Lezione 9

I microcontrollori della Microchip, hanno una caratteristica che non tutti possiedono: la possibilità di andare in SLEEP, ovvero di "addormentarsi" durante l'esecuzione di un qualsiasi programma, ovviamente sotto il controllo del progettista!
Questa feature, fa sì che il chip assorba pochissimo, infatti si registrano correnti inferiori al microamper, permettendo di fatto un'agevole alimentazione a pile o a batteria.
Una applicazione che sfrutta tipicamente questa caratteristica è la tastiera a combinazione per cassaforti: in questo caso c'è la necessità di contenere i consumi per evitare di dover far intervenire un fabbro nel momento in cui le pile cessano la loro corretta erogazione di energia. Un'altra classica applicazione è il telecomando ad infrarossi del televisore.

L'istruzione SLEEP

Durante la stesura del programma, l'istruzione che ci permette di far entrare il controller in modalità sleep è proprio l'istruzione chiamata "SLEEP"!
Quando si giunge a questa istruzione (che non ha operandi), il contatore del watchdog ed il suo prescaler vengono azzerati, il bit TO (Time-Out) del registro di stato viene settato a "1" ed il bit PD (Power Down) dello stesso registro viene posto a "0". Inoltre, l'oscillatore viene bloccato.
Attenzione però quando parliamo di oscillatore, perchè con questo termine intendiamo sempre l'oscillatore relativo al clock del micro, in quanto il micro ha al suo interno anche un oscillatore per l'watchdog ed uno eventuale per il convertitore. Quindi, se noi blocchiamo l'oscillatore del clock, non significa che li blocchiamo tutti. Per quanto riguarda l'oscillatore del watchdog comunque, per determinate applicazioni è possibile disabilitarlo senza compromettere la funzionalità del sistema e, in ogni caso, non assorbe la stessa corrente dell'oscillatore di clock.
Quando il chip si "risveglia", ovvero l'oscillatore di clock riprende a funzionare, si deve attendere un tempo di latenza pari a 1.024 per il tempo di clock prima che il program counter indirizzi l'istruzione da eseguire ed il controller la esegua. In pratica, se abbiamo un clock a 4 MHz, con l'istruzione di 1 microsecondo, avremo il risveglio dopo 1.024 x 1 = 1.024 microsecondi, cioè circa 1 millisecondo, come si può vedere nella tabella 1 e nella figura 1.

Oscillator Configuration	Power-up		Wake-up from SLEEP
	PWRT Enabled	PWRT Disabled	
XT, HS, LP	72 ms + 1024TOSC	1024TOSC	1024TOSC
RC	72 ms	—	—

Tabella 1. Tempistica di risveglio del controller

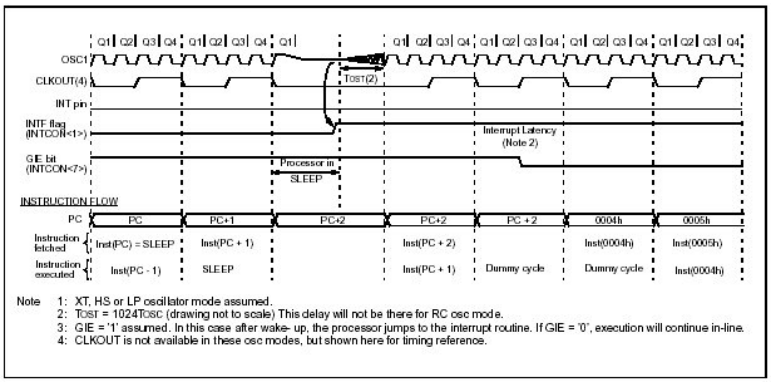
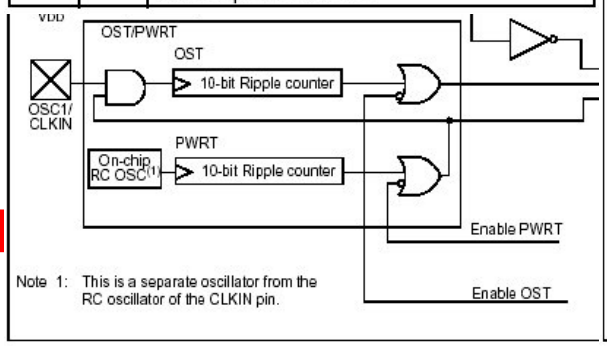


Figura 1. Timing-diagram dopo un'istruzione SLEEP

Dobbiamo far necessariamente notare che ben diverso è il risveglio da interrupt o da reset, infatti nel primo caso il program counter si carica con l'indirizzo del vettore di interrupt, nel secondo caso si carica con l'indirizzo del vettore di reset. In figura 2 è possibile vedere il diagramma a blocchi dell'intero sistema di reset del PIC16C84, mentre nella tabella 2 si può capire che cosa cambia nei registri dopo tale operazione.
In effetti, dopo un qualsiasi reset, è possibile capire quale sia stata la causa andando ad interrogare i bit TO e PD del registro di stato. In figura 3 riportiamo il contenuto del registro di stato, sempre utile da avere sott'occhio, mentre dalla tabella 3 possiamo capire quale sia stata la causa del reset o del risveglio.

Register	Address	Power-on Reset	$\overline{MCLR}$ Reset during: – normal operation – SLEEP WDT Reset during normal operation	Wake-up from SLEEP: – through interrupt – through WDT Time-out
W	—	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF	00h	---- ----	---- ----	---- ----
TMR0	01h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	02h	0000h	0000h	PC + 1(2)
STATUS	03h	0001 1xxx	000q quuu(3)	uuuq quuu(3)
FSR	04h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTA	05h	---x xxxx	---u uuuu	---u uuuu
PORTB	06h	xxxx xxxx	uuuu uuuu	uuuu uuuu
EEDATA	08h	xxxx xxxx	uuuu uuuu	uuuu uuuu
EEADR	09h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCLATH	0Ah	---0 0000	---0 0000	---u uuuu
INTCON	0Bh	0000 000x	0000 000u	uuuu uuuu(1)
INDF	80h	---- ----	---- ----	---- ----
OPTION_REG	81h	1111 1111	1111 1111	uuuu uuuu
PCL	82h	0000h	0000h	PC + 1
STATUS	83h	0001 1xxx	000q quuu(3)	uuuq quuu(3)
FSR	84h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TRISA	85h	---1 1111	---1 1111	---u uuuu
TRISB	86h	1111 1111	1111 1111	uuuu uuuu
EECON1	88h	---0 x000	---0 q000	---0 uuuu
EECON2	89h	---- ----	---- ----	---- ----
PCLATH	8Ah	---0 0000	---0 0000	---u uuuu
INTCON	8Bh	0000 000x	0000 000u	uuuu uuuu(1)

$\overline{TO}$	$\overline{PD}$	Condition
1	1	Power-on Reset
0	x	Illegal, $\overline{TO}$ is set on $\overline{POR}$
x	0	Illegal, $\overline{PD}$ is set on $\overline{POR}$
0	1	WDT Reset (during normal operation)
0	0	WDT Wake-up
1	1	$\overline{MCLR}$ Reset during normal operation
1	0	$\overline{MCLR}$ Reset during SLEEP or interrupt wake-up from SLEEP



R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit7							bit0

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

bit 7: **IRP**: Register Bank Select bit (used for indirect addressing)
0 = Bank 0, 1 (00h - FFh)
1 = Bank 2, 3 (100h - 1FFh)
The IRP bit is not used by the PIC16F8X. IRP should be maintained clear.

bit 6-5: **RP1:RP0**: Register Bank Select bits (used for direct addressing)
00 = Bank 0 (00h - 7Fh)
01 = Bank 1 (80h - FFh)
10 = Bank 2 (100h - 17Fh)
11 = Bank 3 (180h - 1FFh)
Each bank is 128 bytes. Only bit RP0 is used by the PIC16F8X. RP1 should be maintained clear.

bit 4: $\overline{TO}$: Time-out bit
1 = After power-up, CLRWDT instruction, or SLEEP instruction
0 = A WDT time-out occurred

bit 3: $\overline{PD}$: Power-down bit
1 = After power-up or by the CLRWDT instruction
0 = By execution of the SLEEP instruction

bit 2: **Z**: Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1: **DC**: Digit carry/$\overline{borrow}$ bit (for ADDWF and ADDLW instructions) (For $\overline{borrow}$ the polarity is reversed)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

bit 0: **C**: Carry/$\overline{borrow}$ bit (for ADDWF and ADDLW instructions)
1 = A carry-out from the most significant bit of the result occurred
0 = No carry-out from the most significant bit of the result occurred
Note: For $\overline{borrow}$ the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high or low order bit of the source register.

Il software di test

Per valutare praticamente l'istruzione SLEEP, abbiamo bisogno di costruirci una piccola basetta di prova il cui schema elettrico è visibile in figura 4.

Il programma che proponiamo invece, è siglato PROG19 e funziona in questo modo: dopo l'accensione si mette in sleep e si risveglia solo quando uno dei quattro pulsanti viene premuto. In tal caso accende il led corrispondente, attende il rilascio del pulsante e poi si rimette in sleep.

Tabella 3. Effetti di un reset sui bit TO e PD del registro di stato

Figura 3. Il registro di stato

```
TITLE 'PROG19: Prova SLEEP'
;-----------------------------
                Time Clock Counter
                ram Counter
STAT    EQU 03H                 ; Registro di stato
PORTA   EQU 05H                 ; Porta A
PORTB   EQU 06H                 ; Porta B
EDATA   EQU 08H                 ; Data EEPROM
EADR    EQU                       ress EEPROM
ECON1   EQU                     to delle operaz. in EEPROM
ECON2   EQU                     i ECON1
INTCON  EQU                     , 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                 ;
WTEMP   EQU 10H                 ;
STEMP   EQU 11H                 ;
;---------------------------------------------------------------
#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 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 PD      STAT,3          ; Power Down bit
;---------------------------------------------------------------
        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    DORMI           ;Uscita dalla subroutine di int.
        btfss   P1              ; Testo se premuto P1
        goto    PULS1           ;
```

```
        btfss     P2              ; Testo se premuto P2
        goto      PULS2           ;
        btfss     P3              ; Testo se premuto P3
        goto      PULS3           ;
        btfss     P4              ; Testo se premuto P4
        goto      PULS4           ;
        goto      DORMI           ;
PULS1   movlw     b'11110001'     ; Accensione led 1
        movwf     PORTB           ;
        goto      RIL             ; Memorizza nuovo stato
PULS2   movlw     b'11110010'     ; Accensione led 2
        movwf     PORTB           ;
        goto      RIL             ; Memorizza nuovo stato
PULS3   movlw     b'11110100'     ; Accensione led 3
        movwf     PORTB           ;
        goto      RIL             ; Memorizza nuovo stato
PULS4   movlw     b'11111000'     ; Accensione led 4
        movwf     PORTB           ;
        goto      RIL             ; Memorizza nuovo stato
RIL     call      DELA2M          ; Attesa rilascio pulsante
        btfss     P1              ;
        goto      RIL             ;
        btfss     P2              ;
        goto      RIL             ;
        btfss     P3              ;
        goto      RIL             ;
        btfss     P4              ;
        goto      RIL             ;
DORMI   movlw     b'0000'         ; Setta uscite PORTA a zero
        movwf     PORTA           ;
        movlw     b'11110000'     ; Setta uscite PORTB a zero
        movwf     PORTB           ;
        bcf       RBIF            ; Reset RBIF
                                  ;
ENDRB   bsf       RP0             ;
        bsf       GIE             ; Abilita interrupt
        bsf       RBIE            ; Abilita RB Interrupt
        bcf       RP0             ;
        swapf     STEMP,0         ; Ripristino stato
        movwf     STAT            ;
        swapf     WTEMP,1         ; Ripristino W
        swapf     WTEMP,0         ;
        sleep                     ;
;---------------------- subroutine RITARDO 20 mS ----------------
; con frequenza 3,2768 MHz
;----------------------------------------------------------------
DELA2M bcf        FR01,7          ; Memoria P1
        bcf       FR01,6          ; Memoria P2
        bcf       FR01,5          ; Memoria P3
        bcf       FR01,4          ; Memoria P4
LPDEL   clrwdt                    ; Azzera watchdog
        nop                       ; Nessuna operazione
        nop                       ;        "
        nop                       ;        "
        nop                       ;        "
        movf      RTCC,0          ; Controllo se finiti 20mS
        SKPZ                      ;
```

```
       goto      LPDEL         ;
       movlw     .128          ; Carico 128 in W
       movwf     RTCC          ; Copio W nel registro RTCC
       return                  ; Ritorna dopo la CALL
;-----------------------------------------------------------------
; START PROGRAM
;-----------------------------------------------------------------
START  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
       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'0000'       ; Setta a zero uscite su PORTA
       movwf     PORTA         ;
;---------------------- main program ----------------------------
MAIN   call      DELA2M        ; Attesa 20mS
       goto      DORMI         ;
;-----------------------------------------------------------------
       END
```

Come si vede chiaramente, il corpo del programma quasi non esiste, essendo tutto gestito sotto interrupt.Non ci metteremo adesso a spiegare riga per riga che cosa accade, perchè dovreste essere in grado di capirlo da soli, dato che sono eseguite operazioni già viste in precedenza, ma preferiamo dare alcuni consigli sull'impiego dell'istruzione sleep.
Prima di tutto, dobbiamo tener presente che l'assorbimento di tutto un circuito, non è dato solamente dall'assorbimento del chip, ma dalla totalità degli assorbimenti. Quindi, se per esempio una volta in sleep facciamo rimanere un led acceso, l'assorbimento totale sarà dato proprio dall'assorbimento di tale led. Per questo motivo, nel nostro esempio abbiamo deliberatamente lasciato le porte non utilizzate (porta A) e quelle relative dei led a zero, ma ciò non basta, perchè così facendo una piccola corrente continuerà a scorrere su quei pin, basterà che lo proviate con un multimetro.
Allora dovrete, prima di inserire l'istruzione sleep, portare in ingresso (che equivale ad un three-state) tutti i pin con la modifica dei registri TRISX, dove X indica la porta da settare.

Quando poi il chip si risveglierà, dovrete rimpostare le direzioni delle porte, come dopo ogni reset. Noterete che così facendo otterrete assorbimenti inferiori al microamper (sempreché abbiate disabilitato l'watchdog).

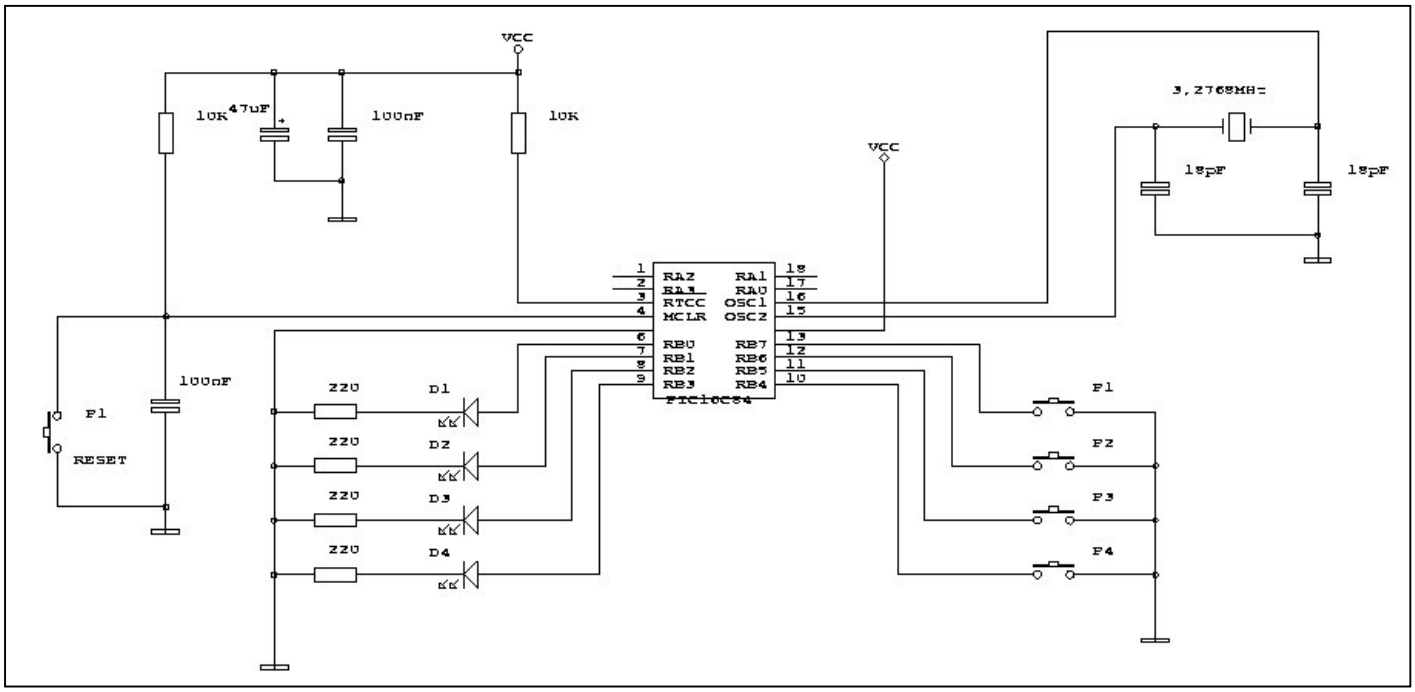


Figura 4. Schema per la prova del software

-continua.