di Stefano Barbanti IV3LZQ

Un semplice programmatore per microcontrollori PIC per porta seriale

Programma tutti i PIC esistenti

1ª parte

Introduzione

I microcontrollori PIC della Microchip, dotati di memoria Flash/EEPROM, sono tra i più utilizzati in ambito hobbistico per il loro basso costo, la facile reperibilità e la loro riprogrammabilità (10000 volte...); inoltre l'alto numero di applicazioni già pronte li rendono appetibili a chi intenda provare ad utilizzarli anche in ambito radioamatoriale, ad esempio per realizzare un frequenzimetro, un keyer o il controllo di un VFO DDS. Questo tanto per citare dei progetti apparsi su questa rivista negli ultimi anni e sicuramente interessanti per l'OM autocostruttore.

Avendo praticamente tutti i componenti nel cassetto, e la voglia di provare a costruire un programmatore per la porta seriale pressochè universale, nel senso che programmi quasi tutti i PIC esistenti, ho realizzato questo semplice circuito, dal costo molto contenuto (< 10 euro).

Avviso importante ai lettori

Purtroppo, sebbene una delle idee alla base della realizzazione di questo programmatore fosse proprio di poterlo utilizzare anche con gli adattatori USB-RS232, per i PC portatili e fissi che non dispongono di una vera porta seriale, dopo una settimana di prove, con almeno 5 adattatori diversi e altrettanti software di programmazione ho dovuto gettare la spugna...

Quindi se il vostro PC fisso o portatile non ha una vera porta seriale (ossia integrata già nella scheda madre o su slot PCI per i PC desktop) NON realizzate questo programmatore, perchè al massimo riuscirete a leggere il contenuto del PIC, ma non a scriverlo !

Per ovviare al problema ho in costruzione un altro programmatore, sempre per porta seriale, studiato apposta per funzionare con gli adattatori USB-RS232, che presenterò in un prossimo articolo; ovviamente però tale programmatore ha al suo interno un PIC programmato apposta per risolvere il problema, quindi non è semplice come questo... Tuttavia riutilizzerò gran parte della circuiteria qui presentata e comunque i concetti base sono sempre validi, per cui vi invito comunque alla lettura di questo articolo anche se il vostro moderno PC (hi..) non dispone di una vera porta seriale!

Lo standard RS232 e i PC moderni

Lo standard RS232 (EIA-232) è nato negli anni '70 per stabilire un protocollo di comunicazione tra un generico computer (DTE = Data Terminal Equipment), il PC nel nostro caso, e un generico dispositivo periferico (DCE = Data Communication Equipment), il nostro programmatore. Il protocollo definisce una serie di segnali di interscambio, i livel-

Fig. 1 - Programmatore

Fig. 2 - Adattatori USB RS232

Fig. 3 - Porta seriale vera

Funzione e denominazione RS232	Piedino sul connettore DB9	Segnale di uscita dal PC verso la periferica (il nostro programmatore)	Segnale di ingresso del PC, ricevuto dalla periferica (il nostro programmatore)	Segnale usato per il nostro programmatore
Transmit Data (TXD)	3	PC --> Periferica		Si
Receive Data (RXD)	2		Periferica --> PC	No
Request To Send (RTS)	7	PC --> Periferica		Si
Clear To Send (CTS)	8		Periferica --> PC	Si
DCE Ready (DSR)	6		Periferica --> PC	No
Signal Ground	5			Si
Received Line Signal detector (DCD)	1		Periferica --> PC	No
DTE Ready (DTR)	4	PC --> Periferica		Si
Ring Indicator	9	-	-	No

Tab.1 – Segnali RS232

li di tensione assegnati ai livelli logici "0" e "1" trasmessi, e le procedure di interscambio dati (data handshaking).

Qui prenderemo in considerazione solo i primi due, dato che il programmatore e il software che lo controlla non usano le procedure di interscambio dati del protocollo originale RS232, ma solo il suo mezzo fisico, ossia la porta seriale e i suoi piedini.

I segnali di interscambio dello standard RS232

Come potete osservare dalla Tab.1 i vari piedini della porta seriale, connettore DB9, ossia il connettore a destra nella foto di Fig.3, hanno tutti una denominazione e una funzione definita dallo standard originale.

I segnali in oggetto possono assumere valore logico "0" oppure "1" e quale valore assumono dipende dal livello di tensione generato dal dispositivo di controllo.

I livelli di tensione originari sono i seguenti:

Stato del segnale	Livello di tensione
"0", On, Space, Active	Da +3V a +25V
"1", OFF, Mark, Inactive	Da -3V a -25V

Ovviamente i valori sono riferiti al pin 5, Signal Ground.

Nei PC fissi (desktop) moderni, dotati di porta seriale normalmente i livelli sono limitati a + 12V per lo "0" e a -12V per l'"1", mentre sui portatili (laptop) difficilmente si eccedono i ± 8...10V per i livelli massimi.

I migliori adattatori USB-RS232 sono paragonabili come livelli di tensione a una porta di PC portatile, mentre i più scarsi non vanno oltre i ± 5...8 V.

Inoltre anche la capacità di pilotaggio in corrente dei pin della porta seriale è definita, dove il massimo carico applicabile e pilotabile é un carico di 5kΩ; ovviamente se il carico ha un valore ohmico inferiore il driver "si siede" e il valore di tensione cala.

Avendo chiari i livelli di tensione e il carico massimo pilotabile risulta possibile progettare un qualunque circuito che si debba interfacciare con la porta RS232 nella maniera opportuna.

Inoltre la velocità di trasmissione dati della RS232 va tipicamente da 9,6 kBaud (k bit/s) fino a 187 kBaud.

Come si programma un PIC in modo seriale

Diamo velocemente uno sguardo a come si presentano "esternamente" i microcontrollori PIC, che possono essere in contenitore DIL o SMD a 16-18-20 e 40 pin tipicamente.

Hanno tutti sempre i piedini per l'alimentazione (Vdd) e la massa (Vss), i due piedini per l'oscillatore esterno (OSC1 e OSC2), una serie di piedini che possono fungere da ingresso o da uscita digitale (ma qualcuno anche da ingresso analogico solitamente a seconda del tipo e della dotazione di bordo se con ADC o DAC) (RA0...RB0..) e il pin di Master Clear e Reset (MCLR).

Per la struttura interna e le dotazioni come memoria e quant'altro vi rimando al sito della Microchip, vedi bibliografia, dove potrete scaricare i datasheet (e non solo) per il tipo di PIC a cui siete interessati (es. PIC 16F84A, PIC 16F628A, etc.)

Ricordatevi poi i PIC dotati di memoria programma e dati riscrivibile (flash/EEPROM) da utente elettricamente (ad esempio con il programmatore in oggetto) sono quelli contrassegnati dalla lettera "F" dopo il codice della famiglia, che sono le prime due cifre della sigla, es famiglia = 16, con memoria flash/EEPROM = F, tipo = 628 e versione = A.

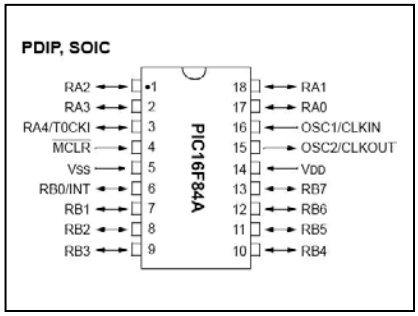
I PIC che non hanno la flash sono solitamente programmabili una volta sola (One Time Programming) e poi non piu' riscrivibili, oppure ci sono quelli cancellabili mediante esposizione a raggi UV tramite l'apposita finestra sul dorso.

Ma torniamo alla procedura di programmazione, che per ciascun PIC é descritta in dettaglio in un apposito datasheet fornito dalla Microchip e rintracciabile dal relativo sito come già detto (Programming Specification).

In generale, per programmare un microcontrollore PIC in modo seriale, ossia un dato dopo l'altro, vengono usati:

- un piedino tra quelli normalmente configurabili come ingresso o uscita digitale, che in modalità di programmazione viene dedicato a fungere da piedino di clock, denominato

Fig. 4 - Pinout 16F84A



Command	Mapping (MSb ... LSb)						Data
Load Configuration	X	X	0	0	0	0	0, data (14), 0
Load Data for Program Memory	X	X	0	0	1	0	0, data (14), 0
Read Data from Program Memory	X	X	0	1	0	0	0, data (14), 0
Increment Address	X	X	0	1	1	0	
Begin Erase Programming Cycle	0	0	1	0	0	0	
Begin Programming Only Cycle	0	1	1	0	0	0	
Load Data for Data Memory	X	X	0	0	1	1	0, data (14), 0
Read Data from Data Memory	X	X	0	1	0	1	0, data (14), 0
Bulk Erase Program Memory	X	X	1	0	0	1	
Bulk Erase Data Memory	X	X	1	0	1	1	

Fig. 5 - Comandi 16F84A

"Clock" (es. per il 16F84A il pin 12, RB6)

- un piedino tra quelli normalmente configurabili come ingresso o uscita digitale, che in modalità di programmazione viene dedicato a fungere da piedino di ingresso/uscita dati da e per il PIC, denominato "Data" (es. per il 16F84A il pin 13, RB7)
- il piedino di Master Clear – Reset a cui viene applicata la tensione di programmazione (compresa tra i 12 e i 14Vdc), (es. per il 16F84A é il pin 4, / MCLR)
- il piedino di normale alimentazione (Vdd) e la massa (Vss) (es. il pin 14 e il pin 5 rispettivamente per il 16F84A)

La sequenza di programmazione consiste prima di tutto nel fare entrare il PIC nella modalità di programmazione, inviargli poi tramite i pin di Clock e Data il comando da eseguire secondo un certo codice (comando a 6 bit, ad es. passa in modalità Scrittura memoria di codice programma) e poi passandogli sempre tramite i due piedini Clock e Data il valore da scrivere; l'indirizzo a cui scrivere parte dalla prima locazione e viene poi successivamente incrementato con l'apposito comando "incrementa indirizzo". Non esiste solo il comando di scrittura del codice programma, ma anche quello di lettura, di cancellazione totale, di scrittura della configurazione HW, e svariati altri, vedasi la Fig.5, con i comandi del PIC 16F84A.

La maggior parte dei PIC richiede solamente che sia presente la tensione di alimentazione normale, 5V tra i piedini Vdd (+) e Vss (zero, massa) prima di iniziare la sequenza di programmazione, mentre alcuni hanno delle modalità di programmazione dove anche la Vdd viene applicata in una determinata sequenza, rispetto agli altri segnali.

Per fare entrare un PIC - ad esempio il 16F84A - nel modo di programmazione, il programmatore, guidato dal programma di controllo residente nel PC, forza a livello logico basso i due piedini Clock (RB6) e Data (RB7) (pin 12 e 13) e il segnale /MCLR (pin 4), portando poi quest'ultimo al valore di Vihh (Vpp), ossia nel range tra 12 e 14V. Il piedino / MCLR viene poi mantenuto al valore di tensione di programmazione Vpp per tutto il resto del ciclo di programmazione/verifica. Segue poi la sequenza di dati e clock sui due piedini appositi, secondo una data temporizzazione, prescritta dalle specifiche di programmazione.

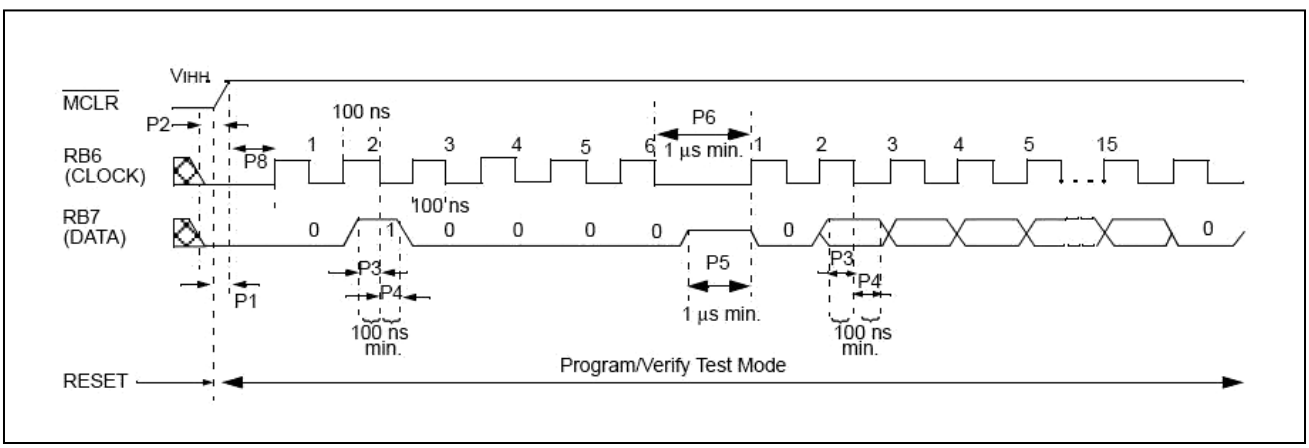
La modalità di lettura (verifica) del PIC funziona in modo similare, solo che il piedino Data viene utilizzato nel modo opposto ossia come uscita digitale comandata dal PIC, che così invia al programmatore il contenuto della locazione di memoria specificata nei bit di indirizzo; da osservare che anche in fase di verifica il pin /MCLR viene tenuto alla tensione di programmazione Vpp (12..14V).

Oltre non vado perchè di più di solito non serve e soprattutto per non farvi venire il mal di testa; comunque fa sempre bene avere un'idea di come funziona il tutto.

Descrizione del circuito del programmatore

Il circuito del programmatore è compatibile con lo standard "JDM", un semplice circuito di programmazione così chiamato dalle iniziali del suo creatore, vi-

Fig. 6 - Ciclo programmazione 16F84A



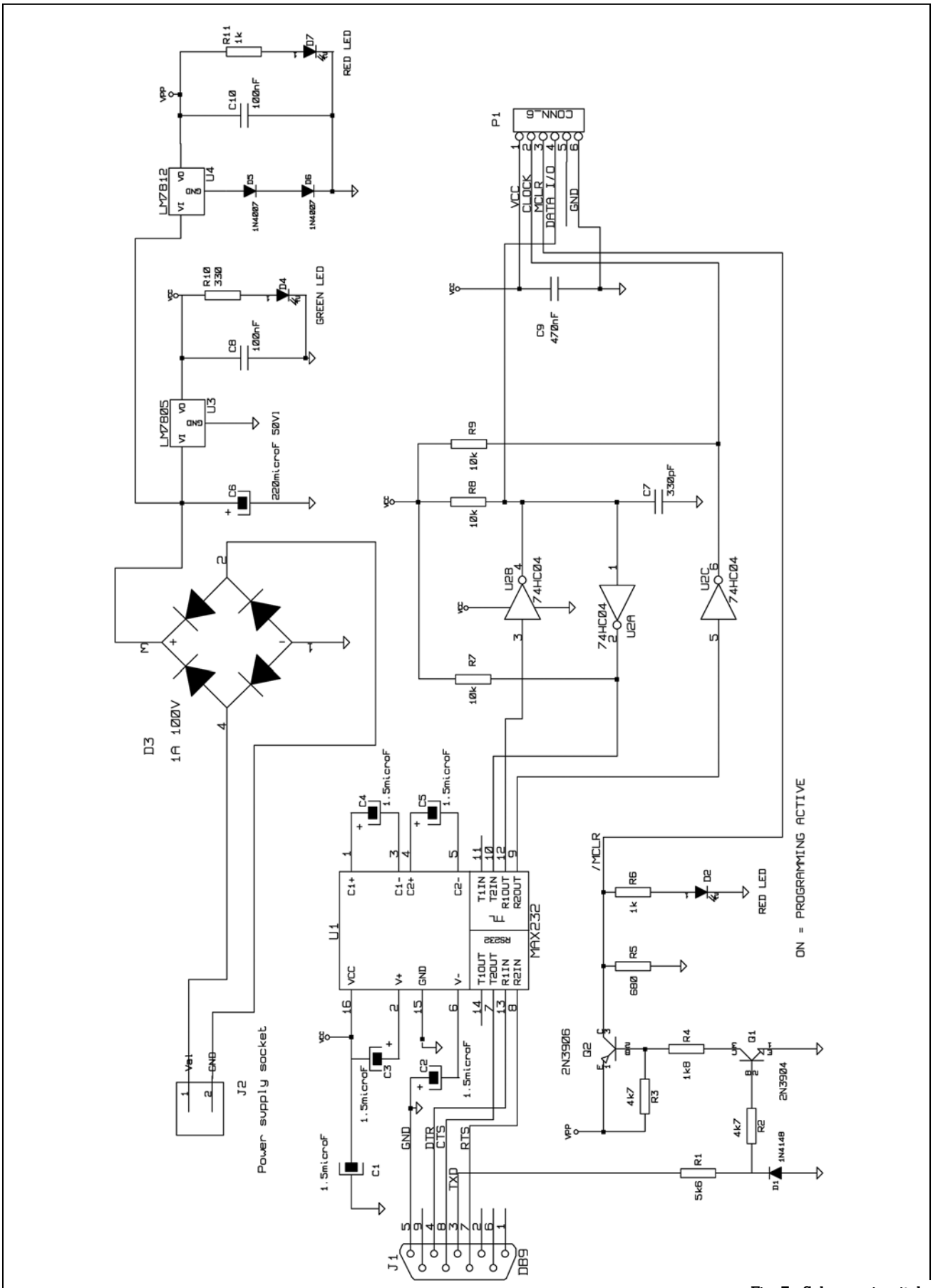


Fig. 7 - Schema circuitale

Funzione e denominazione RS232	Piedino sul connettore DB9	Funzione Segnale sul programmatore
Transmit Data (TXD)	3	PC --> Programmatore **Segnale /MCLR**
Request To Send (RTS)	7	PC --> Programmatore **Segnale CLOCK**
Clear To Send (CTS)	8	Programmatore -->PC **Segnale DATA_OUT** (dato letto dal PIC)
Signal Ground	5	Ground
DTE Ready (DTR)	4	PC --> Programmatore **Segnale DATA_IN** (dato da scrivere sul PIC)

Tab. 2

sto che tutti i software di programmazione per PC supportano tale standard; vi rimando al link in bibliografia per maggiori informazioni al riguardo.

Tale programmatore (standard JDM) utilizza i segnali della porta seriale come da tabella 2.

Il protocollo usato é una sorta di I2C con handshaking, tanto per dare un'idea a chi è familiare con tale protocollo.

In rete si trovano molti schemi derivati dal JDM originale, tuttavia tali schemi sono "passivi" nel senso che creano la tensione di alimentazione del PIC (+ 5 V, Vdd o Vcc) e quella di programmazione (Vpp, tra +12 V e 14 V) direttamente dalla porta seriale stessa. Questa tecnica però non è compatibile con il più delle porte seriali dei PC portatili in quanto tende a caricare troppo la porta stessa.

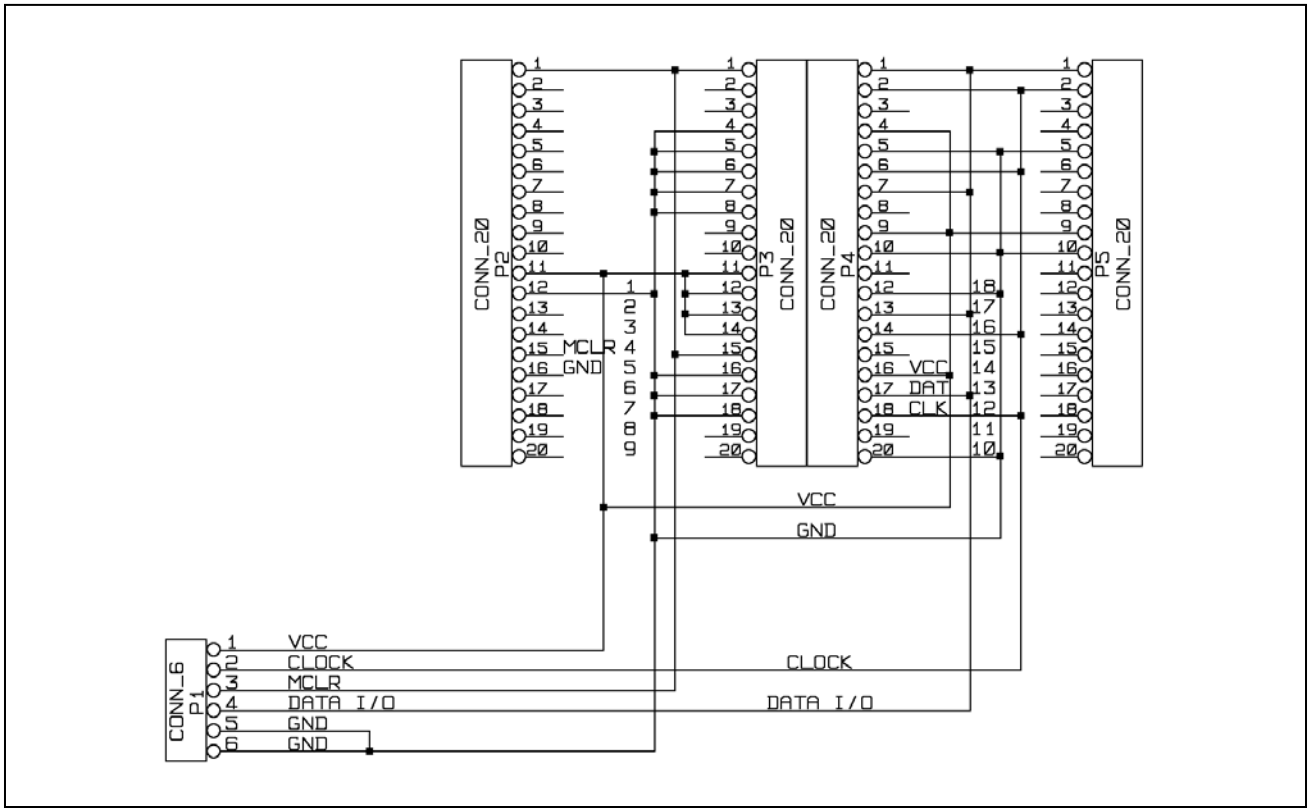
Inoltre i segnali della porta seriale non sono compatibili con i livelli TTL dei piedini di ingressi/uscite del microcontrollore PIC da programmare, per cui serve un traslatore di livello, in entrambe le direzioni, che carichi la porta seriale entro i limiti prescritti (5 kΩ al massimo).

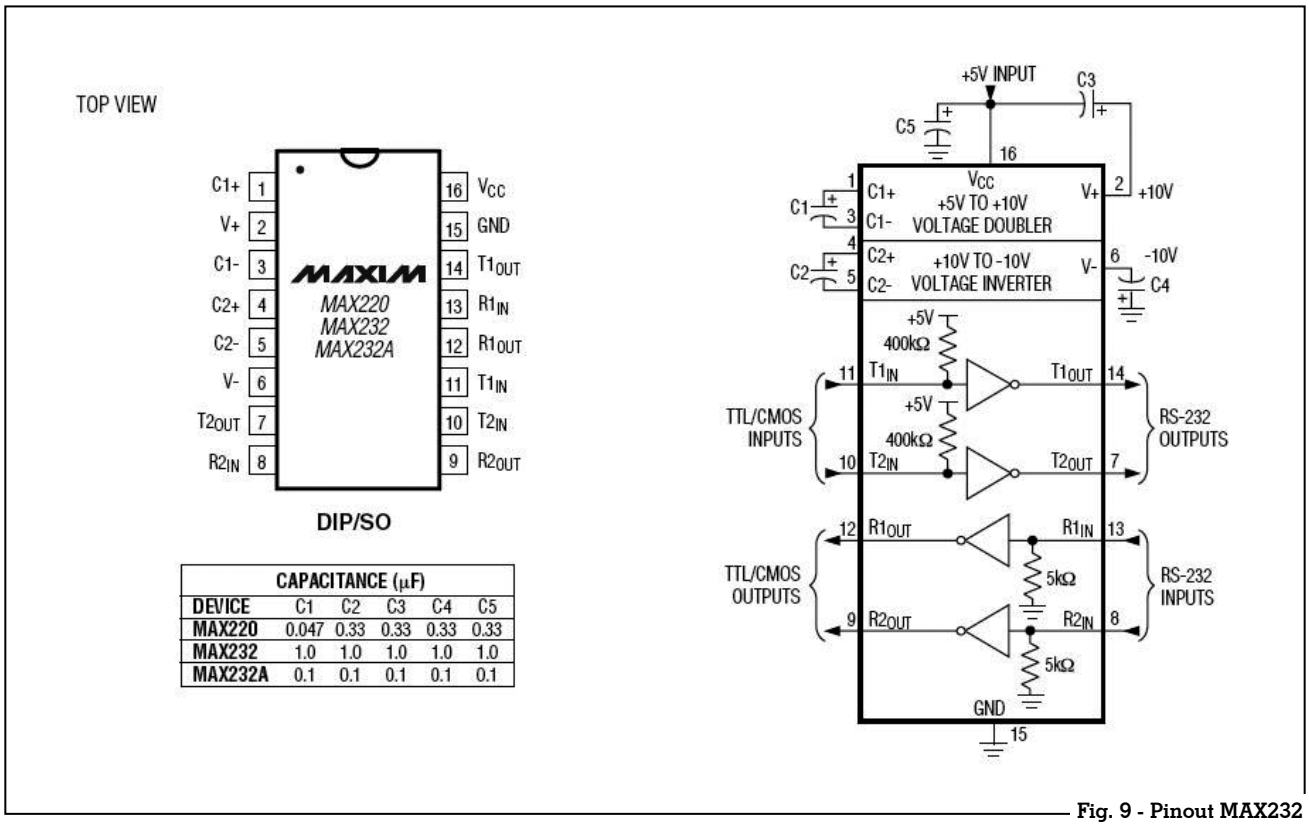
Tale funzione é svolta dal circuito MAX232 (o equivalenti) della MAXIM, denominato U1 sullo schema visibile in Fig.7, nel mio caso riciclato da una scheda costruita a suo tempo per un ICOM...

Tale integrato, vedi datasheet dal sito della MAXIM, contiene due traslatori di livello - invertitori, in entrambe le direzioni, da/per RS232 <--> TTL, come visibile in Fig.8, e delle apposite circuiterie interne per generare dall'alimentazione a +5V le due tensioni di +10V e -10V necessarie per i livelli logici compatibili con lo standard RS232.

Poichè abbiamo tre segnali dal PC verso il programmatore e uno in direzione opposta, in teoria sarebbero stati necessari due MAX232, ma visto che il segnale /MCLR - TXD pin 3 DB9 della RS232 - é sostanzialmente un segnale a livello, tenuto attivo per tutto il ciclo di programmazione/verifica, e che deve comunque avere una circuiteria apposita per applicare la tensione di programmazione al PIC (13V nominali), il tutto si è risolto con due

Fig.8 - Schema zoccolo universale)





CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	0.047	0.33	0.33	0.33	0.33
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Fig. 9 - Pinout MAX232

transistors, Q1 e Q2, e qualche resistenza.

Quando il segnale di TXD sul pin 2 del connettore DB9 della seriale é a livello logico "0" della RS232, ossia tra i +3V e i +8V...12V, Q1 é in saturazione - il diodo D1 é interdetto – e anche Q2 é in saturazione, quindi i +13V appaiono sulla linea / MCLR. Viceversa, se la tensione sul pin 2 del DB9 é negativa, il diodo D1 conduce, proteggendo la giunzione base-emettitore di Q1, che rimane interdetto, così come Q2.

Quindi gli altri due segnali dal PC verso il programmatore, la linea di Clock e di Data, sono assegnati ai due traslatori di livello – invertitori, R1 e R2 del MAX232 – U1. Questi segnali, essendo invertiti logicamente dal MAX232 devono essere di nuovo invertiti prima di essere applicati al PIC. A tale scopo provvedono le porte logiche NOT del 74HC04, anche questo scovato nella cassettiera.

Il segnale sul pin Data del PIC é bidirezionale, quindi serve un NOT e un traslatore di livello-invertitore del MAX232 da TTL a RS232, per poter pilotare il segnale CTS dal programmatore verso il PC.

Il resto del circuito é poi dedicato a generare le due tensioni di alimentazione, ossia i +5V e il +13V della tensione di programmazione, con due regolatori di tensione monolitici, che consiglio di impiegare in contenitore TO-220, in modo particolare il 7805, visto che in ingresso servono almeno 16...17V per garantire 13V stabili.

Il ponte a diodi si può eliminare se non si prevede di alimentare in alternata il tutto e se non si dispongono di almeno 18...20V in ingresso.

(Continua)