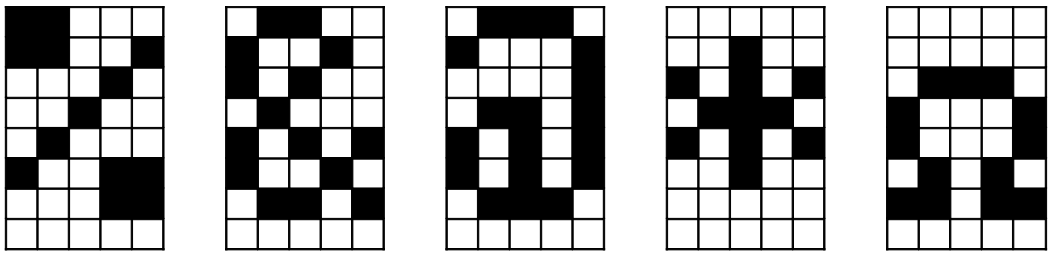
DISPLAY LCD HD44780U(LCD-II)

1. Descrizione

1.1 Introduzione

Molti dispositivi a microcontrollore usano un visualizzatore (display) LCD per mostrare delle informazioni, uno dei tipi più usati può mostrare 16 caratteri alfanumerici su una fila oppure 32 caratteri su due file sovrapposte. Per rappresentare il carattere si usa una matrice di 40 punti disposti su 8 righe ciascuna di 5 punti. Rendendo scuri i punti in modo selettivo si possono realizzare caratteri anche molto complessi.
Il carattere alfanumerico può essere un numero da 0 a 9, una lettera dell'alfabeto oppure un carattere particolare. A titolo di esempio la figura illustra come sono costruiti i simboli %, &, @, *, Ω.



È prevista anche una zona di memoria dove salvare caratteri personalizzati costruiti dall'utente.
Se si considera un visualizzatore ad una riga di 16 caratteri si vede che i punti da gestire sono 16 x 40 = 640. È ovvio che una quantità così elevata richiede l'uso di dispositivi complessi che di solito sono dei microcontrollori dedicati. Uno dei più diffusi è il HD44780 della Hitachi, usato non solo dalla casa produttrice per i suoi visualizzatori ma anche da molti altri produttori di LCD in tutto il mondo. Perciò qui saranno trattati i visualizzatori Hitachi LCD la cui struttura hardware e gestione funzionale si possono considerare come degli standard. Molto spesso l'insieme di visualizzatore LCD e di microcontrollore è chiamato Liquid Crystal Module o LCM.
Questi moduli o LCM sono poco costosi, facili da usare e disponibili in diversi formati, fino a 8 righe da 80 caratteri.
Per semplificarne l'uso i display Hitachi hanno una memoria ROM interna in cui è disponibile un insieme di 640 caratteri tra cui simboli ASCII, caratteri giapponesi (caratteri kata), lettere greche e simboli matematici.
Ognuno dei caratteri è accessibile singolarmente mandando il codice corrispondente al driver/controller (il solito microcontrollore HD44780U) montato sul retro del display. Poiché la ROM è scritta durante la fabbricazione del microcontrollore l'associazione tra codice e carattere è fissa. Sono disponibili quattro tipi di tabelle, riportate nelle ultime pagine
L'uso di codici a 8 bit permette di ridurre le linee per il pilotaggio che può essere fatto usando un bus a 8 bit o addirittura 4 bit. Nel primo caso servono, oltre all'alimentazione di +5Vcc e la massa, 11 linee di I/O; nel secondo solo 7 più l'alimentazione. Inoltre, quando il display non è' abilitato, le uscite vengono messe in alta impedenza ed è quindi possibile utilizzare le linee collegate al visualizzatore anche per altri scopi.

1.2 Caratteristiche tecniche

Le caratteristiche principali di un tipico modulo LCM sono:

- matrice disponibile 5x8 e 5x10 punti (5 colonne per 8 o 10 righe : 40 o 50 punti/carattere)
- tensione di alimentazione 2.7÷5.5 V
- interfacciamento con bus di MPU ad alta velocità – 2 MHz (quando Vcc=5 V)
- interfacciamento a 4 oppure 8 bits
- 80 x 8 bit diplay RAM (80 caratteri massimi)

- 9920-bit in ROM generatore di caratteri per un totale di 240 fonts:
    - 208 caratteri fonts (5x8 dot)
    - 32 caratteri fonts (5x10 dot)
- 64x8 bit generatore di caratteri in RAM:
    - 8 caratteri fonts (5x8 dot)
    - 4 caratteri fonts (5x10 dot)
- 16-common x 40-segment driver lcd
- duty cycles programmabili:
    - 1/8 per una linea di 5x8 punti con cursore
    - 1/11 per una linea di 5x10 punti con cursore
    - 1/16 per due linee di 5x8 punti con cursore
- Ampia gamma di istruzioni disponibili quali:
    - clear display, cursor home, display on/off, display character blink, cursor shift, display shift
- reset automatico, all'accensione, che inizializza il controller-driver
- oscillatore interno con resistori esterni
- basso consumo, valori tipici sono 1 mA a 5 V

Naturalmente queste caratteristiche possono variare dea un costruttore all'altro e vanno verificate nei data sheet del modulo che si utilizza.

1.3 Piedinatura

Nei visualizzatori fino a 80 caratteri di solito ci sono 14 o 16 terminali. Nella versione a 16 terminali i due aggiunti servono quasi sempre per la cosiddetta "retroilluminazione" cioè alimentano dei LED che illuminano lo schermo, rendendolo visibile anche con luce ambiente bassa.
Dei 14 terminali sempre presenti i primi 3 sono di alimentazione (Vcc, Vss e Vee per il contrasto), 3 di controllo e 8 per il bus dati. Le 3 linee di controllo utilizzate sono:
Enable (E): La transizione dal livello alto al livello basso determina l'acquisizione del dato presente sulle linee dati..
Read/Write (R/W): Imposta la direzionalità dei dati presenti sul bus $DB_7 \div DB_0$. Se R/W = 0 il display legge, altrimenti scrive sul bus.
Register Select (RS): Quando è 0 indica che una istruzione sta per essere scritta sul bus, se è a 1 significa che un sta per essere scritto un carattere.

La piedinatura riportata in tabella 1 è quella tipica a un display di massimo 80 caratteri. Nella tabella 2 quella tipica di un display con più di 80 caratteri.
N.B. La piedinatura può cambiare da un visualizzatore all'altro perciò va **sempre preventivamente controllata** sul data-sheet del componente.

Tabella 1 – Esempio di piedinatura di un visualizzatore fino a 80 caratteri

Pin	Simbolo	Livello	I/O	Funzione
1	Vss	-	-	Alimentazione (GND)
2	Vcc	-	-	Alimentazione (+5V)
3	Vee	-	-	Regolazione contrasto [1]
4	RS	0/1	I	0 = Input istruzioni; 1 = Input Data
5	R/W	0/1	I	0 = Scrivi sul display ; 1 = Leggi da display
6	E	1, 1→0	I	Enable
7	DB0	0/1	I/O	Linea 0 bus dati (LSB)
8	DB1	0/1	I/O	Linea 1 bus dati
9	DB2	0/1	I/O	Linea 2 bus dati
10	DB3	0/1	I/O	Linea 3 bus dati
11	DB4	0/1	I/O	Linea 4 bus dati
12	DB5	0/1	I/O	Linea 5 bus dati
13	DB6	0/1	I/O	Linea 6 bus dati
14	DB7	0/1	I/O	Linea 7 bus dati (MSB)

Tabella 2 – Esempio di piedinatura di un visualizzatore con più di 80 caratteri

Pin	Simbolo	Livello	I/O	Funzione
1	DB7	0/1	I/O	Linea 7 bus dati (MSB)
2	DB6	0/1	I/O	Linea 6 bus dati
3	DB5	0/1	I/O	Linea 5 bus dati
4	DB4	0/1	I/O	Linea 4 bus dati
5	DB3	0/1	I/O	Linea 3 bus dati
6	DB2	0/1	I/O	Linea 2 bus dati
7	DB1	0/1	I/O	Linea 1 bus dati
8	DB0	0/1	I/O	Linea 0 bus dati (LSB)
9	E1	1, 1→0	I	Enable riga 0 e 1
10	R/W	0/1	I	0 = Scrivi sul display ; 1 = Leggi da display
11	RS	0/1	I	0 = Input istruzioni; 1 = Input Data
12	Vee	-	-	Regolazione contrasto [1]
13	Vcc	-	-	Alimentazione (+5V)
14	Vss	-	-	Alimentazione (GND)
15	E2	1, 1→0	I	Enable riga 2 e 3
16	nc			

1.4 Invio di dati

Il trasferimento di dati o istruzioni al visualizatore è sincrono, cioè avviene in corrispondenza di un impulso di clock. Più precisamente il microcontrollore che controlla lo schermo legge i dati presenti in ingresso al visualizzatore LCD in corrispondenza del fronte di discesa sul terminale di Enable.
Naturalmente nel caso di interfacciamento a 8-bit il riconoscimento del dato avviene con un singolo impulso di enable mentre nel caso di interfacciamento a 4-bit il trasferimento è fatto in due tempi.
In particolare l'interfaccia a 4 bit usa i 4 bit MSB del display e richiede due impulsi di Enable, con il primo fronte di discesa si trasferiscono i 4 bit più significativi, con il secondo i 4 bit LSB del dato.
In figura 1.3 è mostrato l'andamento delle forme d'onda con interfaccia a 4 bit. In corrispondenza del primo fronte di discesa dell'Enable sono trasferiti i 4 bit più significativi, sul secondo fronte di discesa i 4 bit meno significativi.

[1] A questo terminale è applicata una tensione variabile, di solito inferiore a 4 V, che determina il contrasto tra i punti scuri che compongono il carattere e i punti chiari che fanno da sfondo.

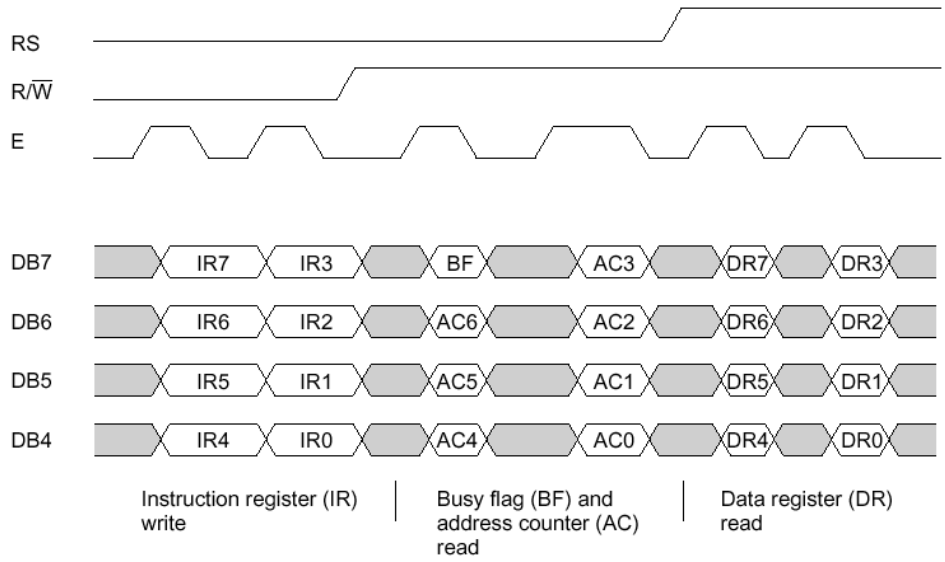


Figura 1.3

1.5 Inizializzazione

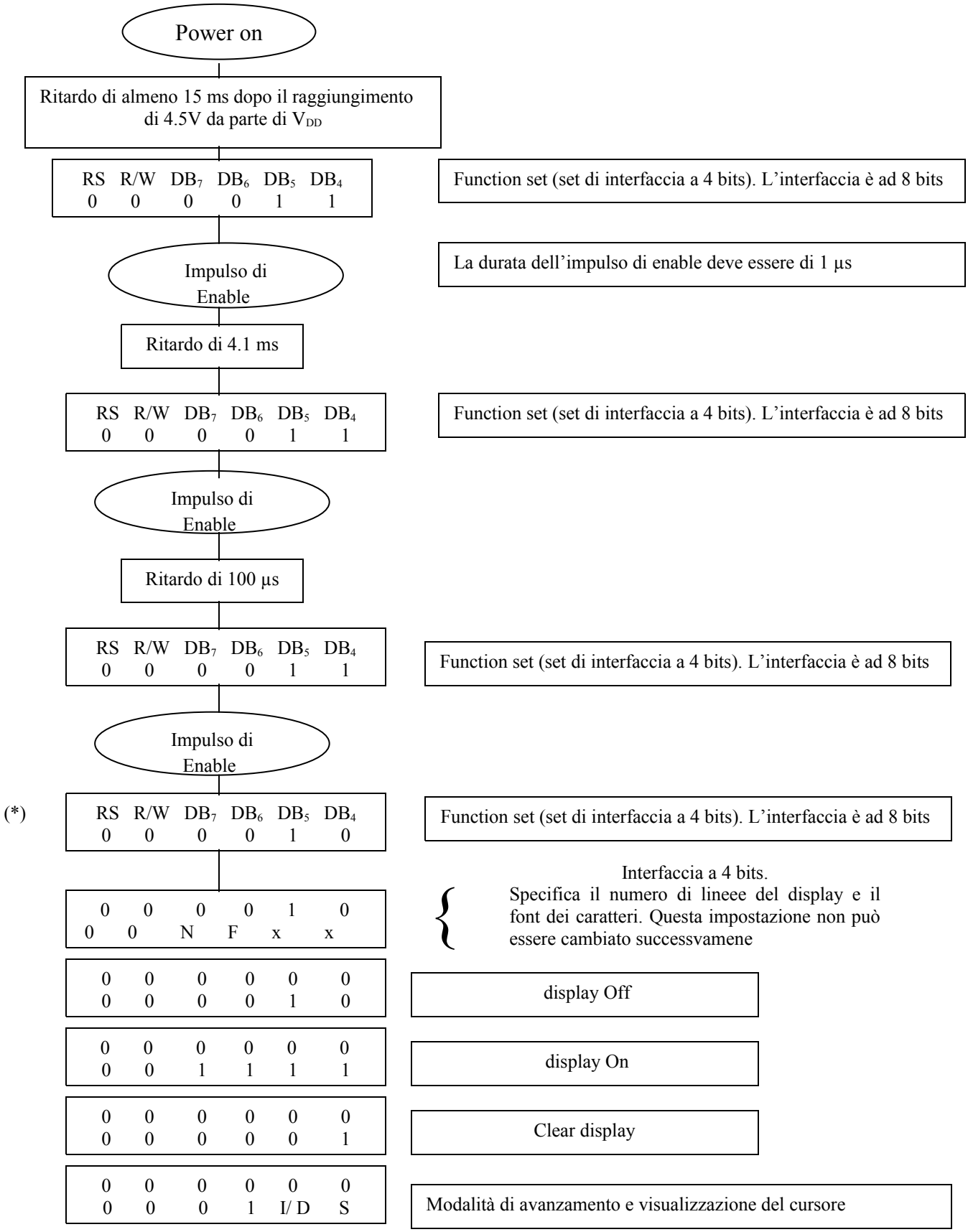
All'accensione il circuito di controllo è resettato e sono impostate alcune modalità di funzionamento. Ad esempio la più ovvia delle impostazioni è che lo schermo è cancellato all'accensione per evitare di mostrare caratteri non voluti. Altre impostazioni riguardano la posizione del cursore che di solito è nel primo carattere della prima riga, oppure il movimento del cursore verso destra quando si scrivono più caratteri in sequenza.

Perchè l'impostazione iniziale sia fatta correttamente è necessario che la tensione di alimentazione raggiunga il valore di regime in tempi sufficientemente brevi, di solito inferiori a 10 ms. In caso contrario la procedura di impostazione iniziale può non essere corretta.

Poiché non si può essere sempre sicuri che la tensione di alimentazione salga in modo sufficientemente veloce e a volte le impostazioni iniziali non rispondono alle nostre esigenze, conviene sempre effettuare impostare personalmente il funzionamento del modulo.

L'insieme di istruzioni che seguono sono un buon esempio di inizializzazione (sinonimo di impostazioni iniziali) che imposta, tra l'altro, l'interfaccia a 4 bit.

Diagramma di flusso per l'inizializzazione di un LCD con interfaccia a 4 bit



(*) continuare ad inserire degli impulsi di enable anche fra le istruzioni successive

1.6 SET di ISTRUZIONI

tabella 3

Istruzione	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Descrizione	Durata con f_{osc} = 270 kHz
Pulisci Display	0	0	0	0	0	0	0	0	0	1	Pulisce il display e riporta il cursore nella posizione 0.	1,64 ms
Cursore in posizione 0	0	0	0	0	0	0	0	0	1	X	Riporta il cursore nella posizione 0. Il display è traslato nella posizione originale (il contenuto DDRAM non cambia).	1,64 ms
Set Mode	0	0	0	0	0	0	0	1	I/D	S	Imposta la direzione del cursore (I/D) e specifica lo shift del display(S). Queste operazioni vanno eseguite durante la lettura/scrittura	40 µs
Controllo ON/OFF display	0	0	0	0	0	0	1	D	C	B	Setta ON/OFF il display (D), ON/OFF il cursore (C) e ON/OFF il lampeggio del cursore (B).	40 µs
Cursore / Traslazione	0	0	0	0	0	1	S/C	R/L	X	X	Imposta il movimento del cursore o la traslazione del display (S/C) e la direzione (R/L). Il contenuto DDRAM non varia.	40 µs
Function Set	0	0	0	0	1	DL	N	F	X	X	Imposta il numero di bit dell'interfaccia, la lunghezza delle righe e i fonts.	40 µs
Imposta indirizzo CGRAM	0	0	0	1	Indirizzo CGRAM						Imposta l'indirizzo CGRAM. I dati CGRAM possono essere ricevuti e inviati dopo questa istruzione.	40 µs
Imposta indirizzo DDRAM	0	0	1	Indirizzo DDRAM							Imposta l'indirizzo DDRAM. I dati DDRAM possono essere ricevuti e inviati dopo questa istruzione.	40 µs
Lettura busy Flag e contatore indirizzo	0	1	BF	Indirizzo CGRAM / DDRAM							Leggi il busy flag (BF) che indica se l'istruzione è stata eseguita e leggi il contenuto del contatore dell'indirizzo CGRAM o DDRAM (dipende dall'istruzione precedente).	1 µs
Scrivi CGRAM o DDRAM	1	0	Dato da scrivere								Scrive un dato su CGRAM o DDRAM.	40 µs
Leggi CGRAM o DDRAM	1		Dato da leggere								Legge un dato da CGRAM o DDRAM	40 µs

Legenda :

DDRAM = Display Data RAM.
CGRAM = Character Generator RAM (memoria che contiene i caratteri).
Indirizzo DDRAM è la posizione del cursore.

Per il significato delle lettere **N, F, I/D e S** si veda la seguente tabella 4.

Tabella 4

Bit	Impostazioni	
I/D	0 = Decrementa posizione cursore	1 = Incrementa posizione cursore
S	0 = Nessuna traslazione display	1 = Traslazione display
D	0 = Display spento	1 = Display Acceso
C	0 = Cursore spento	1 = Cursore acceso
B	0 = Lampeggio cursore spento	1 = Lampeggio cursore acceso
S/C	0 = Sposta il cursore	1 = Trasla il display
R/L	0 = Trasla a sinistra	1 = Trasla a destra
DL	0 = Interfaccia a 4 bit	1 = Interfaccia a 8 bit
N	0 = 1/8 o 1/11 "Duty cycle" (1 riga)	1 = 1/16 "Duty cycle" (2 righe)
F	0 = 5x7 punti	1 = 5x10 punti
BF	0 = Disponibile	1 = Operazione in corso

1.7 Indirizzi DDRAM

DDRAM deriva da Display Data RAM, la memoria in cui sono contenuti i caratteri che sono visualizzati sullo schermo. Ad esempio per far comparire la lettera “A” nel posto riservato al terzo carattere sarà necessario scrivere il codice corrispondente alla lettera A nella locazione di memoria corrispondente al terzo carattere. Per semplificarne l’identificazione l’indirizzo della locazione di memoria corrisponde esattamente alla posizione del carattere nello schermo, cominciando a contare da sinistra, cioè il terzo carattere è scritto nella locazione n° 3 della DDRAM.

tabella 5 Display a 1 riga:

Dimensione Display	Posizione carattere (dec)	Indirizzo DDRAM (hex)
1x8	00 .. 07	00 .. 07
1x16	00 .. 15	00 .. 0F
1x20	00 .. 19	00 .. 13
1x24	00 .. 23	00 .. 17
1x32	00 .. 31	00 .. 1F
1x40	00 .. 39	00 .. 27

Nei visualizzatori a 2 righe le cose si complicano leggermente, in particolare per ottenere l’indirizzo del carattere della seconda riga bisogna aggiungere 40 (in esadecimale) alla posizione del carattere. Riprendendo l’esempio precedente il terzo carattere della seconda riga sarà memorizzato nella locazione 40 + 03 = 43 (in esadecimale).

tabella 6 Display a 2 righe:

Dimensione Display	Posizione carattere (dec)	Indirizzo DDRAM (hex)
2x16	00 .. 15	1° riga: 00 .. 0F (*) 2° riga: 40 .. 4F (*)
2x20	00 .. 19	1° riga: 00 .. 13 2° riga: 40 .. 53
2x24	00 .. 23	1° riga: 00 .. 17 2° riga: 40 .. 57
2x32	00 .. 31	1° riga: 00 .. 1F 2° riga: 40 .. 5F
2x40	00 .. 39	1° riga: 00 .. 27 2° riga: 40 .. 67

(*) Tenere conto del 1 logico del bit MSB nella riga relativa alla scrittura/lettura in DDRAM (tabella 3).

1.8 ESEMPI pratici di INTERFACCIA

È già stato accennato alla possibilità di interfacciare il visualizzatore ad un PIC 16F87X utilizzando solamente 6 linee: 4 bit per il bus dati, un bit di Enable ed un bit per R/S. Nella figura è riportato lo schema elettrico dei collegamenti tra microcontrollore e visualizzatore nella scheda usata per le esercitazioni in laboratorio di TDP.
Si può vedere che il bit R/W (piedino 5 del LCM) è collegato direttamente a massa (cioè LCM in lettura) perché scrivendo opportunamente il programma è possibile gestire le temporizzazioni internamente al PIC senza dover leggere col PIC il "busy flag" (BF) del display.

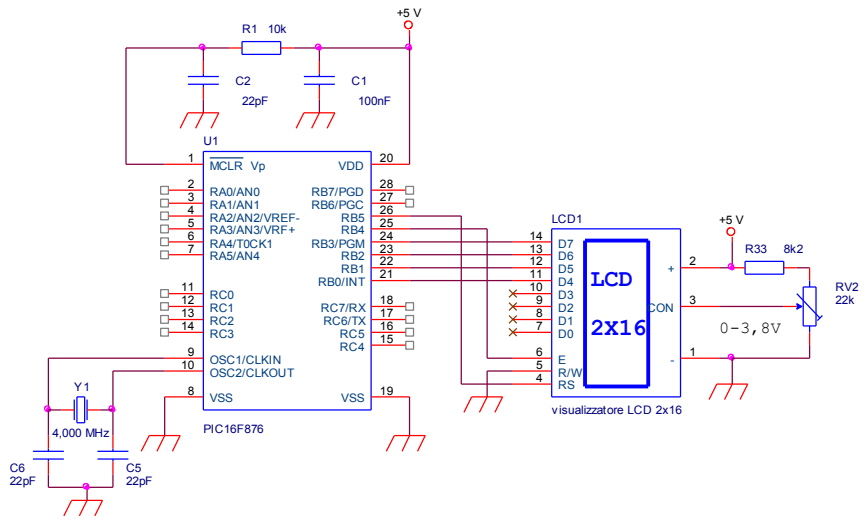


Figura 2 – collegamenti tra µC e LCD nella demoboard

Il circuito formato da R33 e RV2 fornisce una tensione variabile tra 0 e 3,8 V al terminale di regolazione del contrasto (3 del LCD) e permette di scurire a piacere lo sfondo su cui deve risaltare il carattere. Il trimmer è stato collegato in modo che ruotandolo in senso orario lo sfondo diventi più scuro.

A titolo di esempio sono riportati programmi in linguaggio assembly del PIC 16F876, usati nella scheda per esercitazioni, che svolgono alcune delle funzioni principali: l'impostazione iniziale del LCM, l'invio di un dato o di un'istruzione usando 4 bit, la scrittura di un carattere sullo schermo.

CONNESSIONE UNIVERSALE

La disposizione dei terminali varia da un costruttore all'altro e può succedere che uno stampato, progettato per un LCM sia incompatibile con un LCM di un produttore diverso. Per eliminare i problemi di incompatibilità si è deciso di adottare uno standard che risolva definitivamente il problema. Infatti nel laboratorio di TDP il LCM è montato su un piccolo stampato che ospita anche un connettore per cavo piatto da 16 poli, un connettore analogo è previsto sullo stampato che ospita il microcontrollore e il collegamento tra i due circuiti è affidato ad un cavo piatto a 16 poli con due connettori.

Pin	Simbolo	Funzione
1	+	positivo alimentazione (+5V)
2	-	negativo alimentazione (GND)
3	CON	Regolazione contrasto
4	D/I	Dati / Istruzioni
5	R/W	Read / Write
6	E	Enable
7	D0	Linea 0 bus dati (LSB)
8	D1	Linea 1 bus dati
9	D2	Linea 2 bus dati
10	D3	Linea 3 bus dati
11	D4	Linea 4 bus dati
12	D5	Linea 5 bus dati
13	D6	Linea 6 bus dati
14	D7	Linea 7 bus dati (MSB)
15	A	positivo illuminazione (Anodo dei LED)
16	C	negativo illuminazione (Catodo dei LED)

La piedinatura del connettore è sempre la stessa, indipendente dal LCM. La compatibilità è affidata al piccolo stampato che ospita il LCM e che è fatto su misura per ogni tipo di modulo. Un ulteriore vantaggio di questa soluzione è che la posizione dello stampato principale è del tutto indipendente da quella dello schermo, che di soito è montato su una delle pareti del contenitore dell'apparecchiatura.
La piedinatura scelta per il connettore, da montare sullo stampato che ospita il microcontrollore, è riportata in tabella.

Esempio di codifica, in linguaggio assembly della Microchip, dell'algoritmo di inizializzazione di un display rappresentato dal flow-chart di pag. 5 : count1 e count2 sono due variabili usate dalla routine di ritardo (delay). L'algoritmo è comprensivo del controllo della linea RS di selezione istruzioni/caratteri inviati al display.

```
INIDSP  BCF     PORTB,RS        ;INVIO DI ISTRUZIONI AL DISPLAY: RS A LIVELLO L
        BCF     PORTB,E         ;EN A LIVELLO L
        MOVLW   0FFH
        MOVWF   COUNT1          ;CARICO COUNT1 CON FF HEX
        MOVLW   028H
        MOVWF   COUNT2          ;CARICO COUNT2 CON 28 HEX
        CALL    DELAY           ;RITARDO DI 30ms PER SETUP DI Vdd
        MOVLW   033H            ;SET DI INTERFACCIA A 4 BITS
        CALL    VISBYTE
        MOVLW   032H
        CALL    VISBYTE
        MOVLW   028H            ;2 LINEE- 5X7 CHR. FONT
        CALL    VISBYTE
        MOVLW   08H             ;TURN OFF DISPLAY AND CURSOR
        CALL    VISBYTE
        MOVLW   0FH             ;TURN ON DISPLAY AND CURSOR
        CALL    VISBYTE
        MOVLW   06H             ;SHIFT AUTOMATICO A DESTRA
        CALL    VISBYTE
        MOVLW   01H             ;CLR DISPLAY
        CALL    VISBYTE
        BSF     PORTB,RS        ;INVIO DI ISTRUZIONI AL DISPLAY: RS A LIVELLO H
        RETURN
```

Esempio di codifica, in linguaggio assembly della Microchip, dell'algoritmo di invio di caratteri/istruzioni ad un display LCD, nel caso di interfacciamento su bus a 4 bits.

```
VISBYTE MOVWF   CHAR            ;SALVO IL BYTE ISTRUZIONE O CARATTERE DA INVIARE AL LCD IN UNA
        MOVLW   02              ;VARIABILE DI APPOGGIO 'CHAR'
        MOVWF   LOOP2           ;CARICO 2 IN LOOP2 PERCHÉ DUE SONO I NIBBLE DA INVIARE
NIBBLE2 SWAPF   CHAR,F          ;SCAMBIO I DUE NIBBLE DI CHAR
        MOVLW   B'11110000'     ;AZZERO IL NIBBLE BASSO PER CONSERVARE
        ANDWF   PORTB,F         ;GLI STATI DI E,RS,B6 E B7: PORTB → DDDD0000
        MOVLW   B'00001111'     ;AZZERO IL NIBBLE ALTO DELLA VARIABILE CHAR SWAPPATA
        ANDWF   CHAR,W          ;CHAR→ 0000DDDD
        IORWF   PORTB,F         ;OR LOGICO TRA IL NIBBLE ALTO DI PORTB E QUELLO BASSO DI CHAR
        BSF     PORTB,E         ;SET DI ENABLE
        NOP                     ;IL DATA-SHEET RICHIEDE UN Enable cycle time > 1 µs.
        BCF     PORTB,E         ;CLR DI ENABLE PER L'ACQUISIZIONE DEL NIBBLE BASSO
        MOVLW   0FFH
        MOVWF   COUNT1
        MOVLW   28H
        MOVWF   COUNT2          ;ATTESA PER ACQUISIZIONE DI 6 ms POICHE' IL
        CALL    DELAY           ;RITARDO OPPORTUNO E' DI ALMENO 5ms
        DECFSZ  LOOP2,1
        GOTO    NIBBLE2
        RETURN
```

Esempio di codifica, in linguaggio assembly della Microchip, dell'algoritmo di un programma principale per la visualizzazione di caratteri su di un display LCD.

```
START   BSF     STATUS,RP0
        BCF     STATUS,RP1      ;SELEZIONO IL BANCO 1 DI MEMORIA
        MOVLW   B'11000000'
        MOVWF   TRISB           ;DIREZIONAMENTO IN USCITA DEI DEI BITS 0-5 DEL PORTB
        BCF     STATUS,RP0      ;RITORNO AL BANCO 0 DI MEMORIA
        CALL    INIDSP          ;CHIAMATA ROUTINE DI INIZIALIZZAZIONE DEL DISPLAY
        MOVLW   'S'
        CALL    VISBYTE         ;CHIAMATA DELLA ROUTINE DI VISUALIZZAZIONE DEI CARATTERI
        MOVLW   'E'
        CALL    VISBYTE
        MOVLW   'V'
        CALL    VISBYTE
        MOVLW   'E'
        CALL    VISBYTE
        MOVLW   'R'
        CALL    VISBYTE
        MOVLW   'I'
LOOP
```

```
GOTO  LOOP
```

Queste tabelle, ognuna individuata da un codice di due cifre, mostrano l'associazione tra simbolo e codice. Si vede facilmente che i 4 bit più significativi individuano la colonna, i 4 meno significativi la riga. Da notare come allo stesso codice (e quindi alla stessa posizione) in generale corrispondano caratteri diversi nella diverse tabelle perciò è fondamentale conoscere quale tabella è memorizzata nel LCM che si sta usando.

HD66712U – Relationship between Character Codes and Character Pattern (ROM Code: A00).

Lower Bits \ Upper Bits	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
xxxx0000	CG RAM (1)			0	@	P	`	p				—	夕	ミ	α	p
xxxx0001	CG RAM (2)		!	1	A	Q	a	q			。	ア	チ	ム	ä	q
xxxx0010	CG RAM (3)		"	2	B	R	b	r			「	イ	ツ	メ	ß	θ
xxxx0011	CG RAM (4)		#	3	C	S	c	s			」	ウ	テ	モ	ε	∞
xxxx0100	CG RAM (5)		$	4	D	T	d	t			、	エ	ト	ヤ	μ	Ω
xxxx0101	CG RAM (6)		%	5	E	U	e	u			・	オ	ナ	ユ	σ	ü
xxxx0110	CG RAM (7)		&	6	F	V	f	v			ヲ	カ	ニ	ヨ	ρ	Σ
xxxx0111	CG RAM (8)		'	7	G	W	g	w			ァ	キ	ヌ	ラ	q	π
xxxx1000	CG RAM (1)		(	8	H	X	h	x			ィ	ク	ネ	リ	ﾞ	x̄
xxxx1001	CG RAM (2)		)	9	I	Y	i	y			ゥ	ケ	ノ	ル	ﾟ	y
xxxx1010	CG RAM (3)		*	:	J	Z	j	z			ェ	コ	ハ	レ	j	千
xxxx1011	CG RAM (4)		+	;	K	[	k	{			ォ	サ	ヒ	ロ	x	万
xxxx1100	CG RAM (5)		,	<	L	¥	l	|			ャ	シ	フ	ワ	¢	円
xxxx1101	CG RAM (6)		-	=	M	]	m	}			ュ	ス	ヘ	ン	£	÷
xxxx1110	CG RAM (7)		.	>	N	^	n	→			ョ	セ	ホ	ﾞ	ñ	
xxxx1111	CG RAM (8)		/	?	O	_	o	←			ッ	ソ	マ	ﾟ	ö	■

Tabella 8 – corrispondenza tra codici e caratteri visualizzati.

HD66712U - Relationship between Character Codes and Character Pattern (ROM Code: A01)

Tabella 9 – corrispondenza tra codici e caratteri visualizzati.

HD66712U - Relationship between Character Codes and Character Pattern (ROM Code: A02)

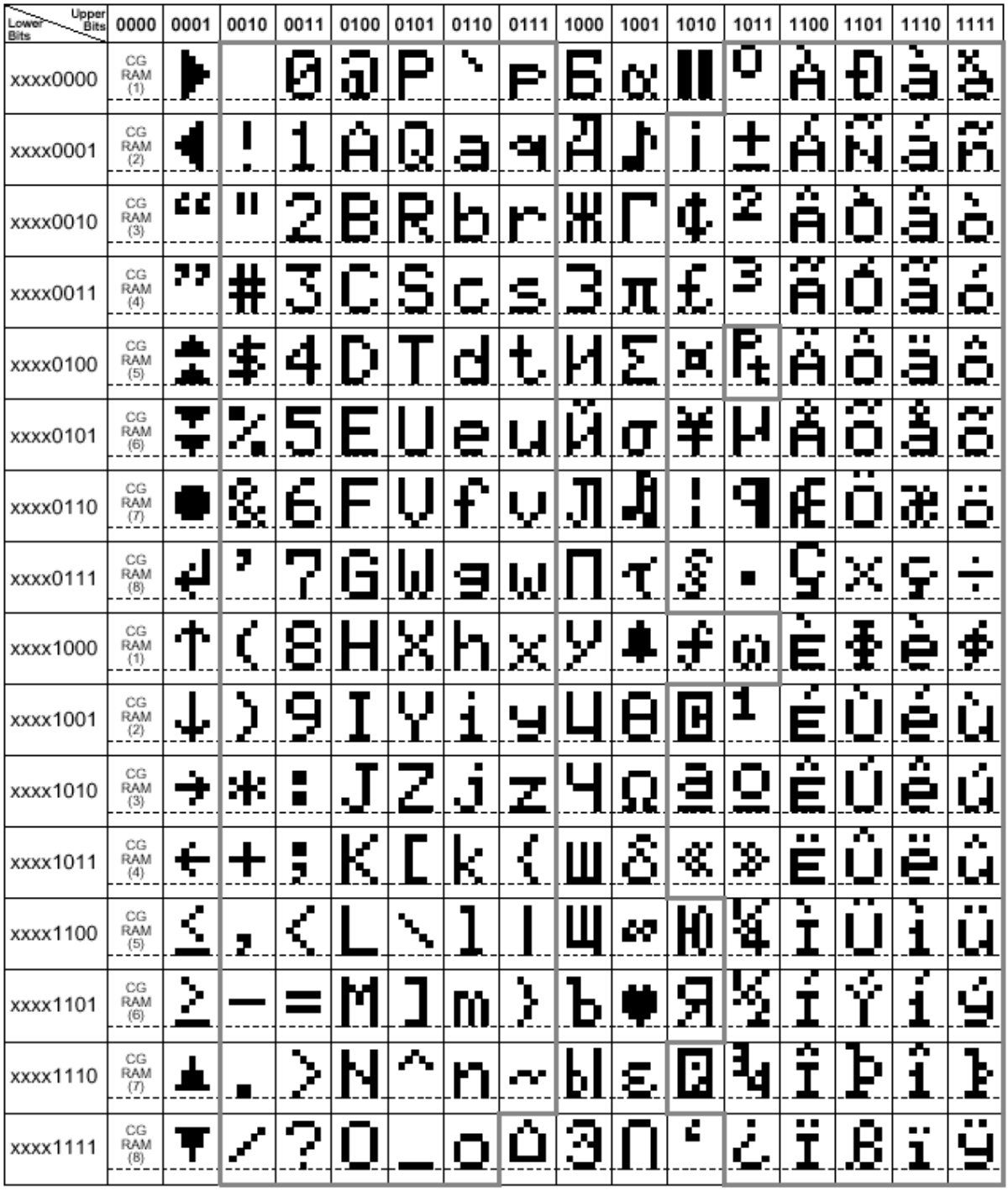


figura 10 – corrispondenza tra codici e caratteri visualizzati.

HD66712U - Relationship between Character Codes and Character Pattern (ROM Code: A03)

figura 11 – corrispondenza tra codici e caratteri visualizzati.