Lezione 8

In questa puntata, ci dedicheremo all'impiego dei cosiddetti "interrupt", che tanto incutono temore ai più inesperti. Ma che cosa sono questi interrupt?
Come abbiamo già visto, un programma non è altro che una sequenza di istruzioni che il controllore decodifica ed esegue. La normale logica vuole che le istruzioni vengano eseguite una dopo l'altra, in modo tale da poter gestire il funzionamento del controller in modo "passo per passo". Ma ci sono eventi che può far comodo rilevare in qualsiasi momento dell'esecuzione di un programma, come per esempio l'arrivo di una comunicazione seriale, oppure l'overflow di un timer, oppure un fronte su di un pin. Ecco allora che entrano in gioco gli interrupt: ad un preciso evento richiesto, il controllore interompe (da interrupt!) quello che stava facendo e passa ad una subroutine detta "subroutine di interrupt".
Nel caso della seriale, tale subroutine si incaricherà di ricevere correttamente tutto il byte trasmesso dall'esterno. Al termine della subroutine di interrupt, il programma riprende dal punto in cui era stato lasciato.

Le nostre applicazioni

Le sorgenti di interrupt disponibili sul PIC16C84 sono quattro: un fronte programmabile sul pin RB0, l'overflow del timer TMR0, il cambio di stato di un pin della porta "B" da RB7 a RB4 ed il termine della fase di scrittura nella EEPROM dei dati.
Vedremo allora come si possono scrivere programmi che lavorano con interrupt, trasportabili su ogni microcontrollore della famiglia PIC16CXXX e 17CXX.

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7: **GIE**: Global Interrupt Enable bit
1 = Enables all un-masked interrupts
0 = Disables all interrupts

Note: For the operation of the interrupt structure, please refer to Section 8.5.

bit 6: **EEIE**: EE Write Complete Interrupt Enable bit
1 = Enables the EE write complete interrupt
0 = Disables the EE write complete interrupt

bit 5: **T0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4: **INTE**: RB0/INT Interrupt Enable bit
1 = Enables the RB0/INT interrupt
0 = Disables the RB0/INT interrupt

bit 3: **RBIE**: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2: **T0IF**: TMR0 overflow interrupt flag bit
1 = TMR0 has overflowed (must be cleared in software)
0 = TMR0 did not overflow

bit 1: **INTF**: RB0/INT Interrupt Flag bit
1 = The RB0/INT interrupt occurred
0 = The RB0/INT interrupt did not occur

bit 0: **RBIF**: RB Port Change Interrupt Flag bit
1 = When at least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

Figura 1. Configurazione del registro INTCON

Ma prima di passare ai programmi di esempio, andiamo a vedere quali risorse hardware ci vengono messe a disposizione dal PIC per poter usufruire di questi interrupt.
In figura 1, abbiamo il registro di gestione degli interrupt: analizziamolo in dettaglio. Il bit 0, RBIF, (RB Interrupt Flag) viene settato dall'hardware quando sui pin RB4, RB5, RB6, RB7 si ha un cambio di stato ed il reset deve essere fatto via software. Il bit 1, INTF, (INTerrupt Flag) viene settato dall'hardware quando avviene un interruzione per un fronte sul pin RB0. A proposito di questo interrupt, dobbiamo precisare che la programmazione del fronte avviene impostando il bit 6

(INTEDG) del registro OPTION, come si vede in figura 2. Anche in questo caso il reset viene fatto via software. Il bit 2, RTIF, (RTcc Interrupt Flag) viene settato anch'esso dall'hardware quando il timer TMR0 va in overflow. Il reset deve sempre avvenire via software. Il bit 3, RBIE, è il bit che abilita o meno il generarsi di un interrupt sul cambio di stato della porta RB<7..4>. Il bit 4, INTE, abilita o meno il generarsi di un interrupt ad ogni fronte sul pin RB0. Il bit 5, RTIE, abilita o meno un interrupt dal timer TMR0, mentre il bit 6, EEIE, abilita o meno il generarsi di un interrupt al termine di una eventuale scrittura sulla EEPROM dei dati. Infine il bit 7, GIE, (Global Interrupt Enable) abilita o meno tutte le precedenti sorgenti di interrupt.

Lo schema elettrico del prototipo necessario alle nostre prove è quello di figura 3. Per come sono stati configurati i pin, si vede subito che la sorgente di interrupt relativa al fronte sul pin RB0 non sarà disponibile, in quanto tale pin viene impiegato come uscita. Sfrutteremo invece l'interrupt della porta RB<7..4> e quello dell'overflow del timer.

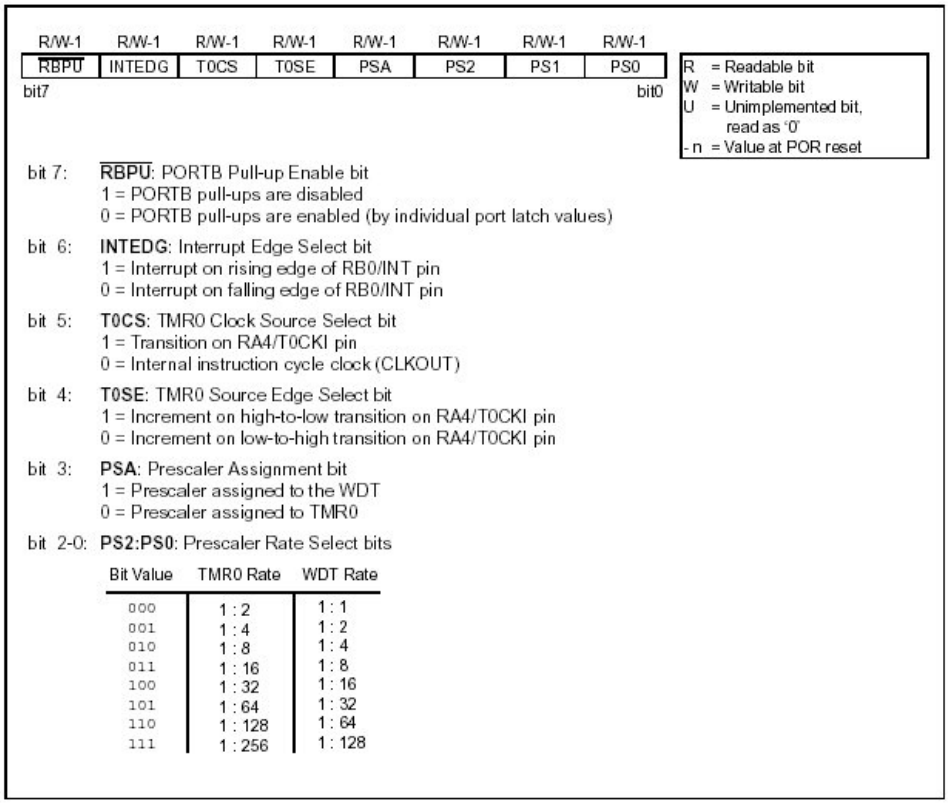

Figura 2. Configurazione del registro OPTION

Vediamo allora il programma:

```
 TITLE 'PROG15: Prova 1 interrupt RB<7..4>'
 list F=INHX8M,P=16C841
;-----------------------------------------------------------------
RTCC    EQU 01H                 ; Real Time Clock Counter
PCL     EQU 02H                 ; Program Counter
STAT    EQU 03H                 ; Registro di stato
```

```
PORTA  EQU 05H                  ; Porta A
PORTB  EQU 06H                  ; Porta B
EDATA  EQU 08H                  ; Data EEPROM
EADR   EQU 09H                  ; Address EEPROM
ECON1  EQU 88H                  ; Stato delle operaz. in EEPROM
ECON2  EQU 89H                  ; Vedi ECON1
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                  ;
FR02   EQU 0DH                  ;
FR03   EQU 0EH                  ;
STEMP  EQU 02EH                 ; Registro memorizzazione stato
WTEMP  EQU 02FH                 ; Registro memorizzazione W
;-----------------------------------------------------------------
#define RP0    STAT,5           ; Flag selezione banco ram
#define RBIF   INTCON,0         ; RB Interrupt Flag
#define RBIE   INTCON,3         ; RB Interrupt Enable
#define GIE    INTCON,7         ; Global Interrupt Enable
;-----------------------------------------------------------------
#define P1 PORTB,7              ; Pulsante 1
#define P2 PORTB,6              ; Pulsante 2
#define P3 PORTB,5              ; Pulsante 3
#define P4 PORTB,4              ; Pulsante 4
#define D1 PORTB,0              ; Led 1
#define D2 PORTB,1              ; Led 2
#define D3 PORTB,2              ; Led 3
#define D4 PORTB,3              ; Led 4
;-----------------------------------------------------------------
       ORG       0
       goto      START         ;Reset vector

       ORG       4             ;Interrupt vector
       movwf     WTEMP         ;potrebbe essere in banco 0 o 1
       swapf     STAT,0        ;Swap STAT in TEMP
       bcf       RP0           ;Selezione banco 0
       movwf     STEMP         ;Memorizzo stato
                               ;
       btfss     RBIF          ;Interrupt da RB?
       goto      ENDINT        ;Uscita dalla subroutine di int.
       decfsz    FR03          ;Test se FR03 = 0
       goto      ENDINT        ;per rilascio
       decfsz    FR02          ;Test se FR02 = 0
       goto      RILASC        ;per rilascio
       comf      PORTB,0       ;Copia porta B complement. in W
       andlw     b'11110000'   ;Maschera per pulsanti
       movwf     FR01          ;Copia input in FR01
       swapf     FR01,0        ;Swap FR01 in W
       xorwf     PORTB         ;OR-EX tra porta B e W (toggle)
       bcf       RBIF          ;Reset RBIF
       movlw     .10           ;Carico 10 in W
       movwf     FR02          ;Copio W in FR02
RILASC movlw     .100          ;Carico 100 in W
       movwf     FR03          ;Copio W in FR03
                               ;
```

```
ENDINT bsf       GIE           ;Abilita interrupt RB
       bcf       RP0           ;
       swapf     STEMP,0       ;Ripristino stato
       movwf     STAT          ;
       swapf     WTEMP,1       ;Ripristino W
       swapf     WTEMP,0       ;
       retfie                  ;
;-----------------------------------------------------------------
; START PROGRAM
;-----------------------------------------------------------------
START  clrf      PORTB         ; Azzera uscite su porta B
       bsf       STAT,5        ; Seleziona SRAM banco 1
       movlw     b'0000'       ; RA out
       movwf     TR_A          ;
       movlw     b'11110000'   ; RB0..RB3 out   RB4..RB7 in
       movwf     TR_B          ;
       clrf      INTCON        ; Disabilita interrupt
       bsf       RBIE          ; Abilita RB Interrupt
       bsf       GIE           ; Abilita Global 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'       ;
       movwf     PORTA         ;
;---------------------- main program ----------------------------
MAIN   clrwdt                  ;
       goto      MAIN          ;
;-----------------------------------------------------------------
       END
```

Con questo sorgente, abilitiamo il solo interrupt relativo alla variazione di stato della porta RB sui pin da 7 a 4. Per prima cosa si abilita l'RBIE e, successivamente, il GIE. Ciò implica che, ogni volta che premiamo un pulsante, verrà generato un interrupt di tipo RBIF, settando il bit corrispondente e saltando al vettore degli interrupt dopo ORG 4.
Le prime operazioni da eseguire sono il salvataggio del registro W e del registro di stato, perchè al termine della subroutine di interrupt sarà necessario riaverli con gli stessi valori contenuti prima dell'interrupt. Poi si va a testare se l'interrupt è stato generato proprio dalla variazione della porta RB<7..4> ed infine si passa alla routine vera e propria di gestione dell'evento. Qui abbiamo implementato un ritardo di alcuni millisecondi prima di togglare il led corrispondente al pulsante premuto, per il solito problema del rimbalzo sul pulsante. Al termine della gestione dell'evento, passiamo al ripristino del registro di stato e del registro W.
Come potete osservare, il ciclo MAIN è vuoto, ossia il programma "looppa" in continuazione sulle due istruzioni CLRWDT e GOTO MAIN. Non appena un pulsante viene premuto, l'interrupt relativo si attiva e si passa alla sua gestione. In questo modo, è possibile inserire nel loop del MAIN il programma vero e proprio, senza più curarsi della gestione dei pulsanti.

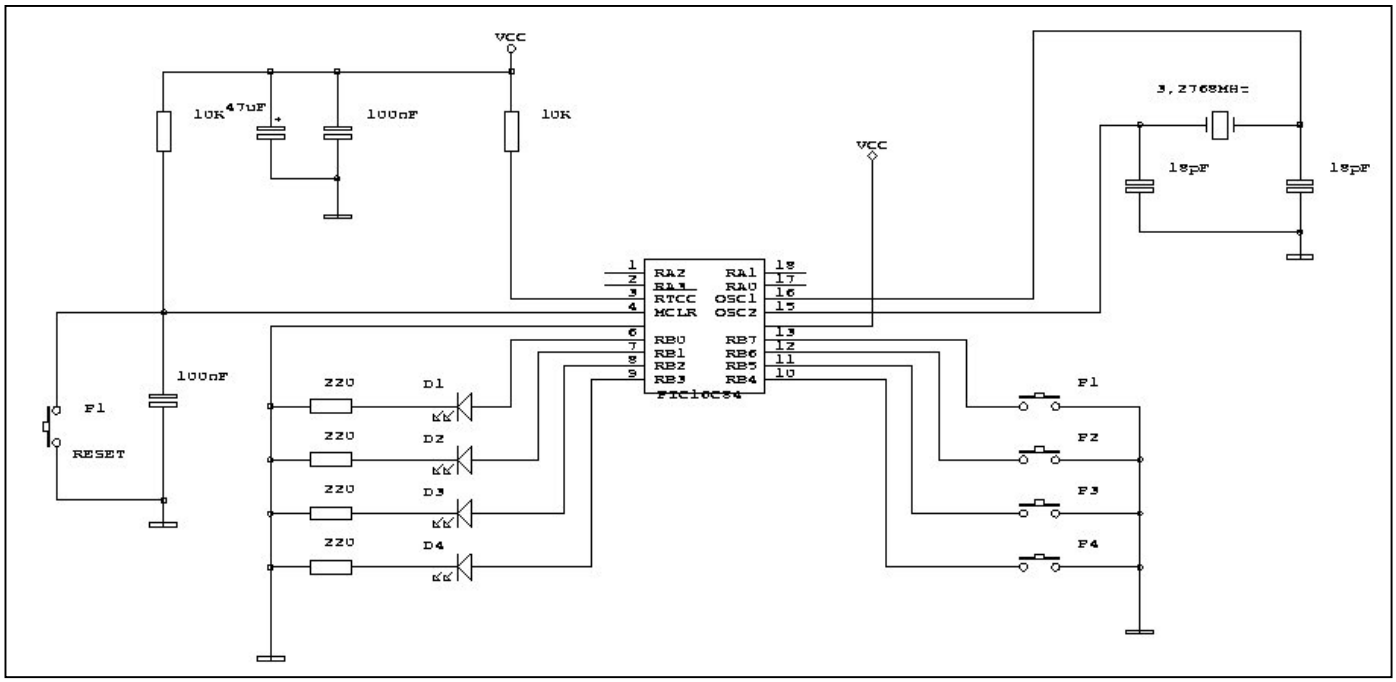

Figura 3. Schema elettrico del circuito di test

-continua.