Lezione 4

Uno degli scogli maggiori per chi inizia a lavorare con i microcontroller, è l'interfacciamento con tastiere a matrice. La cosa potrebbe a prima vista sembrare complessa, ma in realtà è implementabile con poche righe di software. Se la tastiera fosse composta da n pulsanti con un capo in comune, o connesso al positivo oppure a massa, sapreste già come risolvere il problema, perchè si tratterebbe soltanto di moltiplicare per n le routine di acquisizione da pulsante viste nella precedente puntata.
Con una tastiera a matrice, invece, non possiamo applicare le stesse routine, poichè avremmo conflittualità su pulsanti di una stessa riga oppure di una stessa colonna.
Cerchiamo allora di capire come è formata una tastiera a matrice, aiutandoci con lo schema di figura 1.

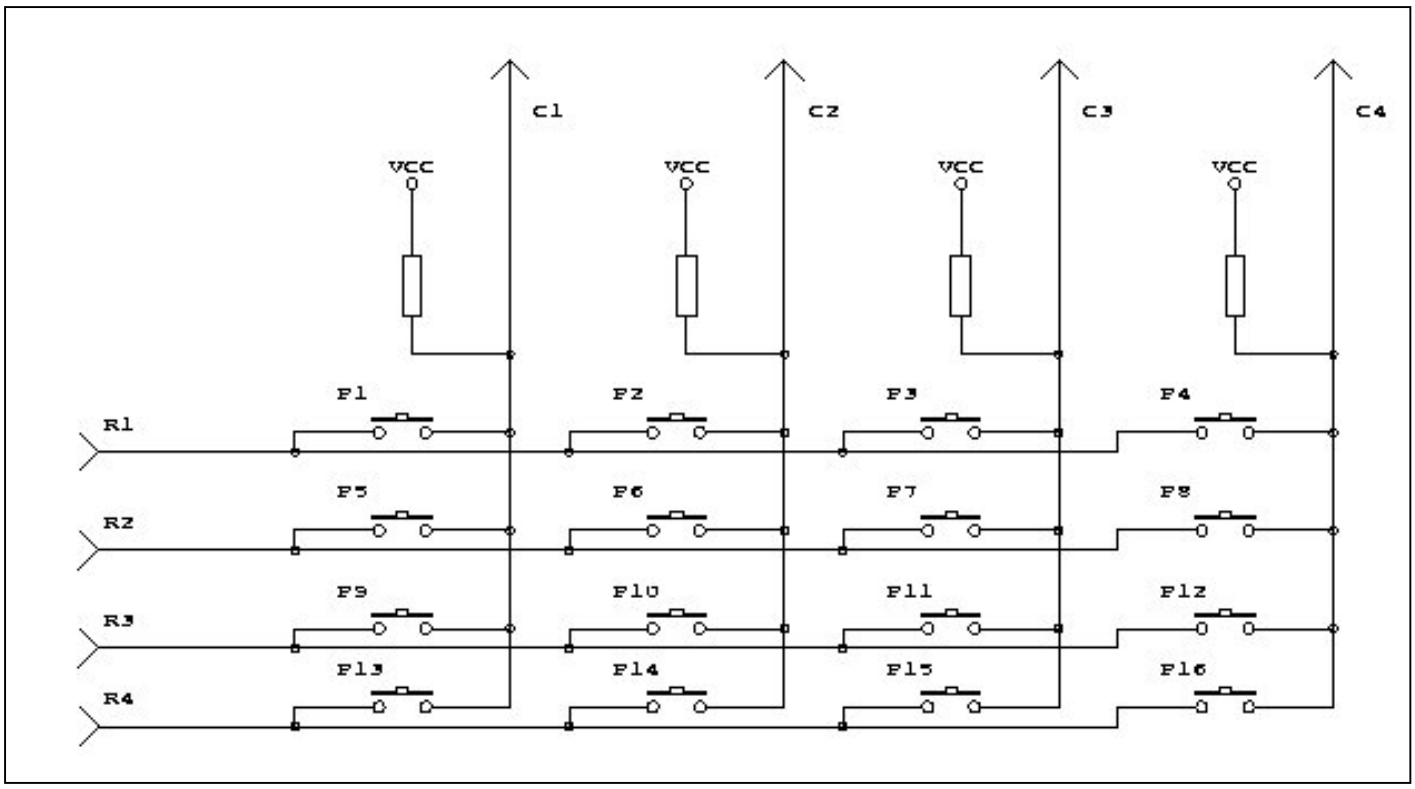

Figura 1. Schema di una tastiera a matrice di 4x4 tasti

Vediamo che per implementare una tastiera a matrice con 16 pulsanti, li dobbiamo connettere in modo tale da ottenere otto collegamenti: quattro per le righe della matrice e quattro per le colonne. Nello schema vediamo anche resistenze di pull-up, ma queste sulle tastiere vere e proprie non ci sono. In pratica, tali resistenze si trovano poi sul circuito elettrico, perchè con esse è possibile realizzare un piccolo algoritmo in grado di decifrare se un tasto viene premuto e quale tasto sia.

Il metodo di scansione

Per ottenere il numero del tasto premuto, ci avvaliamo di software che dobbiamo implementare nel microcontroller e che, ripetutamente va a testare le otto linee della tastiera in un certo modo. Diciamo subito che, per ottenere ciò, le otto linee devono essere divise in due gruppi da quattro: nel nostro esempio le colonne saranno viste dal controller come ingressi (ecco il perchè delle resistenze di pull-up) mentre le righe saranno predisposte come uscite.
In figura 2 troviamo il diagramma temporale di come, generalmente, vengono abilitate le uscite relative alle righe della matrice: quattro abilitazioni formano un ciclo detto "ciclo di scansione".

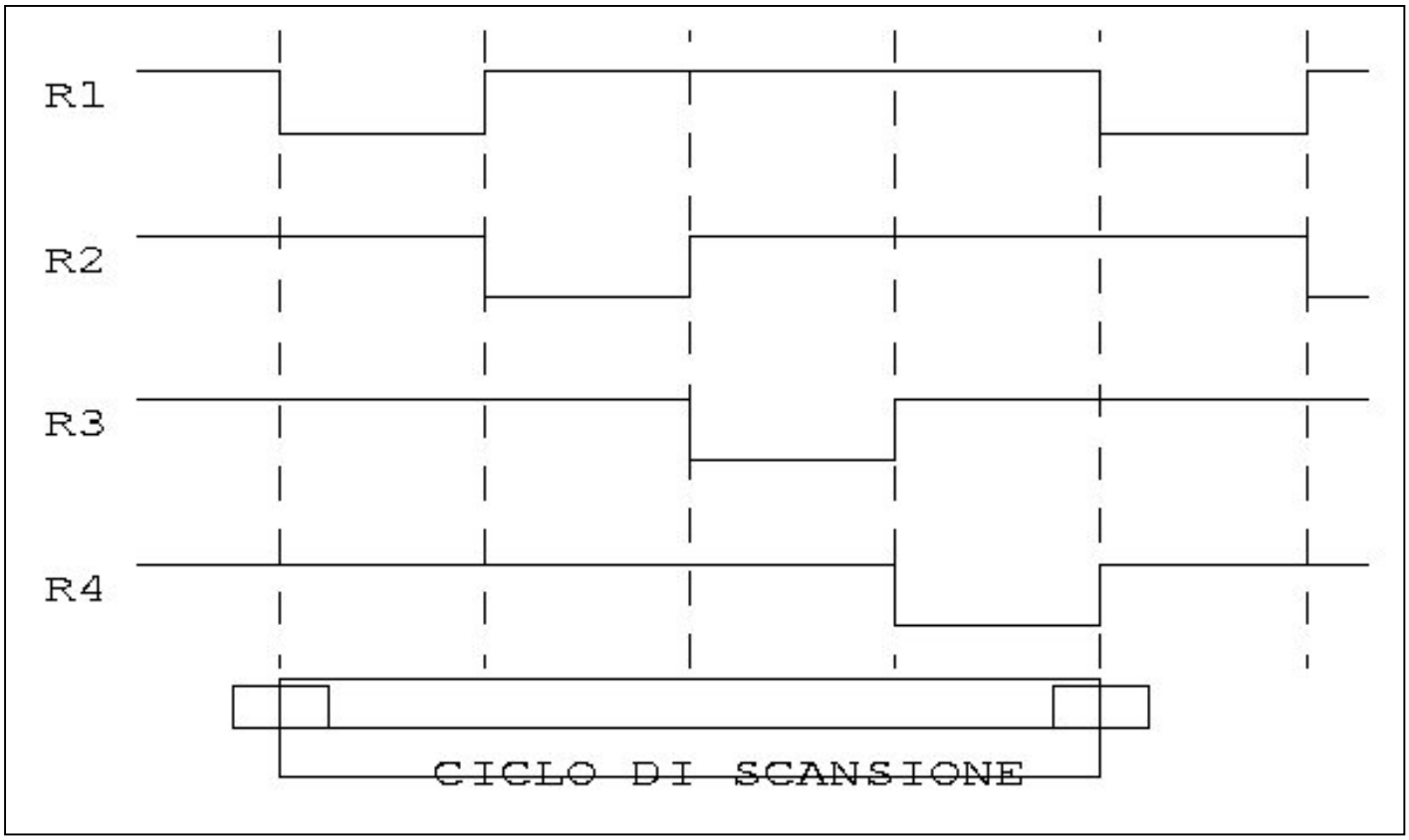

Figura 2. Diagramma temporale del ciclo di scansione

Alla partenza, tutte le uscite del controller sono ad alto livello, quindi sui quattro ingressi troveremo quattro livelli logici alti. Il ciclo inizia con il portare a basso livello l'uscita relativa alla riga 1: allora, se andassimo a leggere ora gli ingressi, avremmo uno "0" nel caso del pulsante premuto, un "1" se tutto rimane come prima. Ad ogni ingresso colonna, corrisponderà il relativi pulsante abilitato dalla riga 1. Eseguita la scansione della riga 1, tale linea viene riportata ad alto livello e si passa alla riga 2. Il procedimento è identico al precedente, solo che questa volta, saranno abilitati soltanto i pulsanti relativi a quella riga. Il ciclo termina quando anche la quarta riga è stata abilitata.

I tempi di scansione possono essere anche molto rapidi, ovviamente anche in funzione della distanza della tastiera dal micro e dal valore delle resistenze di pull-up. Lo stesso meccanismo avrebbe funzionato anche invertendo le righe con le colonne, oppure inserendo delle resistenze di pull-down e abilitando i pulsanti con segnali adalto livello.
Questo sistema ci consente, come vedremo avanti, di sfruttare sia gli ingressi che le uscite in comune con altre periferiche, che potranno essere abilitate separatamente con pin di selezione.
Il circuito di verifica
In figura 3 troviamo lo schema elettrico del circuito che dovremo realizzare per verificare il funzionamento del programma PROG07.ASM.

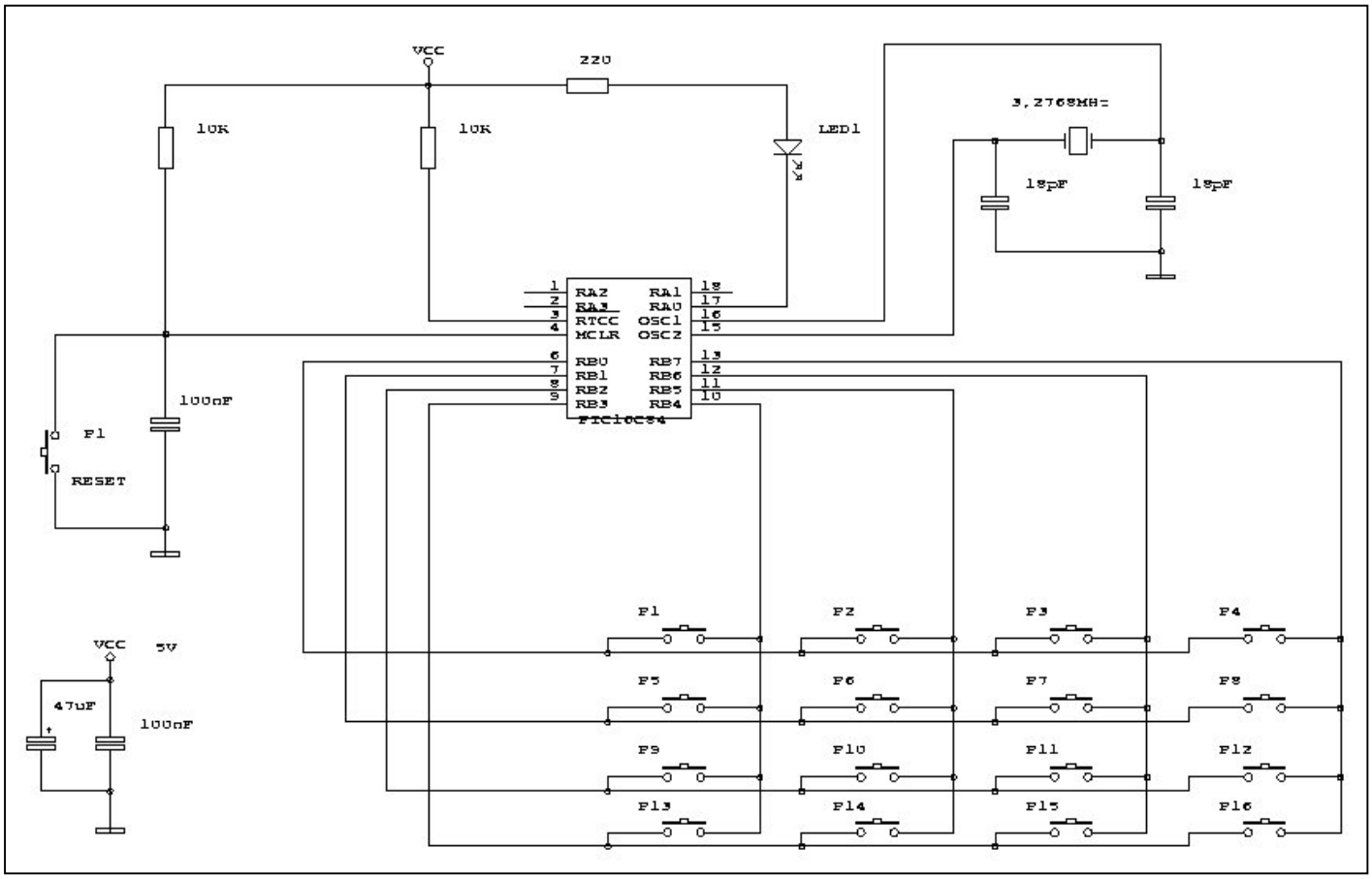

Figura 3. Schema elettrico del circuito di prova

Con questo programma, vedremo come implementare la subroutine di scansione della tastiera.

```
 TITLE 'PROG07: Scansione tastiera'
 list F=INHX8M,P=16C84
;-----------------------------------------------------------------
RTCC   EQU 01H                 ; Real Time Clock Counter
STAT   EQU 03H                 ; Registro di stato
PORTA  EQU 05H                 ; Porta A
PORTB  EQU 06H                 ; Porta B
INTCON EQU 0BH                 ; Registro abilitazione interrupt
TR_A   EQU 85H                 ; Tris A
TR_B   EQU 86H                 ; Tris B
OPTIO  EQU 81H                 ; Registro OPTION
;-----------------------------------------------------------------
FR01   EQU 0CH                 ; Ritardo 20mS
FR02   EQU 0DH                 ; Ritardo 20mS
FR03   EQU 0EH                 ; Buffer pulsante premuto
;-----------------------------------------------------------------
#define LED      PORTA,0       ; Led sulla porta A pin 0
#define C1       PORTB,4       ; Colonna 1
#define C2       PORTB,5       ; Colonna 2
#define C3       PORTB,6       ; Colonna 3
#define C4       PORTB,7       ; Colonna 4
;-----------------------------------------------------------------
       ORG       0
       goto      START         ;Reset vector
       nop
       nop
       nop
       goto      INTERP        ;Interrupt vector
;-------------------- subroutine scansione tastiera -------------
;Torna 0 se nessun tasto premuto, altrimenti il valore del tasto
;-----------------------------------------------------------------
TASTI  movlw     b'11111110'   ; Abilito scansione riga 1
       movwf     PORTB         ;
       clrwdt                  ;
       btfss     C1            ; Test pulsante P1
       retlw     .1            ;
       btfss     C2            ; Test pulsante P2
       retlw     .2            ;
       btfss     C3            ; Test pulsante P3
       retlw     .3            ;
       btfss     C4            ; Test pulsante P4
       retlw     .4            ;
       movlw     b'11111101'   ; Abilito scansione riga 2
       movwf     PORTB         ;
       clrwdt                  ;
       btfss     C1            ; Test pulsante P5
       retlw     .5            ;
       btfss     C2            ; Test pulsante P6
       retlw     .6            ;
       btfss     C3            ; Test pulsante P7
       retlw     .7            ;
       btfss     C4            ; Test pulsante P8
       retlw     .8            ;
       movlw     b'11111011'   ; Abilito scansione riga 3
       movwf     PORTB         ;
       clrwdt                  ;
```

```
        btfss     C1            ; Test pulsante P9
        retlw     .9            ;
        btfss     C2            ; Test pulsante P10
        retlw     .10           ;
        btfss     C3            ; Test pulsante P11
        retlw     .11           ;
        btfss     C4            ; Test pulsante P12
        retlw     .12           ;
        movlw     b'11110111'   ; Abilito scansione riga 4
        movwf     PORTB         ;
        clrwdt                  ;
        btfss     C1            ; Test pulsante P13
        retlw     .13           ;
        btfss     C2            ; Test pulsante P14
        retlw     .14           ;
        btfss     C3            ; Test pulsante P15
        retlw     .15           ;
        btfss     C4            ; Test pulsante P16
        retlw     .16           ;
        movlw     b'11111111'   ; Disabilito scansione tastiera
        movwf     PORTB         ;
        retlw     .0            ; Torna dopo la CALL
; -------------------- subroutine RITARDO 20 mS ----------------------------
; con frequenza 3,2768 MHz
; --------------------------------------------------------------------------
DELA2M clrwdt                   ;
        movf      RTCC,0        ; Controllo se finiti 20mS
        SKPZ                    ;
        goto      DELA2M        ;
        movlw     .128          ;
        movwf     RTCC          ;
        return                  ; Ritorna dopo la CALL
;---------------------------------------------------------------
; START PROGRAM
;---------------------------------------------------------------
START   bsf       STAT,5        ; Seleziona SRAM banco 1
        movlw     b'0000'       ; RA0,RA1,RA2,RA3 out
        movwf     TR_A          ;
        movlw     b'11110000'   ; RB0...RB3 out   RB4...RB7 in
        movwf     TR_B          ;
        clrf      INTCON        ; Disabilita interrupt
        movlw     b'01000110'   ; Prescaler 1:128
        movwf     OPTIO         ; Copia W in OPTION
        bcf       STAT,5        ; Seleziona SRAM banco 0
        movlw     .128          ; Settaggio iniziale RTCC
        movwf     RTCC          ;
        movlw     b'0001'       ; Led off
        movwf     PORTA         ;
        movlw     b'11111111'   ; Disabilita scansione
        movwf     PORTB         ;
;---------------------- main program ----------------------------
MAIN    call      DELA2M        ; Subroutine ritardo 20mS
        call      TASTI         ; Scansiono tastiera
        movwf     FR03          ; Buffer pulsante premuto
        xorlw     .0            ;
        btfsc     STAT,2        ;
        goto      MAIN          ;
        bcf       LED           ; Accensione led
```

```
LPSIG  call      DELA2M        ; Ritardo complessivo x 100mS
       call      DELA2M        ;
       call      DELA2M        ;
       call      DELA2M        ;
       call      DELA2M        ;
       decfsz    FR03          ; Testo se scaduto numero cicli
       goto      LPSIG         ;
       bsf       LED           ; Spegnimento led
WAIFIN call      DELA2M        ;
       call      TASTI         ;
       xorlw     .0            ; Verifica tasto ancora premuto
       btfss     STAT,2        ;
       goto      WAIFIN        ;
       goto      MAIN          ;
;-----------------------------------------------------------------
INTERP                         ; Vettore interruzioni
       retfie                  ;
;-----------------------------------------------------------------
       END
```

Esaminiamo il programma nelle sue funzioni: il led resta spento fino a quando non viene premuto un pulsante della tastiera. Quando ciò avviene, il led si accende per un tempo pari a 100mS moltiplicato il numero del pulsante premuto, dopodichè il led si spegne fino a quando non viene premuto un altro pulsante. Se il pulsante viene lasciato premuto, il led si accenderà solo dopo il suo rilascio e la pressione di un nuovo pulsante.
La prima modifica ai precedenti programmi sta nella direzione delle porte: adesso la porta A è completamente di uscita, mentre la porta B è per metà di ingresso e per metà di uscita.
Il registro OPTION viene caricato con il bit 7 a "0", perchè vogliamo che siano abilitate le resistenze di pull-up sui pin di ingresso della porta B. Così facendo ci siamo risparmiati quattro resistenze da montare sul circuito.
Il ciclo principale del programma, prevede la chiamata alla subroutine di attesa 20mS e poi alla nuova subroutine "TASTI". Questa subroutine, torna con il valore 0 in W se nessun tasto è stato premuto, mentre, se premiamo un pulsante, torna caricando in W il valore corrispondente al tasto premuto. Per esempio il pulsante P1 ritornerà il valore 1, il pulsante P8 il valore 8 ecc.
Il valore che la subroutine pone in W, viene copiato nel registro FR03 e viene confrontato con zero. Se è uguale, nessun tasto è stato premuto e quindi si torna al MAIN, viceversa si accende il led e si attende un tempo pari a 100mS per il numero copiato. Al termine il led viene spento e si ripete la chiamata a "TASTI" per attendere il rilascio del pulsante. La procedura di attesa rilascio è complementare a quella di attesa pressione, ma implementata con la stessa tecnica dello XOR.
Proviamo allora a capire come lavorare con l'operazione XORLW (oppure XORWF). Come ben sappiamo, l'operazione di or esclusivo, restituisce valore "0" se i due (o più) ingressi sono uguali, altrimenti restituisce valore "1". Quindi, facendo lo XOR tra il valore decimale 0 ed il numero copiato in FR03, il bit numero 2 (zero bit) del registro di stato (STAT) varrà 1 se i due numeri sono uguali, 0 altrimenti. Con l'istruzione BTFSC STAT,2 si testa questo bit ed il gioco è fatto.
Ma passiamo ora allo studio della subroutine "TASTI". Inizialmente si pongono le quattro uscite della porta B (relative alle quattro righe della matrice) tutte ad alto livello. Poi si pone a basso livello la RB0 (riga 1) e si vanno a testare in sequenza le quattro colonne corrispondenti ai quattro pulsanti della prima riga. Se una di queste vale zero, il corripondente pulsante è stato premuto e quindi si troverà l'istruzione RETLW N, dove N

è il numero di ritorno in W. Se nessuno di questi pulsanti viene premuto, si riporta l'uscita della riga 1 ad alto livello e si procede nello stesso modo con la riga 2, fino ad arrivare alla quarta riga.
La subroutine "TASTI", è universale, nel senso che è possibile sia ridurla ad un minor numero di righe e/o colonne, sia aumentarla, compatibilmente con il numero di pin disponibili.
Con questo software avrete la possibilità di generare degli impulsi da 100mS a 1,6 secondi molto precisi, semplicemente impostandone il numero sulla tastiera.
-continua.