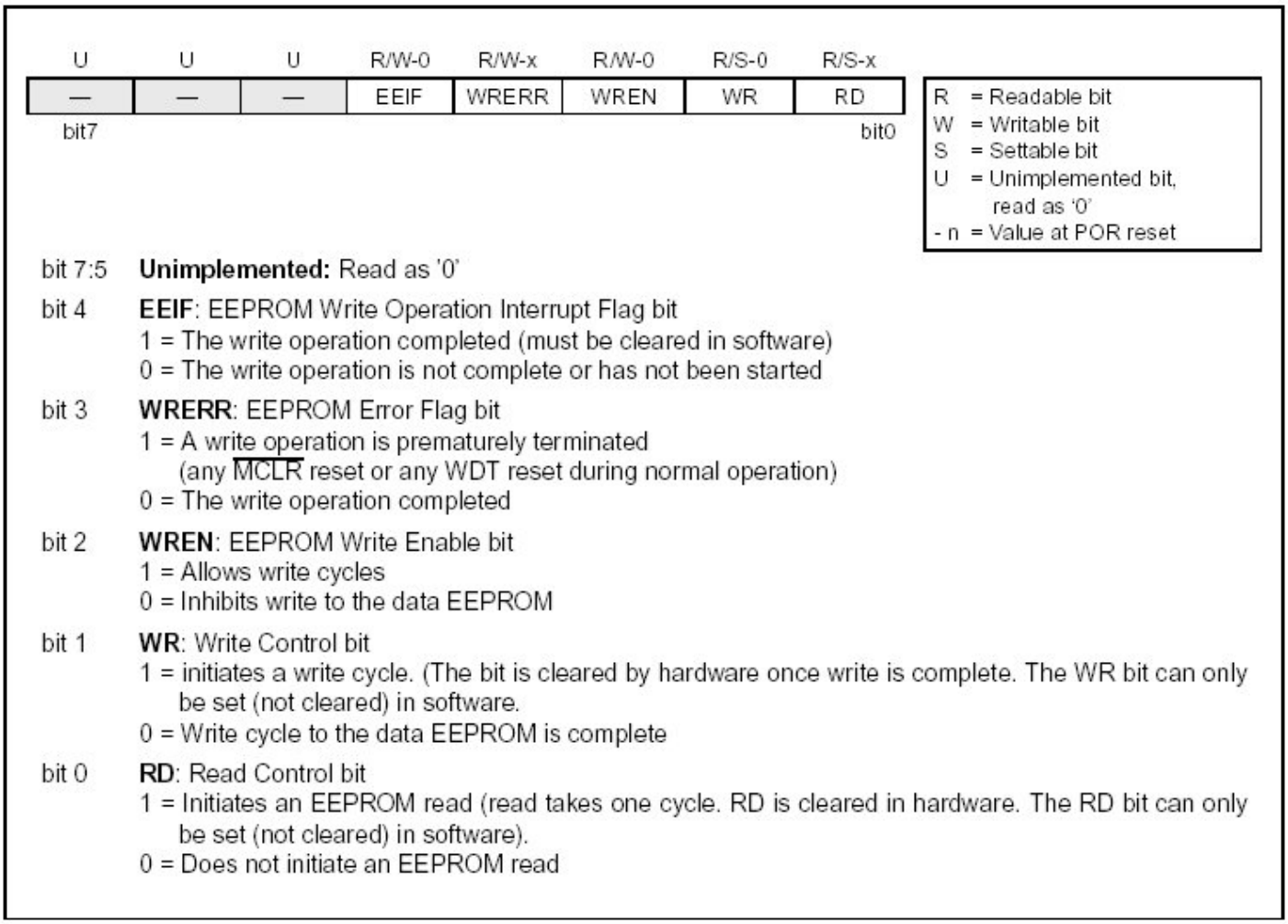
Lezione 6

Come già anticipato nelle prime lezioni, il PIC16C84 non solo ha la memoria di programma in tecnologia EEPROM, ma possiede anche 64 registri da 8 bit di identica struttura. Questo vuol dire che, oltre alla riprogrammabilità del controller senza necessità di lampade ad ultravioletti, è possibile immagazzinarvi dei dati che non saranno persi anche dopo aver tolto l'alimentazione.
Questa proprietà, fa preferire il 16C84 ad altri in varie applicazioni, prima tra tutte quella dei telecomandi ad autoapprendimento o a rolling-code (codice variabile).
Tutti gli altri controller che non hanno questa possibilità, devono essere affiancati da una EEPROM seriale esterna, con conseguente spreco di spazio, sia fisico, che di linee di programma, poichè l'interfacciamento con le EEPROM seriali avviene con protocolli da implementare via software. Con il PIC16C84 invece, la scrittura o la lettura di una cella di memoria (byte) avviene semplicemente con pochissime istruzioni dedicate.

L'hardware dedicato.

Il PIC16C84 ha quattro registri dedicati che gestiscono l'intera memoria EEPROM e le operazioni su di essa: EECON1, EECON2, EEDATA e EEADR. Il registro EEDATA è il buffer di scambio per la lettura e la scrittura della memoria: per scrivere un byte in una cella, il byte stesso viene prima copiato in EEDATA. Viceversa, per leggere il contenuto di una cella, dopo aver dato alcune informazioni che vedremo, troviamo il dato nel registro EEDATA.
Il numero di celle (byte) gestite è di 64 ed il loro indirizzamento va da 00h a 3Fh. Quando un byte viene scritto (o ricritto), prima viene automaticamente cancellato dal micro stesso, per garantire una perfetta memorizzazione del dato.
Il tempo di memorizzazione è di circa 10mS ed è controllato da un timer dedicato interno al controller. Il tempo dato varia comunque con la temperatura e la tensione di alimentazione.
Una importante opzione, consente di proteggere non solo la memoria di programma da letture non autorizzate, ma anche la memoria dei dati.
Il registro EEADR è il registro dedicato dove inserire l'indirizzo della cella su cui eseguire l'operazione e, ovviamente può indirizzare fino a 256 byte di memoria, ma soltanto i primi 64 sono disponibili.
In figura 1 troviamo la segmentazione del registro EECON1: il bit 0 (RD) dà il via alla lettura di una cella. Il tempo necessario è quello di una istruzione. Per questa operazione, il bit deve essere settato via software, poi, al termine, viene azzerato dall'hardware del micro. Il bit numero 1 (WR)invece, dà il via al ciclo di scrittura di una cella. Anche in questo caso, viene settato via software e azzerato dall'hardware a fine operazione. Con il bit numero 2 (WREN) è possibile proteggere i dati della EEPROM da accidentali scritture, perchè se posto a "0" inibisce l'operazione di scrittura. Se settato, la consente. Il bit numero 3 (WRERR) dà indicazione dell'avvenuta memorizzazione o meno. Se azzerato, l'operazione si è conclusa correttamente, se settato, la scrittura non è avvenuta, ad esempio perchè sopraggiunto un MCLR oppure per intervento del watchdog.
Il bit numero 4 (EEIF) è simile al precedente, ma con lo stato invertito e deve essere azzerato via software.
Gli ultimi tre bit non vengono usati. Il registro EECON2 invece non è un registro fisico vero e proprio, ma soltanto una mappatura per il chip.



U	U	U	R/W-0	R/W-x	R/W-0	R/S-0	R/S-x
—	—	—	EEIF	WRERR	WREN	WR	RD
bit7							bit0

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

bit 7:5 **Unimplemented:** Read as '0'

bit 4 **EEIF**: EEPROM Write Operation Interrupt Flag bit
1 = The write operation completed (must be cleared in software)
0 = The write operation is not complete or has not been started

bit 3 **WRERR**: EEPROM Error Flag bit
1 = A write operation is prematurely terminated
(any $\overline{\text{MCLR}}$ reset or any WDT reset during normal operation)
0 = The write operation completed

bit 2 **WREN**: EEPROM Write Enable bit
1 = Allows write cycles
0 = Inhibits write to the data EEPROM

bit 1 **WR**: Write Control bit
1 = initiates a write cycle. (The bit is cleared by hardware once write is complete. The WR bit can only be set (not cleared) in software.
0 = Write cycle to the data EEPROM is complete

bit 0 **RD**: Read Control bit
1 = Initiates an EEPROM read (read takes one cycle. RD is cleared in hardware. The RD bit can only be set (not cleared) in software).
0 = Does not initiate an EEPROM read

Figura 1. Interno del registro EECON1

In figura 2 vediamo il circuito elettrico da realizzare per fare alcuni esperimenti con la memoria EEPROM interna. Sono stati connessi quattro pulsanti sugli ingressi RB4..RB7 e quattro led sulle uscite RB0..RB3, mentre il clock è sempre a 3,2768MHz. I quattro pulsanti sono stati connessi su quei pin per un ben preciso motivo: quando spiegheremo le varie sorgenti di interrupt, una di queste arriverà dal cambio di stato di un pin della porta B<4..7>.
Ma passiamo al programma PROG10.

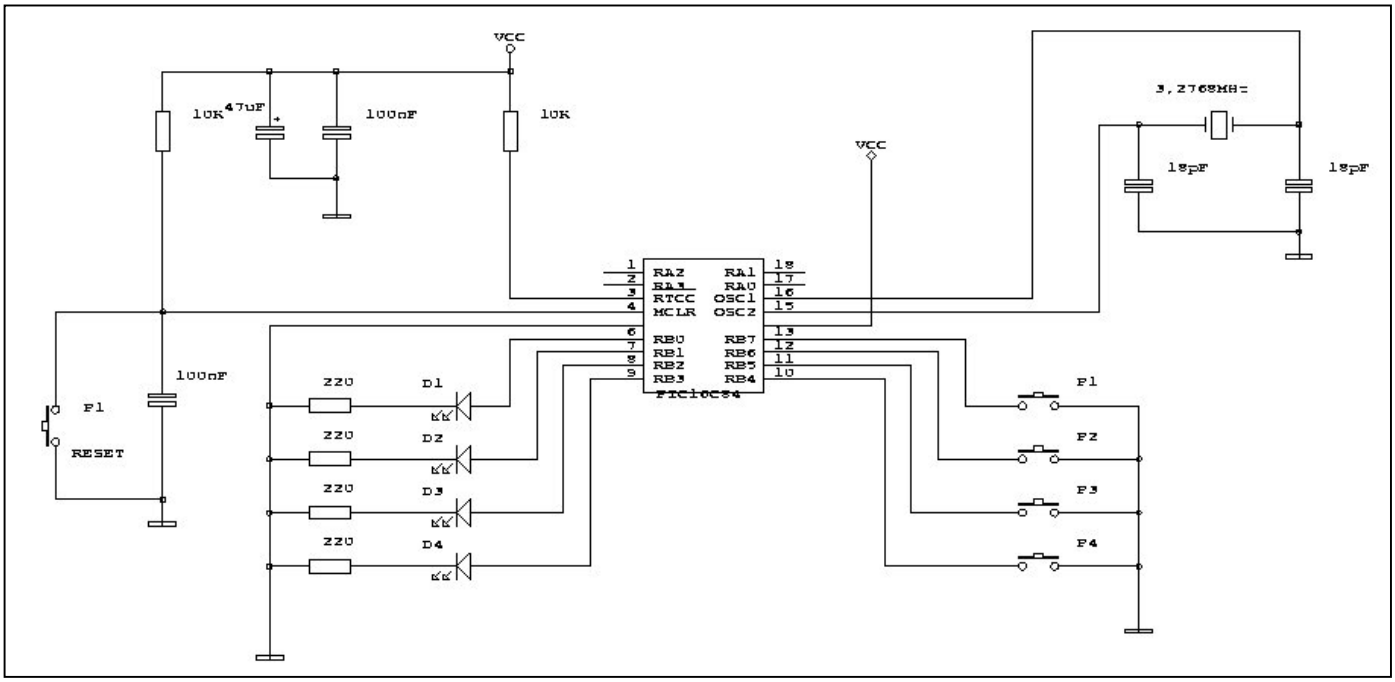


Figura 2. Schema elettrico per il test

```
 TITLE 'PROG10: Prova EEPROM interna'
 list F=INHX8M,P=16C84
;-----------------------------------------------------------------
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                 ; Buffer per pressione tasti
FR02    EQU 0DH                 ;
```

```
FR03   EQU 0EH                 ;
;---------------------------------------------------------------
#define P1 PORTB,7             ; Pulsante 1
#define P2 PORTB,6             ; Pulsante 2
#define P3 PORTB,5             ; Pulsante 3
#define P4 PORTB,4             ; Pulsante 4
;---------------------------------------------------------------
       ORG       0
       goto      START         ;Reset vector
       nop
       nop
       nop
       goto      INTERP        ;Interrupt vector

; ------------ subroutine lettura EEPROM ------------------------
LEGGI  clrf      EADR          ;
       bsf       STAT,5        ; Page 1
       bsf       ECON1,0       ; Leggi EEPROM byte indirizzo 00h
       bcf       STAT,5        ; Page 0
       nop                     ;
       movf      EDATA,0       ; Carica w con il dato letto
       movwf     PORTB         ; Copia stato led
       return                  ;
;--------- subroutine scrittura in EEPROM ----------------------
MEMO   clrf      EADR          ; Memorizza all'indirizzo 0
       movf      PORTB,0       ; Copia la porta B in W
       movwf     EDATA         ; Copia W in EDATA
       bsf       STAT,5        ; Page 1
       bsf       ECON1,2       ; WREN Abilita scrittura
       movlw     55H           ; Carica 55h in w
       movwf     ECON2         ; Copia W in ECON2
       movlw     0AAH          ; Carica aah in W
       movwf     ECON2         ; Copia W in ECON1
       bsf       ECON1,1       ; Abilita scrittura
LPW1   clrwdt                  ; Attendi fine scrittura
       btfss     ECON1,4       ;
       goto      LPW1          ;
       bcf       ECON1,4       ; Clear fine scrittura
       bcf       STAT,5        ; Pagina 0
RIL    call      DELA2M        ; Attesa rilascio pulsante
       btfss     P1            ;
       goto      RIL           ;
       btfss     P2            ;
       goto      RIL           ;
       btfss     P3            ;
       goto      RIL           ;
       btfss     P4            ;
       goto      RIL           ;
       goto      MAIN          ;
;---------------------- 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                  ;
```

```
        btfsc     P1            ; Testo P1
        bsf       FR01,7        ; Cancello memoria
        btfsc     P2            ; Testo P2
        bsf       FR01,6        ; Cancello memoria
        btfsc     P3            ; Testo P3
        bsf       FR01,5        ; Cancello memoria
        btfsc     P4            ; Testo P4
        bsf       FR01,4        ; Cancello memoria
        movf      RTCC,0        ; Controllo se finiti 20mS
        SKPZ                    ;
        goto      LPDEL         ;
        movlw     .128          ;
        movwf     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'       ;
        movwf     PORTA         ;
        call      LEGGI         ; Leggi ultimo stato memorizzato
;---------------------- main program ----------------------------
MAIN    call      DELA2M        ; Attesa 20mS + test pulsanti
        btfss     FR01,7        ; Testo se premuto P1
        goto      PULS1         ;
        btfss     FR01,6        ; Testo se premuto P2
        goto      PULS2         ;
        btfss     FR01,5        ; Testo se premuto P3
        goto      PULS3         ;
        btfss     FR01,4        ; Testo se premuto P4
        goto      PULS4         ;
        goto      MAIN          ;
PULS1   movlw     b'11110001'   ; Accensione led 1
        movwf     PORTB         ;
        goto      MEMO          ; Memorizza nuovo stato
PULS2   movlw     b'11110010'   ; Accensione led 2
        movwf     PORTB         ;
        goto      MEMO          ; Memorizza nuovo stato
PULS3   movlw     b'11110100'   ; Accensione led 3
        movwf     PORTB         ;
        goto      MEMO          ; Memorizza nuovo stato
PULS4   movlw     b'11111000'   ; Accensione led 4
        movwf     PORTB         ;
        goto      MEMO          ; Memorizza nuovo stato
;-----------------------------------------------------------------
INTERP                          ; Vettore interruzioni
        retfie                  ;
```

```
;-----------------------------------------------------------------
        END
```

Nelle definizioni sono stati inseriti i nomi dei registri dedicati e dei quattro pulsanti.
Il software è stato implementato in modo che premendo un pulsante, si accenda il led corrispondente e si spengano gli altri, ed in più venga memorizzato il nuovo stato delle uscite nella prima cella della EEPROM dati.
Se viene tolta e ridata alimentazione, il controller va a leggersi l'ultimo stato scritto nella EEPROM e riconfigura adeguatamente le uscite. Analizziamo ora le due routine di lettura e scrittura memoria, dato che l'interfacciamento con i pulsanti è già stato visto nelle precedenti applicazioni.

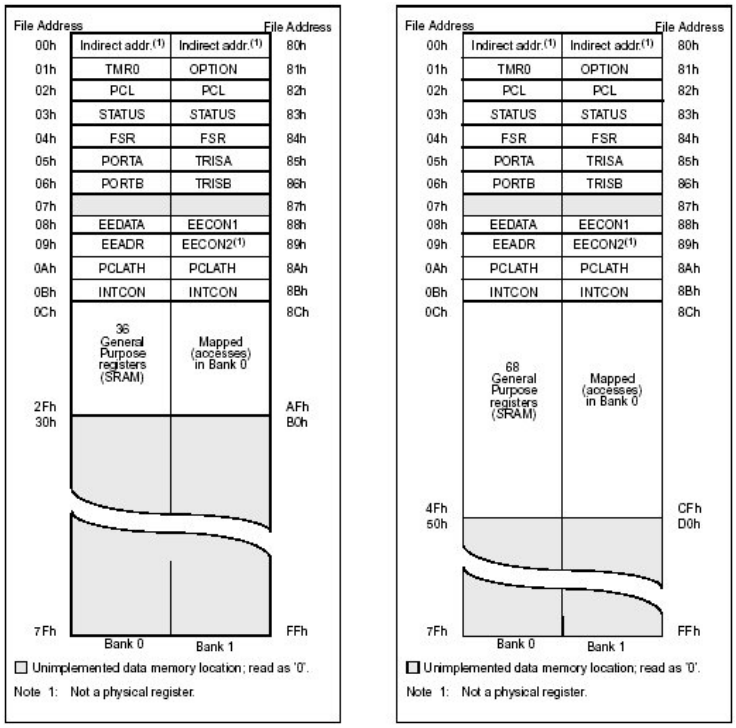


Figura 3. Mappa dei due banchi di registri

La subroutine LEGGI inizia con la selezione del secondo banco di registri, dove si trova il registro EECON1 ed azzera il bit 0 di quest'ultimo per avviare la fase di lettura. Dobbiamo premettere che in precedenza, nel registro EEADR deve essere stato caricato l'indirizzo della cella da leggere (nel nostro caso lo 00h). In figura 3 trovate l'indirizzo di tutti i registri del banco 0 e del banco 1.Poi si ripassa al primo banco di registri e si attende un ciclo. L'istruzione successiva, copia il valore trovato in EEDATA nel registro w (tale valore è proprio lo stato letto nella prima locazione di memoria).
Con l'ultima istruzione si copia lo stato da w alla porta B, poi si torna al programma principale.
Con queste poche istruzioni, abbiamo letto la cella di memoria di indirizzo 00h e ne abbiamo copiato il contenuto sulla porta B (chiaramente solo per quei pin settati come uscite).
La subroutine MEMO è invece leggermente più complessa, poichè sono necessari alcuni passi di programmazione senza i quali la scrittura non avviene correttamente.
Con la prima istruzione, carichiamo il registro EEADR con l'indirizzo della cella in cui scrivere, ovvero l'indirizzo 00h. Con le due successive invece copiamo lo stato della porta B nel registro EEDATA.
Ci sono poi una serie di istruzioni che servono a far partire il ciclo di scrittura e che terminano con il settaggio del bit 1 del registro EECON1.
Da qui si entra in un loop di attesa in cui si testa continuamente il bit 4 sempre di EECON1. Quando questo si setterà, allora il ciclo di scrittura sarà terminato e dovremo azzerare tale bit via software.
Le istruzioni seguenti invece, non fanno altro che attendere che tutti e quattro i pulsanti siano stati rilasciati.

Questo semplice programma, pur nella sua limitatezza applicativa, pone in risalto la metodologia per lavorare con la memoria EEPROM del PIC16C84.
-continua.