Display LCD YB1602A

Naoki Pross

Elettronica

Il display LCD da solo ha integrato anche un driver **SPLC780C** della SUNPLUS che permette di generare i caratteri senza doverli disegnare con i 5x8 pixels per carattere.

Il componente esternamente dispone di una porta **CON1** con 16 pin. Questi sono cablati internamente nel seguente modo:

Pin	Simbolo	Valore	Funzione
1	VSS	0V	Massa alimentazione elettronica
2	VDD	5.0V	Alimentazione elettronica
3	V0		Controllo contrasto
4	RS	`High / Low`	`High` = Dati, `Low` = Comando
5	R/W	`High / Low`	`High` = Read, `Low` = Write
6	E	`High / Low`	Enable / Clock manuale
7	DB0	`High / Low`	Data Bus bit 0
...	...	`High / Low`	...
14	DB7	`High / Low`	Data Bus bit 7
15	LEDA	5.0v	Alimentazione backlight
16	LEDK	0V	Massa alimentazione backlight

Le funzioni di base per controllare il driver sono controllate attraverso i pin `RS`, `R/W`, `E` e il `Data Bus`.

Come ogni circuito elettronico digitale dispone di un clock interno **con il trigger al fianco di discesa**, che inoltre può anche essere controllato manualmente attraverso il pin `E`.

Software

Il display dispone di un sistema di comandi dati in una `instruction table` che permettono di eseguire operazioni particolari come cancellare il display o applicare delle modifiche nella RAM. Perciò é presente un bit di controllo `RS`

che determina se i dati mandati nel bus devono essere eseguiti come comando o allocati nella CGRAM (Character Generator RAM).

Esecuzione dei comandi

Essendo un sistema di controllo puramente elettronico per eseguire dei comandi si deve controllare il clock `E` nel seguente modo.

```
/* NOTA: Dopo ogni operazione è presente un delay di pochi
 *       millisecondi per permettere al diplay di eseguire
 *       l'operazione poichè il clock dell'LCD funziona a
 *       ~270 kHz mentre l'arduino a 16 MHz.
 *
 *      1/270Hz = 3.7 us
 */

#define D_TIME    4
#define P_ENABLE 26
#define P_RS     27
#define P_RW     28

void setup() {
    // I/O setup ...
    // Blocco del clock
    digitalWrite(P_ENABLE, HIGH);
    delayMicroseconds(D_TIME);

    /* Preparazione del comando inviare (dalla instruction table)
       in questo caso 'Clear Display' */
    PORTC = 0x01;
    digitalWrite(P_RS, LOW);
    digitalWrite(P_RW, LOW);
    delayMicroseconds(D_TIME);

    // Esecuzione del comando, Sblocco del clock
    digitalWrite(P_ENABLE, LOW);
    delayMicroseconds(D_TIME);
}
```

Senza utilizzare le librerie di Arduino (quindi con AVR).

```c
#include <avr/delay.h>

#define D_TIME 4

// on PORTB
#define E  0
#define RS 1
#define RW 2

int main() {
    // I/O setup ...
    // 'Clear Display'
    PORTB |= 1<<E;               // blocco clock
    PORTC = 0x01;                // istruzione
    PORTB &= ~(1<<RS | 1<<RW);   // RS = dati, RW = Write
    _delay_us(D_TIME);           // wait
    PORTB |= 1<<E;               // esecuzione

    return 0;
}
```

Inizializzazione

Il display una volta acceso elettricamente esegue una serie di operazioni che resettano il contenuto dello schermo, ma comunque quest'ultimo non è ancora pronto per lavorare perchè é necesario specificare dei valori.

Quindi in ordine si devono eseguire le seguenti operazioni:

- `Function Set` imposta
  - Il numero di bit usati per la comunicazione
  - Il numero di righe del display
  - La dimensione dei caratteri (5x8 o 5x10)
- `Turn Display On`
  - Accende il display
  - Attiva il cursore
  - Imposta se il curore deve lampeggiare
- `Entry Mode Set` imposta
  - La direzione in cui il testo viene scritto
  - Se il testo deve shiftare

Questo si riscrive in C++ nel modo seguente:

```
#define D_TIME    4

#define P_ENABLE 26
#define P_RS     27
#define P_RW     28

void setup() {
    // Function Set
    digitalWrite(P_ENABLE, HIGH);
    delay(D_TIME);

    PORTC = 0b0011100;
    digitalWrite(P_RS, LOW);
    digitalWrite(P_RW, LOW);
    delay(D_TIME);

    digitalWritE(P_ENABLE, LOW); // Esecuzione
    delay(D_TIME);

    // Turn Display On
    digitalWrite(P_ENABLE, HIGH);
    delay(D_TIME);

    PORTC = 0b00001110;
    digitalWrite(P_RS, LOW);
    digitalWrite(P_RW, LOW);
    delay(D_TIME);

    digitalWrite(P_ENABLE, LOW); // Esecuzione
    delay(D_TIME);
}
```

Scrivere

Il testo che si desidera scrivere sul display LCD viene allocato nella CGRAM, controllata dal driver che converte automaticamente la codifica ASCII in una serie di pixels sul display.

Il valore può essere scritto o letto dal bus, perciò si controlla il pin `RW` che

abilita la modalità lettura (read) se a `HIGH` o `1`, e la modalità scrittura (write) se a `LOW` o `0`.

Inoltre il pin `RS` deve essere attivato (`HIGH`) per segnalare che tutti gli 8 bit sul bus sono dati.

```
void write(String text) {
    for (int i = 0; i < text.length(); i++) {
        char c = text.charAt(i);

        digitalWrite(P_RS, HIGH);
        digitalWrite(P_RW, LOW);
        PORTC = c;
        delay(D_TIME);

        digitalWrite(P_ENABLE);
        delay(D_TIME);
    }
}
```

Questa funzione è una ricostruzione di quello che suppongo sia la funzione data dalla libreria lcd di arduino.