Lezione 3

Poichè al termine della seconda lezione ci siamo lasciati con un piccolo compito da svolgere, vediamo subito come potevamo implementare un led lampeggiante con frequenza di 2Hz, ovvero un secondo acceso ed un secondo spento.
Il circuito elettrico è lo stesso dello scorso mese, mentre il programma è il PROG04:

```
 TITLE 'PROG04: Led lampeggiante'
 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                 ; Ritardo 1S
;-----------------------------------------------------------------
       ORG       0
       goto      START         ;Reset vector
       nop
       nop
       nop
       goto      INTERP        ;Interrupt vector
; -------------------- 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'0010'       ; RA0,RA2,RA3 out   RA1 in
       movwf     TR_A          ;
       movlw     b'00000000'   ; RB0...RB7 out
       movwf     TR_B          ;
       clrf      INTCON        ; Disabilita interrupt
       movlw     b'11000110'   ; Prescaler 1:128
       movwf     OPTIO         ; Copia W in OPTION
       bcf       STAT,5        ; Seleziona SRAM banco 0
       clrf      PORTA         ; Uscite della porta A tutte a 0
       clrf      PORTB         ; Uscite della porta B tutte a 0
;---------------------- main program ----------------------------
       movlw     .128          ; Settaggio iniziale RTCC
       movwf     RTCC          ;
RIT1   movlw     .50           ; Carica 50 in W
       movwf     FR03          ; Copia W in FR03
RIT2   call      DELA2M        ; Vai alla subroutine DELA2M
```

```
        decfsz     FR03           ; Decrementa FR03, salta se 0
        goto       RIT2           ; Vai a RIT2
        movlw      b'00000001'    ; Carica W
        xorwf      PORTB,1        ; Esegui l'OR-esclusivo tra W e la
                                  ; porta B (toggla il led 0)
        goto       RIT1           ;
;-----------------------------------------------------------------
INTERP                            ; Vettore interruzioni
        retfie                    ;
;-----------------------------------------------------------------
        END
```

Si nota subito che le sole due istruzioni in più da inserire rispetto al programma PROG03.asm, sono le

```
movlw       b'00000001'
xorwf       PORTB
```

Vediamo perchè con queste due istruzioni si riesce a far accendere e spegnere alternativamente un led: l'operazione booleana OR-ESCLUSIVO, dati due bit in ingresso, restituisce un bit che vale zero se i due ingressi sono uguali, uno se i due ingressi sono diversi. Nel registro W era stato caricato il numero binario 00000001. Il secondo registro su cui operare era la porta B. Ora, qualsiasi valore fosse contenuto nei bit da 7 a 1 della porta B, con lo XOR con W lo avremmo ritrovato dopo l'operazione stessa, poichè lo XOR di 0 con 0 dà 0 e lo XOR di 1 con 0 dà 1. Ma al bit 0 viene eseguito lo XOR con un 1, quindi se tale bit era 1, dopo lo XOR il suo valore sarà 0 e viceversa.

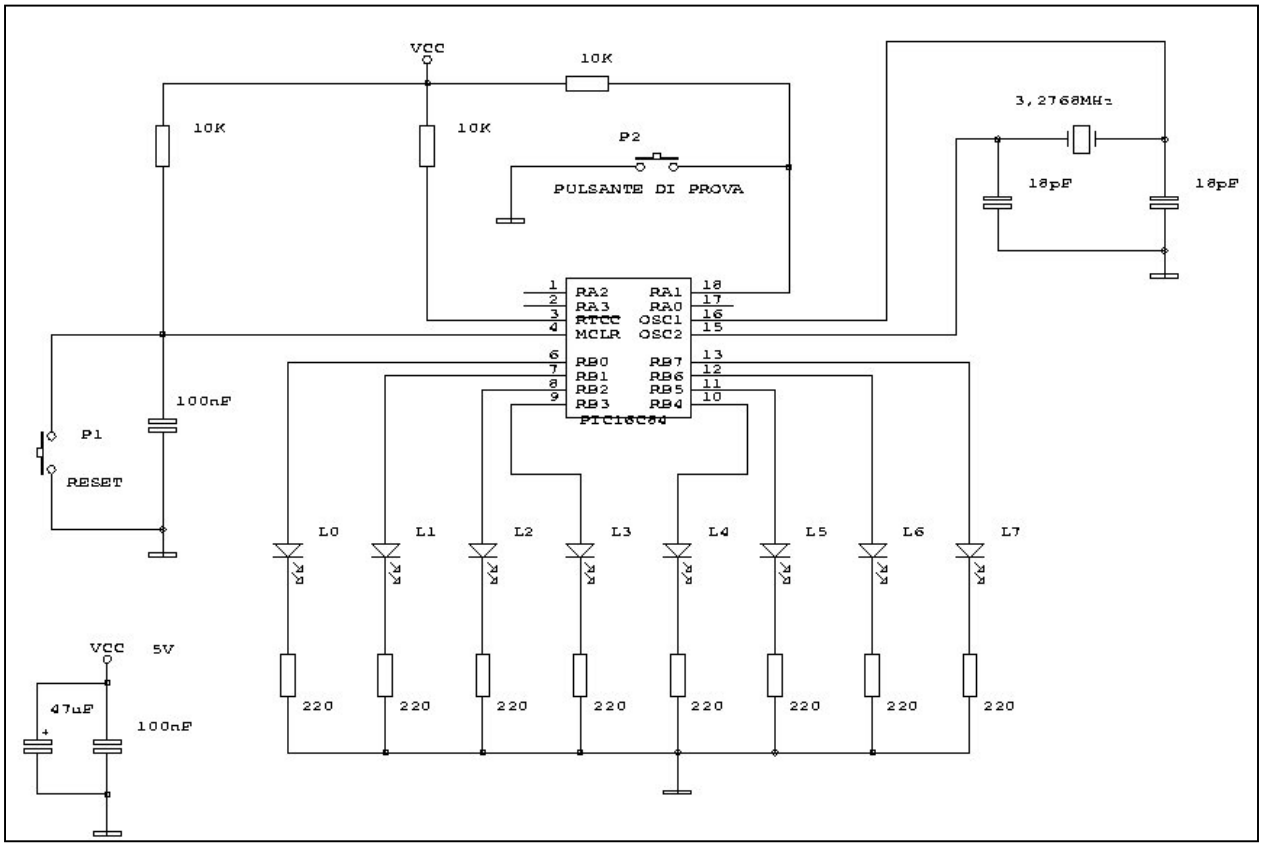

Figura 1. Schema elettrico per le prove con pulsante

Allo stesso modo era possibile, con le solite due istruzioni, far lampeggiare tutti i led alternativamente, semplicemente modificando il valore caricato in W. Per far lampeggiare il led a frequenza doppia invece, è sufficiente modificare il valore con cui viene caricato il registro FR03 da 50 in 25, oppure modificare il valore del prescaler da 1:128 a 1:64.
Dopo aver visto in modo abbastanza preciso come controllare le tempistiche, vediamo ora come reagire a stimoli esterni quali la pressione di un pulsante.

Interfacciamento con pulsanti

Prendendo in esame lo schema elettrico di figura 1, notiamo che il pulsante P2 è connesso sul pin RA1 ed ha una resistenza di pull-up da 10 KOhm. Questo significa che il controller, quando va a leggere il valore del bit 1 sulla porta A, trova un "1" se il pulsante non è premuto, mentre trova uno "0" se il pulsante è premuto. Naturalmente potevamo anche scegliere come ingresso la porta B, che ha già le resistenze di pull-up interne, ma in questo modo potete sfruttare il circuito già realizzato il mese scorso.
Quali sono le problematiche di un ingresso a pulsante? A prima vista nessuna, ma ci accorgeremo che, nella pratica, è possibile trovarsi di fronte a molti inconvenienti, tutti chiaramente risolvibili via software. Il primo di questi è il classico impulso di rumore che dura per pochi microsecondi, ma che potrebbe essere interpretato come la pressione del pulsante. Facciamo presente che noi andremo a testare il pin di ingresso pulsante con l'istruzione BTFSS oppure con l'istruzione BTFSC e che, in funzione del valore letto, stabiliremo se il pulsante è stato premuto o meno. Il secondo problema che può sorgere è il classico "rimbalzo", ovvero l'onda sinusoidale che viene generata dalla pressione del rimbalzo, per cui è possibile vedere il pulsante prima premuto e, dopo soli pochi millisecondi, lo stesso pulsante rilasciato, mentre invece non lo è.
Alla pressione di un pulsante la tensione sul pin del micro assume subito il massimo valore (o il minimo, a seconda se abbiamo resistenze di pull-up o di pull- down) ma dopo poco scende con andamento sinusoidale per poi risaliread una tensione inferiore della precedente e così via, fino all'annientarsi del fenomeno. La durata di queste oscillazioni può variare da poche centinaia da microsecondi a diversi millisecondi. Durante prove di laboratorio, abbiamo constatato che un buon filtro software deve lavorare per almeno 20 mS per annientare queste oscillazioni.
Vediamo allora come implementare un programma che rileva la pressione di un pulsante senza prendere impulsi spuri e senza farsi influenzare dai rimbalzi:

```
 TITLE 'PROG05: Acquisizione da pulsante'
 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                 ; Bit 0 = Memoria P2 per pressione
                               ; Bit 1 = Memoria P2 per rilascio
;-----------------------------------------------------------------
#define LED0     PORTB,0       ; Led 0 sulla porta B pin 0
```

```
#define P2       PORTA,1       ; Pulsante P2 su porta A pin 1
;-----------------------------------------------------------------
       ORG       0
       goto      START         ;Reset vector
       nop
       nop
       nop
       goto      INTERP        ;Interrupt vector
; -------------------- subroutine RITARDO 20 mS ----------------------------
; con frequenza 3,2768 MHz
; --------------------------------------------------------------------------
DELA2M bcf       FR03,0        ; Clear memoria P2 pressione
       bsf       FR03,1        ; Clear memoria P2 rilascio
WAIMS  clrwdt                  ;
       btfsc     P2            ; Testo stato P2 per pressione
       bsf       FR03,0        ;
       btfss     P2            ; Testo stato P2 per rilascio
       bcf       FR03,1        ;
       movf      RTCC,0        ; Controllo se finiti 20mS
       SKPZ                    ;
       goto      WAIMS         ;
       movlw     .128          ;
       movwf     RTCC          ;
       return                  ; Ritorna dopo la CALL
;-----------------------------------------------------------------
; START PROGRAM
;-----------------------------------------------------------------
START  bsf       STAT,5        ; Seleziona SRAM banco 1
       movlw     b'0010'       ; RA0,RA2,RA3 out   RA1 in
       movwf     TR_A          ;
       movlw     b'00000000'   ; RB0...RB7 out
       movwf     TR_B          ;
       clrf      INTCON        ; Disabilita interrupt
       movlw     b'11000110'   ; Prescaler 1:128
       movwf     OPTIO         ; Copia W in OPTION
       bcf       STAT,5        ; Seleziona SRAM banco 0
       clrf      PORTA         ; Uscite della porta A tutte a 0
       clrf      PORTB         ; Uscite della porta B tutte a 0
       movlw     .128          ; Settaggio iniziale RTCC
       movwf     RTCC          ;
;---------------------- main program ----------------------------
MAIN   call      DELA2M        ; Subroutine ritardo 20mS
       btfss     FR03,0        ; Testo P2 per pressione
       goto      ACCEND        ;
       btfsc     FR03,1        ; Testo P2 per rilascio
       goto      SPEGNI        ;
       goto      MAIN          ;
ACCEND bsf       LED0          ; Led 0 ON
       goto      MAIN          ;
SPEGNI bcf       LED0          ; Led 0 OFF
       goto      MAIN          ;
;-----------------------------------------------------------------
INTERP                         ; Vettore interruzioni
       retfie                  ;
;-----------------------------------------------------------------
       END
```

Cerchiamo di capire la logica di questo programma. Il programma principale, chiama in continuazione la subroutine di attesa fine 20mS, e non si preoccupa di testare direttamente lo stato del pulsante P2, ma va a vedere lo stato di due bit del registro FR03 che abbiamo impiegato

come memorie per lo stato del pulsante stesso. Quando richiamiamo la subroutine DELA2M, questi due bit vengono inizializzati rispettivamente a "0" ed a "1". Durante il ciclo di attesa di fine tempo, il pulsante P2 viene testato in continuazione dalle due istruzioni BTFSC e BTFSS. Se il pulsante viene rilasciato anche per pochi microsecondi, la memoria relativa allo stato di premuto (FR03 bit 0) si cancella, quindi al ritorno della subroutine non accetteremo lo stato di pressione tasto. Viceversa, se il pulsante viene premuto anche per pochi microsecondi, non rileveremo lo stato di rilascio pulsante.
Così facendo, abbiamo inserito un filtro software di durata prcisa di 20mS, filtro che, nel caso di implementazione hardware, avrebbe richiesto almeno un gruppo resistenza-condensatore e che non sarebbe stato così preciso. A questo vantaggio, si somma la possibilità di implementare dei tempi di rilevamento pressione e rilascio completamente differenti e lunghi quanto si vuole. In alcuni casi, per esempio, può essere utile non far partire un macchinario se prima non si è mantenuto premuto il pulsante di START per almeno N secondi. Situazioni di questo tipo sono facilmente gestibili via software, semplicemente inserendo dei registri che contano il numero di cicli da 20mS fino ad arrivare al tempo desiderato. Per esercizio provate a modificare il programma PROG05 per far in modo che il led si accenda un secondo dopo la pressione del pulsante e si spenga 3 secondi dopo. All'inizio, sembrerà una cosa complessa, ma con i programmi che avete già studiato, troverete il sistema per farlo.

Il reset hardware

Oltre ai vari reset di tipo software che possiamo avere (ad esempio la semplice istruzione "goto START"), quello più importante ai fini di un buon funzionamento è quello hardware, implementato sul piedino MCLR (Master CLeaR). Quando la tensione su questo pin scende ad uno zero logico, il chip si pone in reset fino a quando sullo stesso pin non rileva un "1" logico.
La soluzione più semplice e meno costosa, è quella applicata al circuito di figura 1: un condensatore verso massa ed una resistenza sul positivo. Quando viene fornita alimentazione al controller, sul pin MCLR avremo un basso livello fino a quando il condensatore non sarà completamente carico. Questa soluzione però crea inconvenienti quando si possono avere dei bruschi abbassamenti della tensione di alimentazione.
Allora, per prevenire questi casi, si debbono adottare soluzioni diverse, tra le quali quelle suggerite in figura 2-a e 2-b. In entrambi i casi, in qualsiasi momento la tensione di alimentazione scenda sotto una soglia prefissata, il chip esegue un reset. la tensione di soglia viene impostata nel primo caso con un diodo zener, nel secondo caso con un partitore resistivo e con la soglia

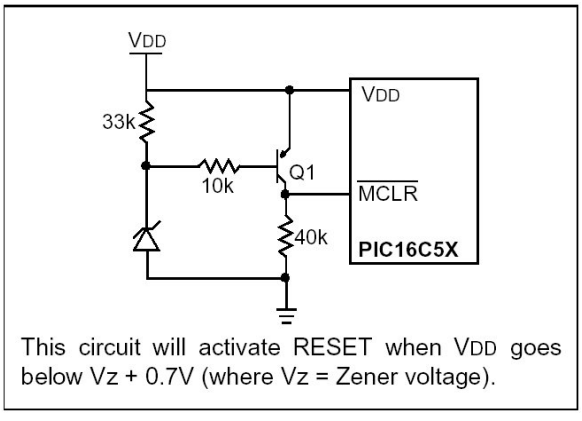

Figura 2-a. Reset con soglia a zener

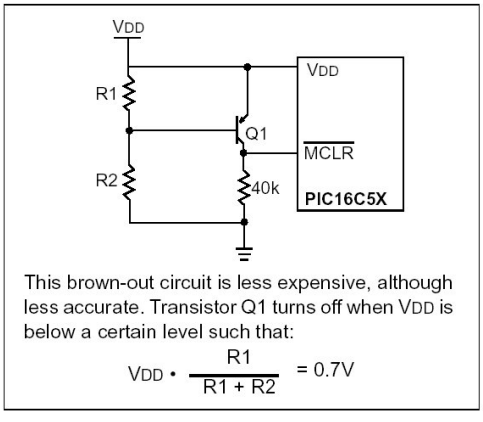

Figura 2-b. Reset con soglia a resistenza

stessa sulla base del transistor Q1.
Il reset, per un microcontroller, è una fase essenziale per un buon funzionamento, poichè con esso vengono inizializzati correttamente tutti i registri dedicati e le porte vengono messe in three-state per tutta la durata del reset stesso.
Oltre al pin di reset, dobbiamo dedicare attenzione anche ai pin dell'oscillatore: se viene impiegato un gruppo RC, i valori consigliati per la resistenza vanno da 2.200 Ohm a 100KOhm, mentre per il condensatore vanno da 10pF a 1.000pF. Chiaramente è possibile adottare anche valori diversi, ma prove pratiche non lo consigliano.
Nelle tabelle 1 e 2 invece, troviamo le capacità dei due condensatori da applicare sui pin dell'oscillatore rispettivamente nel caso di un oscillatore ceramico oppure quarzato.
-continua.

Osc Type	Resonator Freq	Cap. Range C1	Cap. Range C2
XT	455 kHz	22-100 pF	22-100 pF
	2.0 MHz	15-68 pF	15-68 pF
	4.0 MHz	15-68 pF	15-68 pF
HS	4.0 MHz	15-68 pF	15-68 pF
	8.0 MHz	10-68 pF	10-68 pF
	16.0 MHz	10-22 pF	10-22 pF

Tabella 1. Capacità per oscillatori ceramici

Osc Type	Resonator Freq	Cap.Range C1	Cap. Range C2
LP	32 kHz(1)	15 pF	15 pF
	100 kHz	15-30 pF	30-47 pF
	200 kHz	15-30 pF	15-82 pF
XT	100 kHz	15-30 pF	200-300 pF
	200 kHz	15-30 pF	100-200 pF
	455 kHz	15-30 pF	15-100 pF
	1 MHz	15-30 pF	15-30 pF
	2 MHz	15-30 pF	15-30 pF
	4 MHz	15-47 pF	15-47 pF
HS	4 MHz	15-30 pF	15-30 pF
	8 MHz	15-30 pF	15-30 pF
	20 MHz	15-30 pF	15-30 pF

Tabella 2. Capacità per oscillatori quarzati