# PIC
# PROGRAMMAZIONE IN LINGUAGGIO MACCHINA PER PRINCIPIANTI

By Claudio Fin

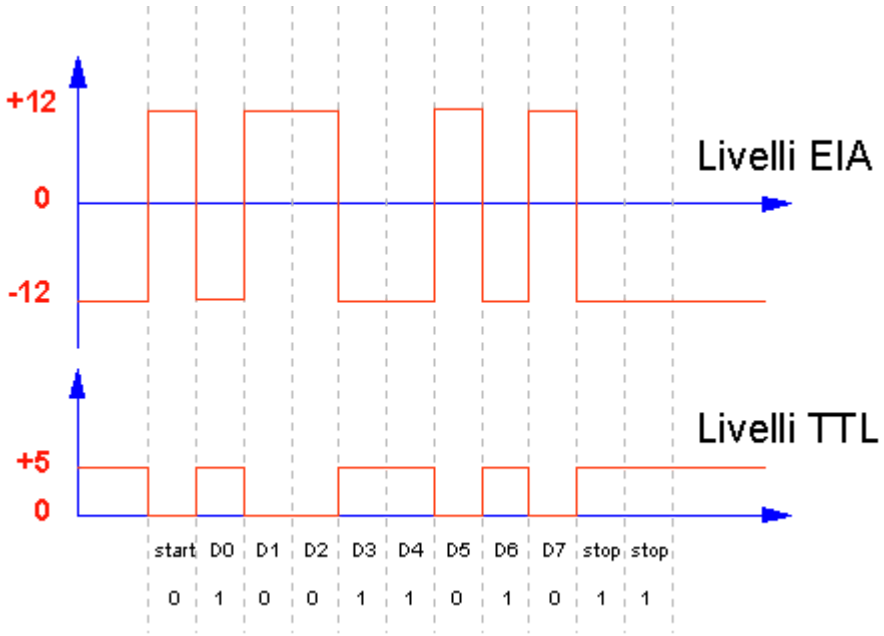
[Precedente] [Indice principale]

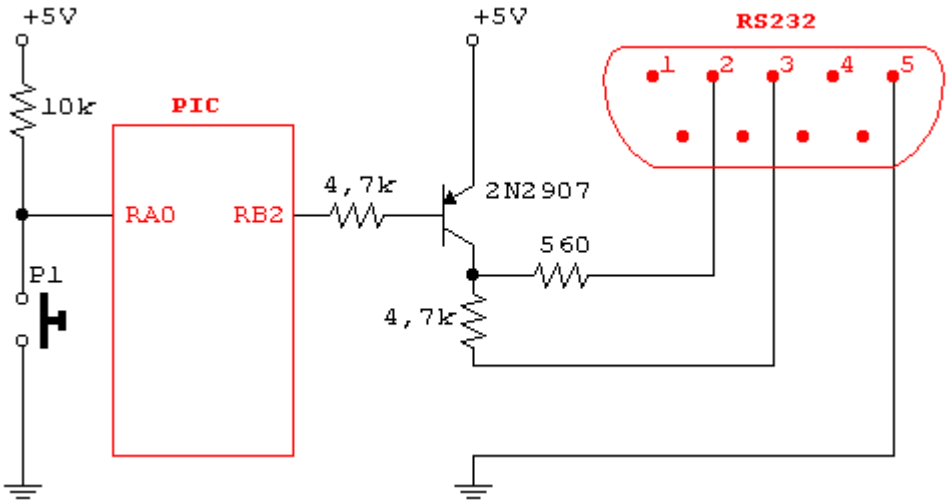
## TRASMISSIONE SERIALE

In generale la trasmissione seriale dei dati consiste nell'inviare in sequenza su un unico filo i diversi bit che compongono questi dati. Ci sono diversi protocolli seriali (sincroni, asincroni, orientati al byte o al bit), qui si parla del classico protocollo asincrono start-stop a bassa velocità utilizzato dalla porta RS232 dei PC.

Il protocollo prevede che per ogni byte da trasmettere venga dapprima inviato un bit a 0 (bit di start), seguito dagli 8 bit del dato (partendo dal bit meno significativo e finendo con quello più significativo), per finire con un bit fisso a 1 (bit di stop). Terminata la trasmissione la linea rimane fissa a 1, che è il «livello di riposo» quando non passano dati.

Inoltre va ricordato che i livelli di tensione sulla porta RS232 (livelli EIA) sono diversi da quelli presenti sui pin del PIC (livelli TTL), pertanto per collegarli occorre sempre interporre un'apposita interfaccia (generalmente composta da un integrato MAX232 con 4 condensatori).






In questo esperimento si vuole inviare il carattere «A» maiuscola dal pin RB2 ogni volta che viene premuto il pulsante P1. Sul PC dovrà essere attivo un programma di emulazione terminale, ad esempio l'hyper terminal di Windows settato a 9600 8-N-1. In questo caso, in cui vogliamo solo provare inviare dei dati dal PIC al PC, ci accontentiamo di un'interfaccia TTL/EIA elementare con un solo transistor.

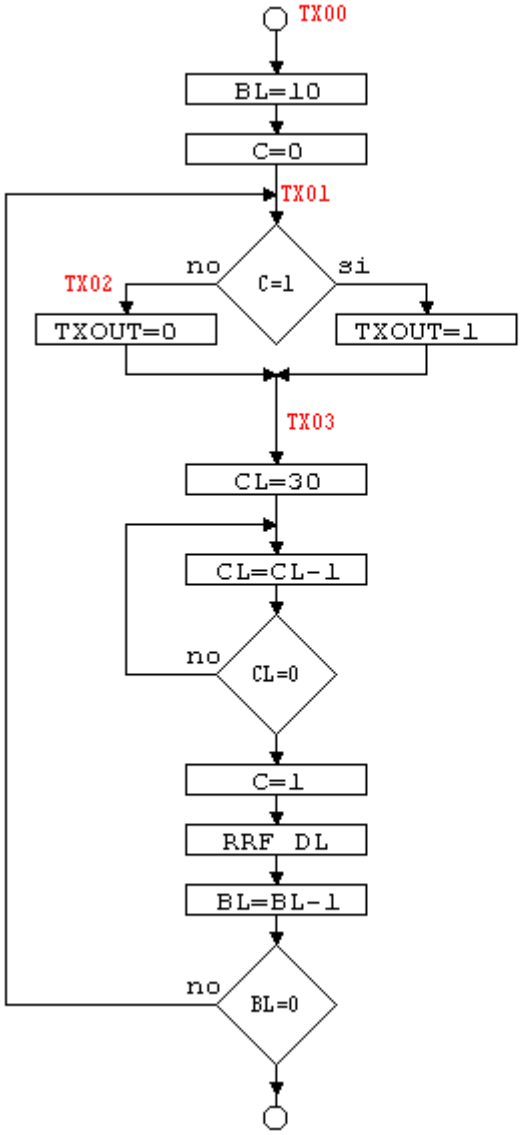
Ragionando in modo modulare e strutturato si può già «imbastire» il ciclo di controllo principale del programma, in fondo non è molto diverso dal conteggio up/down a pulsanti. Si deve attendere la pressione del pulsante, verificare che dopo un certo tempo sia ancora premuto (per evitare i rimbalzi), eseguire le istruzioni necessarie (per ora lasciate in sospeso e indicate con dei puntini), attendere che la pressione del pulsante termini (nel caso fosse ancora premuto), verificare di nuovo che dopo un certo tempo sia ancora rilasciato e tornare ad attendere la pressione.

```
            PROCESSOR   16F628
            RADIX       DEC
            INCLUDE     "P16F628.INC"
            __CONFIG    11110100010000B
;-----------------------------------------------------
            ORG         32          ;Inizio area RAM
H_CONT      RES         1           ;contatori per
L_CONT      RES         1           ;subroutine di ritardo
#DEFINE     PULS        PORTA,0     ;Pulsante (a riposo=1)
;-----------------------------------------------------
            ORG         0
            MOVLW       7
            MOVWF       CMCON       ;PORTA=I/O digitali

MAINLOOP    BTFSC       PULS        ;Attende pressione pulsante
            GOTO        MAINLOOP
            CALL        DELAY       ;Chiama ritardo antirimbalzo
            BTFSC       PULS        ;Se ancora premuto skip
            GOTO        MAINLOOP    ;altrimenti torna a MAINLOOP

;           ..........

ATTRIL      BTFSS       PULS        ;Attende rilascio pulsante
            GOTO        ATTRIL
            CALL        DELAY       ;Chiama ritardo antirimbalzo
            BTFSS       PULS        ;Se ancora rilasciato skip
            GOTO        ATTRIL      ;Altrimenti torna ad ATTRIL
            GOTO        MAINLOOP    ;Torna all'inizio
;-----------------------------------------------------
DELAY       MOVLW       65          ;ritardo 50,053 ms
```

```
            MOVWF       H_CONT
            CLRF        L_CONT
            DECFSZ      L_CONT,F
            GOTO        $-1
            DECFSZ      H_CONT,F
            GOTO        $-3
            RETURN
;-----------------------------------------------------
            END
```

Il programma compie già questo lavoro, ma naturalmente così com'è non produce ancora alcun risultato visibile. Resta solo da sostituire ai puntini un «qualche cosa» che serva ad inviare i giusti segnali dal pin RB2. Questo qualche cosa naturalmente sarà un'apposita subroutine «driver», a cui per esempio possiamo passare il valore da trasmettere in un registro. E' buona norma passare alle subroutine i valori (o i comandi / messaggi / informazioni) attraverso dei registri diversi da W, in quanto W è l'unico registro hardware del micro ed è usato in quasi tutte le operazioni (benchè in alcuni casi semplici possa essere usato per questa operazione), pertanto decidiamo di passare il valore in un registro di nome DL. Una subroutine adatta per questo scopo è la seguente:

```
;-----------------------------------------------------
; Trasmissione seriale di un byte in formato
; 9600 8-N-1 dal pin "TXOUT" (per clock 4MHz)
; Input: DL=valore da trasmettere.
; Registri usati: BL=contatore dei bit, CL=contatore cicli ritardo
;-----------------------------------------------------
TX00        MOVLW       10
            MOVWF       BL          ;Contatore dei bit=10
            BCF         STATUS,C    ;Azzera flag C
TX01        BTFSS       STATUS,C    ;Se flag C=1 skip
            GOTO        TX02        ;altrimenti salta a TX02
            NOP
            BSF         TXOUT       ;Manda a 1 il pin TXOUT
            GOTO        TX03        ;e salta a TX03
TX02        BCF         TXOUT       ;Manda a 0 il pin TXOUT
            NOP
            NOP
TX03        MOVLW       30
            MOVWF       CL          ;CL=30, ritardo durata bit
            DECFSZ      CL,F        ;Decrementa CL, skip se zero
            GOTO        $-1         ;Altrimenti nuovo decremento
            BSF         STATUS,C    ;Setta flag C
            RRF         DL,F        ;Ruota a destra DL
            DECFSZ      BL,F        ;Decrem.contat.bit, skip se 0
            GOTO        TX01        ;Altrimenti torna a TX01
            RETURN
```




```
MOVLW   10
MOVWF   BL
BCF     STATUS,C

BTFSS   STATUS,C
GOTO    TX02
NOP
BSF     TXOUT
GOTO    TX03
BCF     TXOUT
NOP
NOP


MOVLW   30
MOVWF   CL
DECFSZ  CL,F
GOTO    $-1



BSF     STATUS,C

RRF     DL,F


DECFSZ  BL,F
GOTO    TX01


RETURN
```

La subroutine è ottimizzata per produrre una sequenza di bit lunghi esattamente 104µS come previsto dalla velocità di 9600 bit al secondo, in realtà dovrebbero essere 104,166µS, ma il programma a 4MHz non può generare segnali a frazioni più piccole di 1 microsecondo (questa differenza è comunque del tutto trascurabile). La trasmissione completa di un singolo byte avviene perciò esattamente in 1,04 millisecondi.

Ora dobbiamo vedere cosa vuol dire «trasmettere un carattere», visto che in realtà un byte può contenere solo un numero compreso tra 0 e 255. La soluzione è semplice, un carattere è semplicemente un simbolo a cui è stato assegnato convenzionalmente un valore numerico rappresentabile con un byte. Questa convenzione è il set dei caratteri ASCII , che stabilisce ad esempio che la nostra «A» maiuscola vada rappresentata con il valore 65. Questo significa che se inviamo il valore 65 verso un terminale (o emulatore di terminale), questo farà apparire a video una A (se invece volessimo far apparire il numero 65 dovremmo inviare il codice ASCII del simbolo «6» seguito dal codice del «5»).

L'assembly dei PIC permette di specificare un valore in diversi modi

```
In decimale                     65
In esadecimale                  0x41, 041H
```

```
In binario                              01000001B
Specificando direttamente il carattere 'A'
```

Ecco quindi il programma completo, a cui sono state naturalmente aggiunte le definizioni dei registri usati dalla nuova subroutine e il settaggio iniziale della porta B.

```
            PROCESSOR   16F628
            RADIX       DEC
            INCLUDE     "P16F628.INC"
            __CONFIG    11110100010000B
;-----------------------------------------------------
            ORG         32          ;Inizio area RAM
BL          RES         1
CL          RES         1
DL          RES         1
H_CONT      RES         1
L_CONT      RES         1
#DEFINE     PULS        PORTA,0     ;Pulsante (a riposo=1)
#DEFINE     TXOUT       PORTB,2     ;Uscita seriale
;-----------------------------------------------------
            ORG         0
            BSF         TXOUT       ;Prescrive latch di uscita
            BSF         STATUS,RP0  ;Attiva banco 1
            BCF         TRISB,2     ;PORTB=uscita
            BCF         STATUS,RP0  ;Ritorna al banco 0
            MOVLW       7
            MOVWF       CMCON       ;PORTA=I/O digitali

MAINLOOP    BTFSC       PULS        ;Attende pressione pulsante
            GOTO        MAINLOOP
            CALL        DELAY       ;Chiama ritardo antirimbalzo
            BTFSC       PULS        ;Se ancora premuto skip
            GOTO        MAINLOOP    ;altrimenti torna a MAINLOOP

            MOVLW       'A'         ;W=codice della "A"
            MOVWF       DL          ;lo mette in DL
            CALL        TX00        ;lo trasmette

ATTRIL      BTFSS       PULS        ;Attende rilascio pulsante
            GOTO        ATTRIL
            CALL        DELAY       ;Chiama ritardo antirimbalzo
            BTFSS       PULS        ;Se ancora rilasciato skip
            GOTO        ATTRIL      ;Altrimenti torna ad ATTRIL
            GOTO        MAINLOOP    ;Torna all'inizio
;-----------------------------------------------------
DELAY       MOVLW       65          ;ritardo 50,053 ms
            MOVWF       H_CONT
            CLRF        L_CONT
            DECFSZ      L_CONT,F
            GOTO        $-1
            DECFSZ      H_CONT,F
            GOTO        $-3
            RETURN
;-----------------------------------------------------
TX00        MOVLW       10
            MOVWF       BL          ;Contatore dei bit=10
            BCF         STATUS,C    ;Azzera flag C
TX01        BTFSS       STATUS,C    ;Se flag C=1 skip
            GOTO        TX02        ;altrimenti salta a TX02
            NOP
            BSF         TXOUT       ;Manda a 1 il pin TXOUT
            GOTO        TX03        ;e salta a TX03
TX02        BCF         TXOUT       ;Manda a 0 il pin TXOUT
            NOP
            NOP
TX03        MOVLW       30
```

```
            MOVWF       CL          ;CL=30, ritardo durata bit
            DECFSZ      CL,F        ;Decrementa CL, skip se zero
            GOTO        $-1         ;Altrimenti nuovo decremento
            BSF         STATUS,C    ;Setta flag C
            RRF         DL,F        ;Ruota a destra DL
            DECFSZ      BL,F        ;Decrem.contat.bit, skip se 0
            GOTO        TX01        ;Altrimenti torna a TX01
            RETURN
;-------------------------------------------------
            END
```

## TRASMISSIONE SERIALE HARDWARE

Il sistema di trasmissione visto finora è usabile su tutti i pic della serie 12F/16F, in quanto il segnale seriale viene creato via software «modulando» con i giusti tempi la tensione presente sul pin RB2. Il PIC 16F628 dispone però anche di una periferica interna (la USART) in grado di svolgere questa trasmissione completamente in automatico. Questa periferica in uscita è collegata al pin RB2 (per questo motivo è stato scelto negli esempi precedenti), pertanto il circuito rimane identico, inoltre non serve settare esplicitamente come uscita il pin RB2 in quanto lo diventa automaticamente.

Come si può notare la parte principale del programma non subisce alcun cambiamento, all'inizio ci sono le nuove istruzioni per il settaggio della USART, e la subroutine di trasmissione si semplifica enormemente.

```
            PROCESSOR   16F628
            RADIX       DEC
            INCLUDE     "P16F628.INC"
            __CONFIG    11110100010000B
;-------------------------------------------------
            ORG         32          ;Inizio area RAM
H_CONT      RES         1
L_CONT      RES         1
#DEFINE     PULS        PORTA,0     ;Pulsante (a riposo=1)
;-------------------------------------------------
            ORG         0
            BSF         STATUS,RP0  ;banco 1
            MOVLW       25
            MOVWF       SPBRG
            BSF         TXSTA,BRGH  ;9600 BAUD
            BSF         TXSTA,TXEN  ;ABILITA TX
            BCF         STATUS,RP0  ;banco 0
            BSF         RCSTA,SPEN  ;ABILITA SERIALE
            MOVLW       7
            MOVWF       CMCON       ;PORTA=I/O digitali

MAINLOOP    BTFSC       PULS        ;Attende pressione pulsante
            GOTO        MAINLOOP
            CALL        DELAY       ;Chiama ritardo antirimbalzo
            BTFSC       PULS        ;Se ancora premuto skip
            GOTO        MAINLOOP    ;altrimenti torna a MAINLOOP

            MOVLW       'A'         ;codice della "A"
            MOVWF       DL
            CALL        TX00        ;trasmette

ATTRIL      BTFSS       PULS        ;Attende rilascio pulsante
            GOTO        ATTRIL
            CALL        DELAY       ;Chiama ritardo antirimbalzo
```

```
            BTFSS       PULS        ;Se ancora rilasciato skip
            GOTO        ATTRIL      ;Altrimenti torna ad ATTRIL
            GOTO        MAINLOOP    ;Torna all'inizio
;-----------------------------------------------------
DELAY       MOVLW       65          ;ritardo 50,053 ms
            MOVWF       H_CONT
            CLRF        L_CONT
            DECFSZ      L_CONT,F
            GOTO        $-1
            DECFSZ      H_CONT,F
            GOTO        $-3
            RETURN
;-----------------------------------------------------
TX00        BTFSS       PIR1,TXIF   ;Attende trasmettitore libero
            GOTO        $-1
            MOVF        DL,W
            MOVWF       TXREG       ;Invia dato
            RETURN
;-----------------------------------------------------
            END
```

[Segue]

---

Pagina creata nell'agosto 2005 - Ultimo aggiornamento 17-8-2005

---