Linux.

Riferimenti

- Per informazioni sulla porta stampante, vedere http://et.nmsu.edu/~etti/fall96/computer/printer/printer.html che ha qualche esempio. (un copia di questo articolo si trova >qui<)
- Per informazioni sui programmi LCD, controllarehttp://lcdproc.omnipotent.net/ che è un buon programma LCD.
  Grazie a Sven e Reinhold per alcuni link.
- Il codice sorgente del programma LCDInfo: lcdinfo−0.02.tar.bz2.
  Aggiornamenti disponibili su: http://savannah.gnu.org/download/lcdinfo

en −−> it: Fabio Baseggio <base/at/trevinet.it>

2005−01−10, generated by lfparser_pdf version 2.51