628Board

Scheda multi uso per il PIC 16F628

di Roberto D'Amico - IW0GTF

DISCLAIMER

Stato del documento

revisione	data	sintesi dei cambiamenti
1.0	14/03/2005	Prima stesura

Sintesi dei cambiamenti

lista dei principali cambiamenti rispetto la revisione precedente:	

SOMMARIO

1 Scopo del documento

Questo documento vuole descrivere le caratteristiche della scheda 628Board, e spiegare come poter utilizzare le funzionalità di base offerte dalla scheda stessa. E' necessaria la conoscenza, o la necessità di documentarsi da altre fonti, di alcune tecniche necessarie a realizzare la scheda o per la programmazione del microcontrollore.

1.1 Struttura del documento

Il documento è strutturato nei seguenti capitoli il cui contenuto può essere riassunto come segue:

Introduzione: il presente capitolo;
Che cos'è la 628Board: descrive le funzionalità della scheda;
Utilizzare la 628Board: descrive come utilizzare la scheda;

1.2 File allegati

File contenuti nell'archivio 628Board.zip:

628Board.pdf - Questo documento
Nel folder CIRCUITO:
628Board_pcb.pdf - File PDF con il disegno del circuito stampato, può essere stampato per realizzare un lucido per la fotoincisione
628Board_sch.pdf - File PDF con il disegno dello schema elettrico
Nel folder CODE\LCD:
TESTLCD.C - Il nostro programma per il test del display
DMLCD2R.H - File include per la gestione di un display a matrice di punti
delay.c - File con le funzioni per l'include delay.h
delay.h - File include per la gestione dei delay
TESTLCD.HEX - Il programma compilato
Nel folder CODE\UART:
TESTUART.C - Il nostro programma per il test dell'UART
UART628.H - File include per la gestione dell'UART del PIC
DMLCD2R.H - File include per la gestione di un display a matrice di punti
delay.c - File con le funzioni per l'include delay.h
delay.h - File include per la gestione dei delay
TESTUART.HEX - Il programma compilato

2 Cos'è la 628BOARD?

2.1 Schema elettrico

Questo circuito multifunzione per il PIC 16F628, è stato progettato in modo da disporre come funzionalità di base la gestione di un display LCD a matrice di punti e la comunicazione seriale con un PC o altro dispositivo utilizzando l'USART interna del PIC.

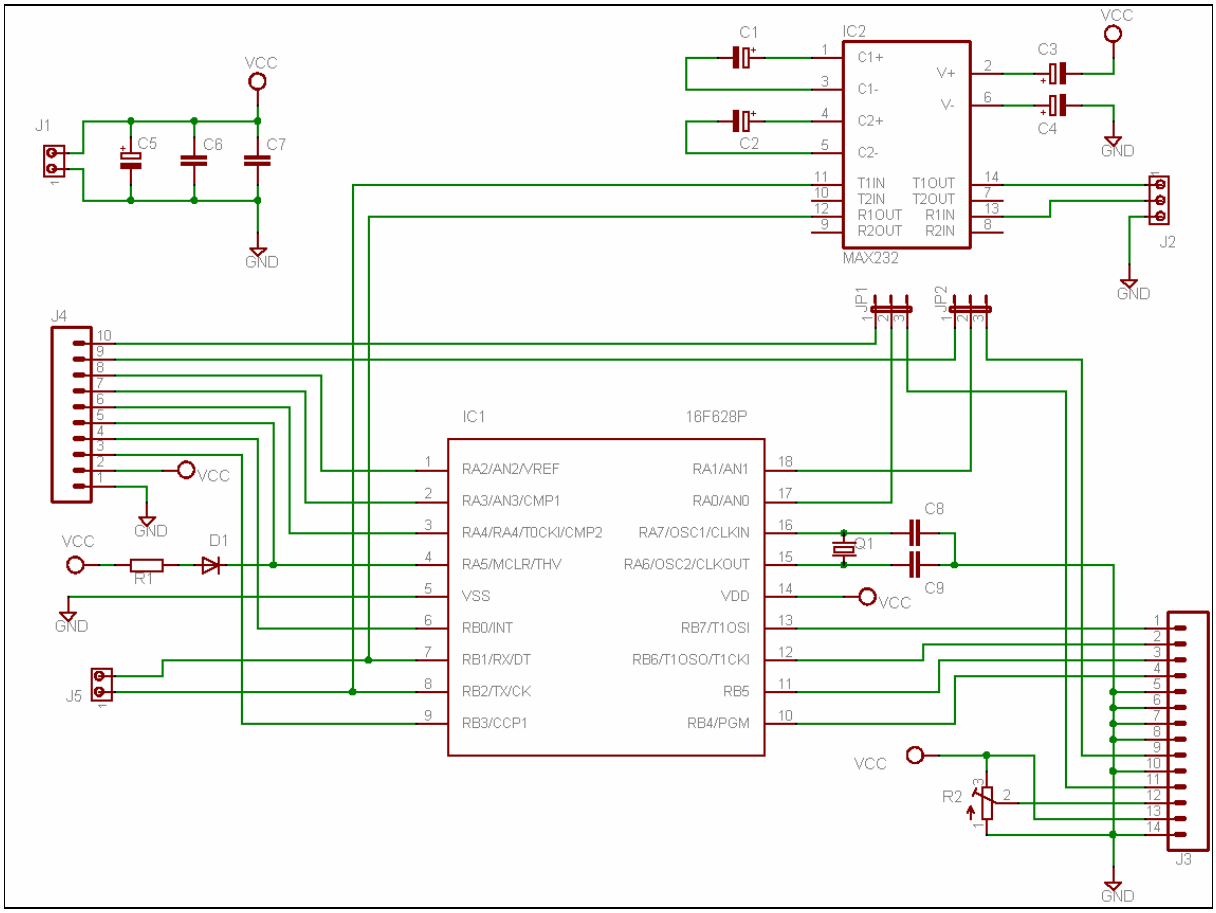

Figura 1 - Schema elettrico della 628Board

Come si può vedere dallo schema elettrico, il circuito è molto semplice, il cuore è un PIC 16F628 (da notare la possibilità di sostituire il PIC con un 16F627 o tendo conto delle differenze interne anche il 16F84) le cui porte RB4, RB5, RB6, RB7, RA0 e RA1, sono utilizzate per gestire il display LCD e riportate sul connettore J3, il quale è predisposto per poter collegare direttamente il display per il funzionamento con bus dati multiplexato. Sullo stampato è presente anche il trimmer per la regolazione del contrasto. Naturalmente nulla vieta che questo connettore venga usato per collegare altre periferiche diverse dal display, per questo motivo le porte RA0 e RA1 possono essere riportate tramite i jumper JP1 e JP2 sul connettore J3 o J4, in modo da dare la massima libertà di utilizzo. Le porte RB1 e RB2 sono utilizzate per la trasmissione dati, quindi collegate al MAX232 per adattare i livelli di tensione a quelli della seriale del computer. Sul connettore J5 sono inoltre riportate le porte RB1 e RB2, nel caso si vogliono usare per altri scopi. In fine il connettore J4 riporta tutte le porte non utilizzate, per comodità come gia detto anche le porte RA0 e RA1 e l'alimentazione per eventuali piccoli circuiti esterni.

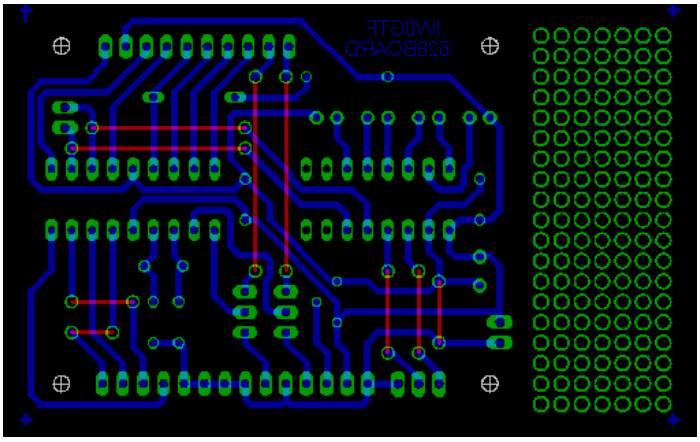

Figura 2 - Circuito stampato

Il circuito stampato è stato sviluppato per essere compatto, ma facilmente realizzabile con strumenti "casalinghi", quindi singola faccia, piste di una certa dimensione che non passano tra i pin degli integrati, sono presenti 9 ponticelli (vedi collegamenti rossi in Figura 2) per realizzare i collegamenti dove non vi era altra soluzione. Inoltre lo stampato presenta una piccola zona mille fori dove è possibile realizzare un piccolo circuito aggiuntivo. Sul circuito non è presente un connettore seriale standard a 9 poli, per permettere a tutti di poterlo montare dove si vuole senza alcun tipo di vincolo, come non è stato inserito nessun circuito per l'alimentazione, che volendo è possibile realizzare con un comunissimo 7805 nell'area forata dello stampato.

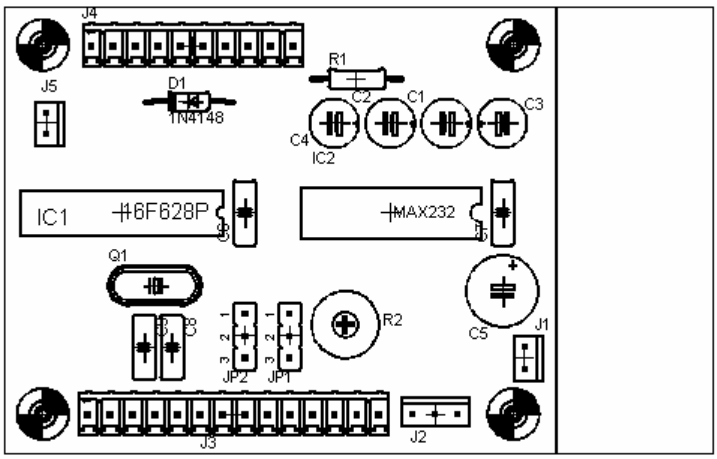

Figura 3 - Serigrafia del circuito

2.2 Elenco componenti

Elenco dei componenti per la realizzazione del circuito:

R1=2,2K ohm ¼ watt
R2=1K trimmer
C1=10µF 16 V elettrolitico
C2=10µF 16V elettrolitico
C3=10µF 16V elettrolitico
C4=10µF 16V elettrolitico
C5=100µF 16V elettrolitico
C6=100nF poliestere
C7=100nF poliestere
C8=22pF disco (x quarzo a 4MHz)
C9=22pF disco (x quarzo a 4MHz)
D1=1N4148
Q1=4 MHz quarzo (fino a 20 MHz)
IC1=PIC16F628 o equivalente
IC2=MAX232 o equivalente
J1= connettore 2 pin
J2= connettore 3 pin
J3= connettore 14 pin
J4= connettore 10 pin
J5= connettore 2 pin
JP1=Jumper a 2 vie
JP2=Jumper a 2 vie

2.3 Montaggio del circuito

Per prima cosa cominciate a realizzare i ponticelli (vedi collegamenti in rosso riportati in Figura 2), consiglio di utilizzare del cavo elettrico tipo doppino telefonico, quello con un solo filo rigido, mantenendo la gomma isolante nella parte che non viene saldata. Si continua con il montaggio della resistenza R1, poi collegate il diodo D1 facendo attenzione alla fascia colorata che è in direzione opposta ad R1 (come visibile in Figura 3). Procedete con il montaggio dei connettori e dei due jumper e il trimmer R2. In seguito montate i condensatori, facendo attenzione alla polarità di quelli elettrolitici. I condensatori elettrolitici sono C10 che va montato tenendo la fascia che indica il negativo verso il connettore dell'alimentazione, C1, C2, C4 vanno montati con il positivo verso C3 e questo ultimo va montato con il positivo verso i precedenti. Non resta che saldare gli zoccoli per gli integrati o gli integrati stessi sul circuito e la scheda è pronta.

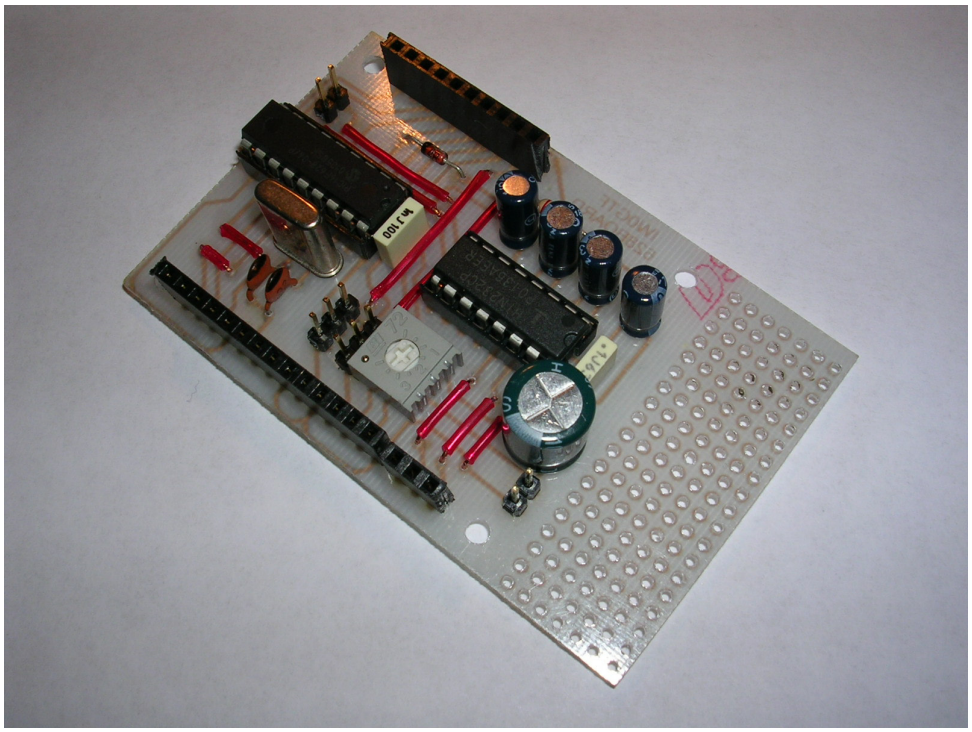

Figura 4 - 628Board ultimata

2.4 In-Circuit Programmer

Alcuni programmatori permettono la programmazione "In-Circuit" dei PIC, questa funzionalità permette, grazie all'utilizzo di un semplice connettore di programmare il microcontrollore, senza necessariamente toglierlo ogni volta dalla scheda e metterlo nel programmatore. La 628Board non prevede un connettore specifico, ma utilizzando i connettori presenti e possibile utilizzare comunque questa funzionalità. Sarà sufficiente realizzare un cavo adattatore per il vostro programmatore, nella Figura 5 sono indicati i pin dei connettori con i relativi segnali che devono essere utilizzati per la programmazione In-Circuit, non usare un cavo più lungo di 20 cm.

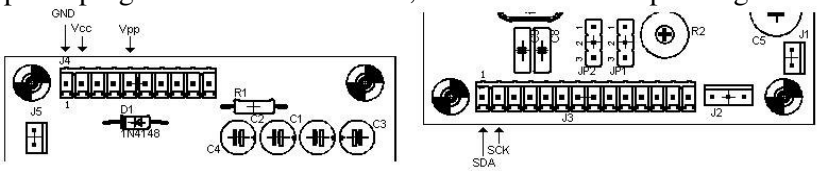

Figura 5 - Segnali sui connettori J4 e J3 per ICSP

3 Utilizzare la 628Board

3.1 Il 16F628

Come già detto su questa scheda è possibile montare anche altri microcontrollori della famiglia PIC equivalenti al 16F628, ma visto che la scheda è stata progettata su questo in particolare si parlerà in seguito sempre di questo modello. Sul datasheet del PIC è possibile trovare tutte le informazioni di cui si ha bisogno, in questo documento si vuole spiegare solo come utilizzare il 628 senza entrare troppo nel dettaglio, quindi cominceremo dando solo alcune informazioni basilari per utilizzarlo.
Tra le funzioni di questo PIC, ci sono i comparatori analogici, che al reset risultano abilitati, quindi se si vogliono utilizzare le porte da RA0 a RA3 come comuni porte digitali è necessario disattivare i comparatori. Per fare questo è sufficiente che nel registro CMCON venga inserito il valore 0x07.
Se si vuole utilizzare la porta RB4 come output è necessario che quando programmate il PIC disabilitiate la programmazione "Low Voltage", altrimenti il pin viene settato automaticamente come ingresso.

3.2 Il display LCD

Ora vedremo come gestire un display LCD a matrice di punti. La maggior parte di questi display utilizza lo stesso protocollo di trasmissione e lo stesso tipo di interfaccia sviluppata dalla Hitachi, che prevede la connessione ad un microcontrollore con un bus dati da 8 o 4 bit e 3 ulteriori line di controllo. Si possono trovare display con diverse caratteristiche di righe e colonne quelle tipiche sono 2x16 o 2x20, ma è possibile trovarne anche di altri tipi come 1x20, 1x16, 2x8, ecc. Fate attenzione in quanto a volte display tipo quelli 1x16 vengono gestiti come se fossero dei 2x8, quindi per scrivere sul secondo gruppo di 8 caratteri è necessario spostarsi alla seconda riga.
La piedinatura standard è quella riportata in tabella, nel nostro caso il circuito è stato realizzato (vedi J3) per utilizzare solo 4 bit per i dati e due segnali di controllo EN e RS, gli altri pin sono collegati a massa, al display è collegato anche un trimmer che serve per la regolazione del contrasto del display. Se disponete di un display retro illuminato questo presenta 16 pin invece di 14, in questo caso verificate correttamente la piedinatura e nei pin restanti potete collegare l'alimentazione come indicato dal datasheet del display. Fate attenzione, la piedinatura del connettore J3 è invertita rispetto a quella del display.

PIN	NOME	DESCRIZIONE
1	GND	Questo pin deve essere collegato a massa
2	Vcc	Questo pin deve essere collegato a +5V dell'alimentazione
3	LCD	Su questo pin deve essere collegata una tensione variabile tra 0 e 5 volt per regolare il contrasto (vedi R2)
4	RS	Questa linea di controllo (Register Select) serve a indicare se il byte che si sta inviando è un dato (RS=1) o un comando (RS=0)
5	R/W	Questa linea di controllo serve ad indicare se si vuole inviare un byte al display (R/W=0) o leggere dal display (R/W=1). Nel nostro caso è collegato direttamente a massa perché non andremo mai a leggere dal display.
6	E	Questa linea di controllo serve ad abilitare (E=1) il display ad accettare dati.
7	DB0	Data bus line 0 – Non usato nel nostro caso
8	DB1	Data bus line 1 – Non usato nel nostro caso
9	DB2	Data bus line 2 – Non usato nel nostro caso
10	DB3	Data bus line 3 – Non usato nel nostro caso
11	DB4	Data bus line 4
12	DB5	Data bus line 5
13	DB6	Data bus line 6
14	DB7	Data bus line 7

Tabella 1 - Piedinatura del display LCD

3.3 L'header file DMLCD2R.H

Prima di vedere il primo programma, analizziamo l'header file per la gestione del display. Questo file contiene le definizioni delle funzioni e delle costanti, necessarie per la gestione del display, da questo file analizzeremo i comandi principali necessari per pilotarlo. Per poter utilizzare il display è necessario inizializzarlo, utilizzando la procedura *lcd_init()* (vedi Listato 1).

```
void lcd_init(void)
{
      LCD_RS=LCD_CMD;
      DelayMs(15);            // Ritardo per l'accensione del display
      LCD_PORT_DATA=(LCD_PORT_DATA & 0x0F) | 0x30;
      lcd_strobe();
      DelayMs(5);
      lcd_strobe();
      DelayUs(100);
      lcd_strobe();
      DelayMs(5);

      LCD_PORT_DATA=(LCD_PORT_DATA & 0x2F) | 0x20;    // set 4 bit mode

      lcd_strobe();
      DelayUs(40);

      // 4 bit mode, 2 or more lines, 5x8 font
      lcd_write(LCD_CMD_FS | LCD_FS_DL4 | LCD_FS_NRM | LCD_FS_F5X7);
      // display off
      lcd_write(LCD_CMD_DC | LCD_DC_DISPLAY_OFF | LCD_DC_CURSOR_OFF |
LCD_DC_CURBLINK_OFF);
      // display on, cursor off, blink off
      lcd_write(LCD_CMD_DC | LCD_DC_DISPLAY_ON | LCD_DC_CURSOR_OFF |
LCD_DC_CURBLINK_OFF);
      // shift entry mode, display not shifted
      lcd_write(LCD_CMD_EMS | LCD_EMS_INCREASE | LCD_EMS_SHIFT_OFF);
}
```

Listato 1 - Procedura per inizializare il display

Analizziamo il codice del Listato 1, la prima riga serve per mandare a 0 la linea di controllo RS del display, in questo modo gli stiamo dicendo che i byte che arriveranno sono dei comandi, prima di cominciare con l'invio dei byte è necessario attendere almeno 15ms affinché il display si sia acceso correttamente e pronto a ricevere i nostri comandi. La procedura indicata nel datasheet prevede che il canale dati sia configurato nel modo indicato, mandando un segnale di abilitazione ad intervalli regolari in fine si pone un valore che indica il numero di bit utilizzati.
Le righe che richiamano la procedura *lcd_write()* servono ad indicare al display la modalità di funzionamento in cui si vuole impostarlo, nel nostro caso abbiamo:

- Bus Dati a 4 bit
- LCD multi linea
- Font 5x7
- Modalità inserimento carattere a successivo

Quelli che vengono inviati in questo momento sono dei veri e propri comandi per la gestione del display, il primo comando è il "Function Set" e serve ad indicare il numero di bit usati, il numero di righe e il formato del font. Il secondo comando serve a spegnere il display, che viene riacceso con il terzo comando. Il quarto infine è l'"Entry Mode Set", che determina il modo di inserimento dei caratteri e se attivare o no lo shift di questi.

La procedura *lcd_write()* serve a dividere il byte da inviare in gruppi di 4 bit.

```
void lcd_write(unsigned char c)
{
      LCD_PORT_DATA=(LCD_PORT_DATA & 0x0F) | (c & 0xF0);
      lcd_strobe();
      LCD_PORT_DATA=(LCD_PORT_DATA & 0x0F) | (c << 4);
      lcd_strobe();
      DelayUs(50);
}
```

Listato 2 - Procedura per l'invio di byte al display

Come si vede nel Listato 2, la procedura prende lo stato della porta a cui è collegato il display, nel nostro caso PORTB i quattro bit più significativi, vengono azzerati i bit usati per inviare i dati e tramite la funzione di OR viene aggiunta a questo valore la parte alta del byte da inviare per prima, la parte bassa viene filtrata tramite lo shift del byte e inviata col secondo gruppo di bit.
Le procedure *lcd_putch()* e *lcd_puts()* servono rispettivamente ad inviare uno o più caratteri contemporaneamente al display. Da notare che in queste funzioni il bit LCD_RS è settato a 1 (LCD_DATA), questo per indicare al display che si sta inviando un dato che dovrà essere visualizzato.
La procedura *lcd_pos_cur()* serve ad indicare il punto in cui si vuole posizionare il cursore, per fare questo è sufficiente inviare al display l'indirizzo della posizione. La Tabella 2 rappresenta gli indirizzi di un tipico display 16x2, fate attenzione perché gli indirizzi sono scritti in esadecimale.

		COLONNE															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
RIGHE	**0**	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
	1	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

Tabella 2 - Indirizzi di un Display 16x2

Se volessimo cominciare a scrivere alla 7° colonna e 2° riga, dovremmo inviare l'indirizzo 0x46. Ora vediamo nello specifico come, per fare questo per prima cosa convertiamo il valore esadecimale in binario, quindi 0x46=0b01000110, questo valore deve essere messo in OR con il comando LCD_SET_ADDRESS (0b10000000) e inviato al display. Ricordate sempre che la funzione *lcd_pos_cur()* richiede come indice iniziale sia per la riga che per la colonna 0, quindi ritornando all'esempio di prima i parametri da passare alla funzione sono quelli indicati di seguito:

```
lcd_pos_cur(1,6);
```

Altra procedura molto utile è *lcd_clear()*, questa invia semplicemente il comando al display per cancellare tutto ciò che viene visualizzato e riporta il cursore all'indirizzo 0x00, il classico home dei computer.
Esiste inoltre la procedura lcd_home(), che riporta il cursore nella posizione home senza però cancellare ciò che è visualizzato sul display.

3.4 Un programma di prova

Vediamo ora un semplice programma di esempio per utilizzare le funzioni descritte.

```
#include <pic.h>
#include "delay.c"
#include "DMLCD2R.h"

__CONFIG (0x3F61);            // Configurazione del programmatore

void main(void)
{
      CMCON=0x07;             // Disabilita i comparatori sulla PORTA
      PORTA=0x00;             // Azzera PORTA
      PORTB=0x00;             // Azzera PORTB
      TRISA=0x00;             // Setta tutte le porte di A come out
      TRISB=0x00;             // Setta tutte le porte di B come out

      lcd_init();             // Inizializza il display
      lcd_clear();            // Cancella il display
      lcd_puts("     Hello"); // Scrive una stringa
      lcd_pos_cur(1,5);       // Si posiziona sulla riga 2 colonna 6
      lcd_puts("World!");     // Scrive una stringa

      for(;;);                // Loop infinito
}
```

Listato 3 - Programma di test per il display (TESTLCD.C)

Analizziamo il programma (Listato 3), le prime righe si riferiscono agli include, in particolare quella relativa a DMLCD2R.H, che contiene le funzioni per gestire il display.
Come indicato nel paragrafo "3.1 Il 16F628", è necessario disattivare i comparatori inserendo nel registro CMCON il valore 0x07. Continuiamo azzerando PORTA e PORTB e settando tutte le porte come output. Ora non bisogna fare altro che inizializzare il display con la funzione *lcd_init()*, cancelliamone il contenuto con *lcd_clear()*, ora si può scrivere sul display usando la procedura *lcd_puts()*. Notate che è stato usato anche *lcd_pos_cur()* per posizionarsi sulla seconda riga. Il ciclo *for* alla fine, ha una struttura un po' strana, che ai meno esperti può sembrare un errore, serve ad eseguire un ciclo infinito, altrimenti il PIC proseguirebbe a leggere il programma in memoria fino alla fine e ricomincerebbe da capo.
Nella foto che segue potete vedere il risultato finale del nostro programma.

Figura 6 - Il programma di esempio in esecuzione

3.5 16F628 e la sua UART

Il PIC 16F628 contiene al suo interno anche un circuito USART per la trasmissione seriale sincrona o asincrona. Nel nostro caso useremo la comunicazione asincrona, quindi da ora in poi parleremo semplicemente di UART, per poter comunicare con un personal computer tramite porta seriale. Grazie al circuito interno il PIC riesce a gestire la comunicazione autonomamente, quindi non è necessario indicargli ogni singolo bit da trasmettere, ma è sufficiente configurare in maniera opportuna alcuni registri.
Per prima cosa è necessario configurare in maniera appropriata il registro SPBRG che serve per impostare il timer che gestisce le temporizzazioni per la comunicazione. Il valore da inserire in questo registro dipende dal baud rate che si richiede, dalla frequenza di funzionamento del PIC, dallo stato dei bit BRGH e SYNC del registro TXSTA. Sul datasheet è possibile trovare una formula per il calcolo di questo valore e una tabella con i valori tipici utilizzati.
Di seguito si devono configurare i registri TXSTA e RCSTA, si consiglia di concludere svuotando il registro RCREG che funziona da buffer di 3 byte per la ricezione.

3.6 L'header file UART628.H

Analizziamo ora l'header file che contine le funzioni per gestire l'UART interna del PIC.
Come già visto nel paragrafo precedente la prima cosa da fare è richiamare la procedura *uart_init()* che serve per abilitare la comunicazione seriale.

```
void uart_init()
{
unsigned char i;
unsigned char buffer;

      SPBRG=BRG_9600_4;       // Baud Rate Generator – BRGH=1,Fosc=4,SYNC=0
      TXSTA=0x24;             // Setta il registro TXSTA
      RCSTA=0x90;             // Setta il registro RCSTA

      buffer=RCREG;           // Svuota il buffer di ricezione
      buffer=RCREG;
      buffer=RCREG;
}
```

Listato 4 - Procedura per inizializzare l'UART del PIC

Come già visto la prima cosa da fare è impostare il valore del registro SPBRG, nell'esempio del Listato 4 il valore è stato calcolato per una velocità di 9600 baud con un quarzo di 4MHz. Nel caso utilizziate un 16F628 con quarzo di diverso valore o vogliate un baud rate differente è necessario cambiare questo parametro. I registri TXSTA ed RCSTA sono impostati per funzionare con 8 bit di dati in modalità asincrona e per abilitare ricezione e trasmissione.
RCREG è un buffer a tre livelli dove viene memorizzato il contenuto dei byte ricevuti, in questa funzione viene memorizzato nella variabile *buffer* al solo scopo di svuotare i tre livelli del registro da eventuali dati incorretti dovuti all'inizializzazione. Ora il PIC è pronto ad inviare e ricevere dati.
La procedura *uart_putch()* serve ad inviare un byte, viene controllato che il buffer di trasmissione sia vuoto prima di inviare il nuovo byte. Quando il PIC è pronto a inviare un nuovo byte è sufficiente metterlo nel registro TXREG, ed il microcontrollore pensa al resto.

```
void uart_putch(unsigned char char_to_send)
{
      while(!TRMT);           // Polling per Tx buffer empty
      TXREG=char_to_send;
}
```

Listato 5 - Procedura per invio di un singolo byte

La procedura *uart_puts()* serve invece ad inviare una stringa intera, questa operazione viene fatta richiamando n volte la uart_putch() quanti sono i caratteri che compongono la stringa.
La funzione *uart_getch()* al pari della uart_putch() attende che il bit di controllo RCIF viene settato, questo indica che è stato ricevuto un byte, restituendo lo stato del registro RCREG, che contiene il byte ricevuto.

```
unsigned char uart_getch()
{
      while(!RCIF);           // Polling per Rx buffer full
      return RCREG;
}
```

Listato 6 - Procedura per la ricezione di un singolo byte

3.7 Connessione tra 628Board e PC

Per poter collegare il PIC alla porta seriale si deve collegare il connettore J2 ad un connettore DIN 9 poli femmina. I collegamenti devono essere effettuati come quelli indicati in figura.

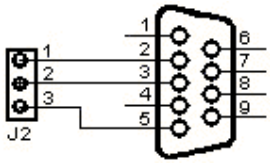

Figura 7 - Schema connesione con connettore DIN 9 Poli

3.8 Configurazione del terminale sul PC

Per prima cosa è necessario configurare il programma Hyper Terminal o simile. Appena lanciate il programma vi viene chiesto di creare una nuova connessione, dategli il nome che preferite e selezionate il pulsante Ok. Nella finestra successiva nell'ultima barra di selezione indicate la porta seriale che utilizzerete (nel mio caso COM2), di nuovo Ok. Ora si apre la schermata per la configurazione della porta COM, impostate i parametri come indicato in Figura 8.

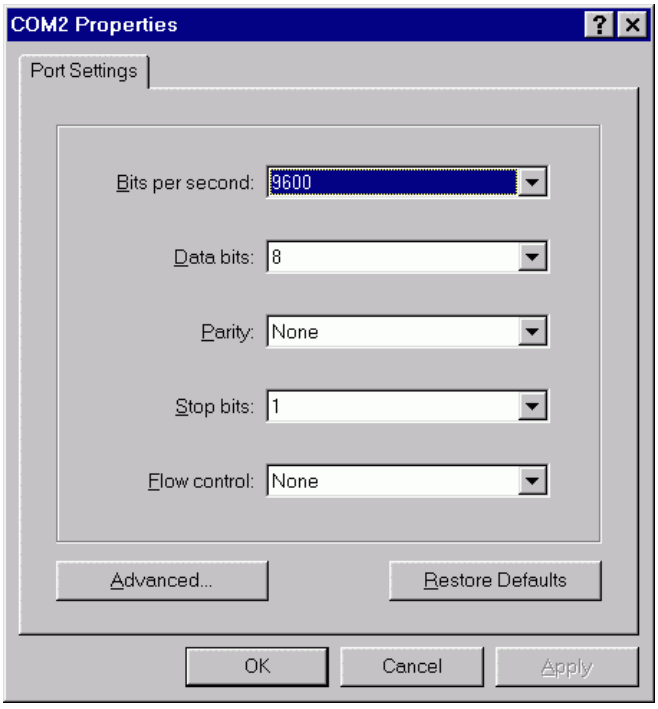

Figura 8 - Impostazioni porta COM del PC

Selezionare ancora una volta il pulsante Ok, ora il computer è in ascolto sulla porta COM da voi indicata.

3.9 Un altro programma di prova

Vediamo ora come utilizzare la trasmissione seriale con un semplice programma che farà interagire la 628Board con il PC.

```
#include <pic.h>
#include "delay.c"
#include "DMLCD2R.h"
#include "UART628.h"

__CONFIG (0x3F61);

void main(void)
{
unsigned char buffer;

CMCON=0x07; // Disabilita i comparatori, per usare la PORTA come I/O
PORTA=0x00; // Azzera la porta A
PORTB=0x04; // Azzera la porta B
TRISA=0x00;
TRISB=0x02; // Setta RB1 come input, il resto come out in particolare RB2

      uart_init();                  // Inizializza l'UART del PIC
      lcd_init();                   // Inizializza il display

      uart_puts("16F628 Ready");    // Invia una stringa al PC
      uart_putch(0x0A);             // Invia un singolo carattere
      uart_putch(0x0D);
      uart_puts("> ");

      lcd_clear();                  // Cancella il display
      lcd_puts("628 Terminal:");    // Scrive un messaggio sul display
      lcd_pos_cur(1,0);             // Si posiziona sulla seconda riga

      while(1)                      // Loop infinito
      {
            buffer=uart_getch();    // Aspetta un carattere dal PC
            lcd_putch(buffer);      // Scrive il carattere sul display
      }
}
```

Listato 7 – Programma di test per l'UART (TESTUART.C)

Come al solito nelle prime righe abbiamo gli include, da notare il nostro UART628.h. Nel main c'è la definizione della variabile buffer, poi la configurazione delle porte del PIC, questa volta abbiamo la porta RB1 che è settata come input per ricevere i pacchetti che ci invierà il PC dalla seriale.
Il programma provvede poi ad inizializzare l'UART del PIC ed il display, tramite le procedure *uart_init()* e *lcd_init()*, di cui abbiamo gia spiegato il funzionamento nei capitoli precedenti.
Il comando *uart_puts("16F628 Ready");* invia la stringa al PC sul quale avevamo già avviato il programma Hyper Terminal. Le due righe successive che inviano i byte 0x0A e 0x0D servono a mandare a capo il cursore sulla finestra del terminale, questi due byte rappresentano i comandi di ritorno a capo e riga successiva. Inseguito viene stampato un ulteriore carattere a mo di prompt per indicare che il PIC è pronto a ricevere i byte dal PC.
Il programma prosegue cancellando il display e scrivendoci sulla prima riga "628 Terminal:", poi posiziona il cursore sulla seconda riga.

La parte finale è ancora un ciclo senza fine, questa volta fatto con un ciclo while, nel quale il PIC aspetta un carattere dal PC, appena lo riceve lo scrive sul display e si rimette in attesa. In Figura 9 si può vedere il programma in esecuzione.

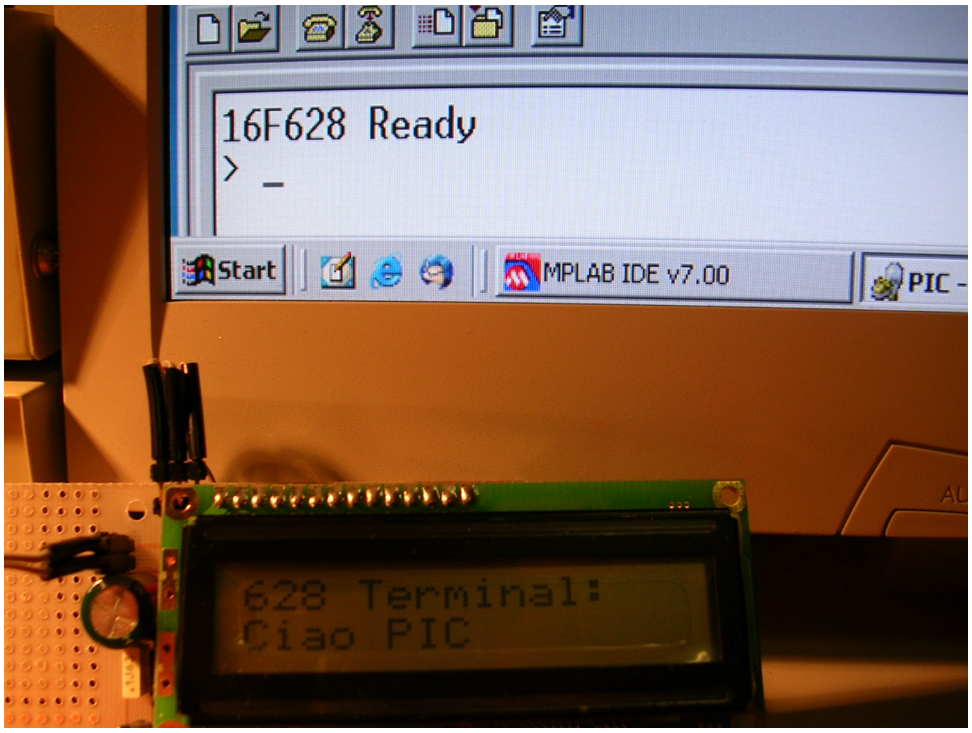

Figura 9 - Esempio di funzionamento del programma

Conclusioni

Vi ringrazio per aver letto il documento fin qui, spero che il progetto vi sia stato utile o quanto meno abbiate apprezzato il lavoro svolto. Conto in un futuro prossimo di realizzare qualche altro progetto che utilizzi questa scheda, in modo da mostrarne altri possibili utilizzi.
Per qualsiasi informazione, commento, richiesta il merito potete contattarmi al seguente indirizzo di posta elettronica iw0gtf@libero.it.
Nel caso in cui realizziate progetti con questa scheda sarei lieto di pubblicarli sul nostro sito, potete contattarmi sempre all'indirizzo indicato in precedenza.

Ringraziamenti

Un grazie particolare va ad Andrea (IW0GTG) che mi ha supportato/sopportato nello sviluppo realizzando diversi prototipi di circuito stampato, ogni volta che gli portavo un nuovo lucido, con qualità e velocità, nonché per aver sopportato i vari QSO in radio in cui gli spiegavo cosa non andava e cercavamo di capire perché.
Grazie anche alla mailing list Roboteck (www.roboteck.org) e tutti i suoi iscritti sempre pronti a dare un aiuto.