Esercitazioni PIC Midrange

Inserto_h: EEPROM

La memoria **EEPROM** (acronimo di Electrically Erasable PROM) è un'area in cui i dati possono essere scritti e letti da programma e che mantiene il suo contenuto anche se manca la tensione di alimentazione.
E' una caratteristica presente in un gran numero di PIC.

Da un punto di vista fisico, la EEPROM è ricavata da una porzione della Flash, quindi, rispetto alla RAM dati, richiede uno speciale algoritmo di accesso, per cui la lettura e la scrittura risultano più lente.
Inoltre, le ripetizioni degli accessi alla EEPROM in scrittura sono limitati: tipicamente, tra 100k e 1M; oltre, si riduce o annulla l'affidabilità della memoria (param. D120). Questo valore si riduce ulteriormente con l'aumento della temperatura oltre gli 85°C (param. D121).

Ne risulta che l'area EEPROM viene preferibilmente usata per salvare dati e variabili il cui richiamo o modifica avviene un numero limitato di volte durante l'esecuzione del programma e la vita del processore.

Vediamo come utilizzarla.

La EEPROM nei Midrange.

I vecchi PIC come 16F84A hanno solo 64bytes di EEPROM; però, tipicamente, i Midrange recenti dispongono di 128 o **256**bytes.
In effetti, dato che l'area è ricavata nella Flash, si tratta di 256 words a 14bit.

Questa area non appare direttamente accessibile in memoria, ma solo attraverso una serie di registri indiretti.

Rispetto ai Baseline, dove la EEPROM è trattata a blocchi, nei Midrange si può accedere alla singola locazione.
Un byte in scrittura cancella automaticamente la posizione e scrive i nuovi dati (cancella prima della scrittura). Il tempo di lettura, esaurita la sequenza di istruzioni necessaria, è immediato, mentre quello di scrittura è alquanto lento ed è controllato da un timer dedicato.
Per modificare il contenuto della memoria (scrittura/cancellazione) occorre una tensione superiore alla Vdd, che viene generata da una pompa di carica integrata.

E' possibile applicare una protezione alla scrittura dell'area EEPROM; in tal caso il programma può continuare a leggere o scrivere la memoria dati e quella programma, ma queste non sono leggibili dall'esterno.

Va osservato che i Midrange hanno caratteristiche diverse per la EEPROM, a seconda del modello. Possiamo dividerli in due gruppi:

- un primo gruppo consente l'accesso alla sola area EEPROM, come, ad esempio, 12F629/675. Questi chip hanno i registri:

  - **EECON1**
  - **EECON2** (non è un registro fisico)
  - **EEDATA**
  - **EEADR**

Qui i dati hanno un formato di 8bit (registro **EEDATA**).
Anche per gli indirizzi basta il registro **EEADR** a 8bit (2^{8}=256).

- altri chip dispongono della possibilità di scrivere/leggere anche l'area Flash della memoria programma, il che consente di implementare bootloader. Per questo sono dotati di più registri:

  - **EECON1**
  - **EECON2** (non è un registro fisico)
  - **EEDAT**
  - **EEDATH**
  - **EEADR**
  - **EEADRH**

Qui i registri sono doppi, una parte bassa (**EEDAT:H**) ed una alta (**EEADR:H**). I dati sono nella forma di word a 14bit e gli indirizzi, a 12bit per i chip con 4K di memoria programma e a 13bit per quelli con 8k, consentono di coprire tutta l'area Flash disponibile.

Questi chip, a loro volta si dividono in due gruppi:

- quelli in cui **la Flash (memoria programma) può essere letta**. In questi chip non sarà possibile scrivere "dati". (es. 16F684, 16F690)

- quelli in cui **la memoria programma può essere letta e scritta**.
  In questi ultimi saranno implementabili funzioni di bootloader o di auto aggiornamento del firmware (es. 16F884, 16F887)

Vediamo le funzioni di questi registri.

Come accennato, non è possibile raggiungere la EEPROM con le normali istruzioni, ma occorre utilizzare dei registri intermedi.
Questi sono **EEADR**, nei quali va scritto l'indirizzo relativo della locazione da raggiungere e **EEDAT** in cui in scrittura si inserisce il dato da scrivere e in lettura si recupera il dato contenuto nella locazione.
In sostanza, se attraverso una normale istruzione si fornisce l'indirizzo di una locazione di RAM da cui si legge o si scrive, per la EEPROM occorre scrivere l'indirizzo da raggiungere nel registro **EEADR** e leggere poi il contenuto della cella nel registro **EEDAT**.
Altrettanto, in scrittura occorrerà scrivere l'indirizzo in **EEADR** e il dato in **EEDAT**.
Un meccanismo interno provvede a spostare i dati da e verso la EEPROM.

Questa struttura è necessaria in quanto la EEPROM si trova nell'area della memoria programma, ovvero su un bus diverso da quello della RAM.
(ved. Architettura Harvard).

Il bus, inoltre, è diverso anche come ampiezza: il bus dati è a 8 bit e il bus istruzioni è, per i Midrange, a 14bit.

I registri di indirizzamento **EEDAT:H** , che sono trattabili con le istruzione ordinarie, sono a 8 bit, leggibili e scrivibili.

EEDAT – EEPROM DATA REGISTER (ADDRESS: 10Ch)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
EEDAT7	EEDAT6	EEDAT5	EEDAT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0
bit 7							bit 0

EEDATH contiene solo 6 bit validi. Questa soluzione dipende dal fatto che le locazioni di EEPROM e Flash sono ampie 14bit.

EEDATH – EEPROM DATA HIGH BYTE REGISTER(1) (ADDRESS: 10Eh)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0
bit 7							bit 0

In sostanza, il solo registro **EEDAT** viene usato per l'accesso alla EEPROM, trattata a 8 bit, mentre per la Flash, a 14bit, occorre la coppia **EEDAT:H** .

Altrettanto si può dire per la coppia **EEADR:H**
Il registro **EEADR** può indirizzare fino a un massimo di 256 indirizzi, il che permette di raggiungere tutta l'area EEPROM. Quando si seleziona una area EEPROM viene usato il solo **EEADR.**
Se il dispositivo contiene meno memoria, i bit più significativi del registro non sono implementati. Ad esempio, se il dispositivo dispone di 128 byte di EEPROM, il bit più significativo di EEADR non è implementato quando si accede alla EEPROM.

EEADR – EEPROM DATA REGISTER (ADDRESS: 10Dh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
EEDAR7	EEDAR6	EEDAR5	EEDAR4	EEDAR3	EEDAR2	EEDAR1	EEDAR0
bit 7							bit 0

EEADR, in combinazione col registro **EEADRH** permette di indirizzare l'area della memoria programma.
Quando si seleziona un valore di indirizzo della memoria programma, l'MSB dell'indirizzo è scritto nel registro **EEADRH** e la parte LSB è scritta nel registro **EEADR**.

EEADRH – EEPROM ADDRESS HIGH BYTE REGISTER(1) (ADDRESS: 10Fh)

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	EEADRH3	EEADRH2	EEADRH1	EEADRH0
bit 7							bit 0

EEADRH ha 4 bit implementati per i chip con 4k di memoria programma, per un totale di 12bit di indirizzamento (***2^12=4096***); i bit implementati diventano 5, per un totale di 13bit di indirizzamento per i chip con 8k di memoria programma (***2^13=8192***).

Differenza tra indirizzo relativo e indirizzo assoluto.

Occorre un chiarimento per evitare confusione tra **indirizzo relativo e assoluto quando si parla di EEPROM.**

Nella lettura o scrittura della EEPROM, gli indirizzi delle locazioni sono **relativi all'ampiezza dell'area**: se essa è da 256 elementi, gli indirizzi vanno da **00h** a **FFh**; se è da 128 elementi, gli indirizzi andranno da 00h a 7Fh; se da 64 locazioni, vanno da 00h a 3Fh.
Per questa ragione è sufficiente il registro di indirizzamento **EEADR** di 8bit di ampiezza.

In assoluto, la EEPROM ha, però, un posizionamento nella mappa di memoria generale. Tipicamente per i Midrange va da **2100h** a **21FFh**.

Questo valore è necessario se si utilizza una direttiva **de** (Declare EEPROM Data Bytes) per scrivere dati a 8bit in area EEPROM. Ad esempio:

```
; Inizializzazione della EEPROM
        org    0x2100

ch_tbl2  de "PICmicro"    ; scrive 8 bytes (caratteri ASCII)in EEPROM
                          ; a partire dall'indirizzo relativo 00h.
```

Oppure:

```
; Inizializzazione della EEPROM
        org    0x215F

; inizializza EEPROM con 12 caratteri ASCII più due valori numerici
; a partire dall'indirizzo relativo 5Fh.

   de "Program v1.0", 0, 0xA
```

Può essere una buona idea pre caricare, all'atto della programmazione del chip, alcuni valori in EEPROM a scopo di diagnosi o di sicurezza.

EEPROM e interrupt.

Se la lettura della EEPROM è soggetta solo alle necessarie istruzioni di accesso, la scrittura, invece, richiede un certo tempo, che, comparato con il ciclo istruzione, è molto grande (5-6ms - param. D122).

Questo vuol dire che, attivata la scrittura di una locazione EEPROM, questa si conclude solamente dopo 5-6ms, durante i quali non ne sono possibili altre.

Questa situazione può essere verificata in polling sul bit **WR** del registro **EECON1**, che, posto a 1 per avviare la scrittura, ritorna a 0 dopo che questa si è conclusa.

E' ovvio che in polling di questa durata può non essere accettabile per l'applicazione.

Per questo la scrittura della EEPROM è sorgente di interrupt, attraverso la solita coppia di bit di abilitazione e flag, che qui prendono il nome di **EEIE** e **EEIF** e si trovano rispettivamente nel registro **PIE2** e **PIR2**.

Il flag **EEIF** (**PIR2<4>**) viene settato quando la scrittura è completata. Deve essere azzerato dal programma.
Se è stato abilitato il bit **EEIE** (**PIE2<4>**) e se è abilitato l'interrupt (**EEIE, PEIE, GIE**), il completamento della scrittura genererà una interruzione, che può anche avere la funzione di wakeup da sleep.

La gestione a interrupt consente di effettuare scritture senza impegnare il processore in lunghi polling.

Lettura e scrittura della EEPROM: registro EECON1.

Le operazioni di accesso alla EPPROM/Flash sono gestite attraverso il registro **EECON1**:

EECON1 – EEPROM CONTROL REGISTER 1 (ADDRESS: 18Ch)

R/W-x	U-0	U-0	U-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	—	—	—	WRERR	WREN	WR	RD
bit 7							bit 0

- Il bit di controllo **EEPGD** determina se l'accesso sarà diretto alla memoria programma o alla EEPROM.
  Al reset è forzato a 0 e questo determina che tutte le operazioni saranno indirizzate alla EEPROM.

- I bit di controllo **RD** e **WR** servono ad inizializzare rispettivamente le operazioni di lettura o scrittura. Questi bit non possono essere azzerati, ma solo posti a 1 dal programma. Il loro ritorno a 0 indica la conclusione dell'operazione di lettura o scrittura.
  Questa caratteristica fa si che non sia possibile interrompere una operazione avviata prima della sua conclusione.
  Al reset sono azzerati per evitare l'avvio di una azione non desiderata.

- Il bit **WREN**, quando settato, consente una operazione di scrittura.
  Per default è azzerato e va attivato da programma. Questo impedisce una scrittura indesiderata.
  Da notare che **WR** e **WREN** hanno azioni diverse: **WR** avvia la scrittura, che, però, deve essere abilitata da **WREN**; un meccanismo di sicurezza in più contro le scritture non desiderate.

- Il bit **WRERR** viene settato quando è un'operazione di scrittura è stata interrotta da un reset MCLR o dal time out del WDT. In queste situazioni, la scrittura andrà ripetuta (i registri di indirizzo e dato non sono stati variati).

EECON2 non è un registro fisico; la sua lettura rende 0. Viene utilizzato esclusivamente nella scrittura della EEPROM per ricevere la sequenza-chiave di sicurezza.

Leggere la EEPROM.

Per leggere una posizione di memoria dati, l'utente deve:

- scrivere l'indirizzo da raggiungere nel registro **EEADR**
- azzerare **EEPGD**
- settare il bit di controllo **RD**

I dati sono disponibili nel prossimo ciclo, nel registro EEDAT che, quindi, può essere letto senza attese. EEDAT conterrà questo valore fino a quando non venga sostituito da una lettura (o scrittura) seguente.
In istruzioni:

```
banksel EEADR
movlw   address
movwf   EEADR
banksel EECON1
bcf     EECON1, EEPGD   ; memoria EEPROM dati
bsf     EECON1, RD      ; attiva lettura
banksel EEDAT
movf    EEDAT, W
```

Il valore contenuto nella locazione puntata è disponibile immediatamente dopo l'attivazione del bit **WR** e non richiede polling o interrupt.

Scrivere la EEPROM.

Per scrivere una locazione di dati EEPROM, l'utente deve:

- scrivere l'indirizzo da raggiungere nel registro EEADR
- scrivere il dati nel registro EEDAT
- impostare la **sequenza-chiave** per la scrittura.

La scrittura non verrà avviata se la sequenza-chiave di cui sopra non è eseguita esattamente:

- scrivere **55h** in **EECON2**
- scrivere **AAh** in **EECON2**
- settare il bit **WR** per ogni byte.

Va osservato che l' interrupt dovrebbe essere disabilitato durante questo segmento di codice, dato che la sequenza-chiave deve essere eseguita senza pause.
Inoltre, il bit **WREN** in **EECON1** deve essere settato per abilitare la scrittura.
Questo meccanismo impedisce la scrittura accidentale di dati in EEPROM a causa di errori

inaspettati.

Per la massima sicurezza, l'utente dovrebbe mantenere a 0 il bit **WREN**, tranne quando si deve aggiornare la EEPROM. Il bit **WREN** non viene cancellato in hardware, ma solo da programma o da un reset.
Dopo l'avvio di una sequenza di scrittura, la cancellazione del bit **WREN** non influenza questo ciclo di scrittura. Il bit **WR** non potrà essere settato fino a che il bit **WREN** non sia settato.

Al completamento del ciclo di scrittura, il bit **WR** è cancellato dall'hardware e viene settato il flag di interrupt (**EEIF**). L'utente può utilizzare l'interrupt abilitando **EEIE** e **GIE/PEIE** oppure usare il flag in polling. Va ricordato che **EEIF** deve essere cancellato dal software.

In istruzioni:

```
banksel EEADR
movlw   address
movwf   EEADR         ; indirizzo in cui scrivere
movlw   data
movwf   EEDAT         ; dato da scrivere
banksel EECON1
bcf     EECON1, EEPGD ; alla EEPROM
bsf     EECON1, WREN  ; abilita scrittura
bcf     INTCON, GIE   ; disabilita tutti gli interrupt
movlw   55h           ; sequenza-chiave   -------------
movwf   EECON2        ;                               |
movlw   AAh           ; NON ALTERARE QUESTA SEQUENZA  |
movwf   EECON2        ;                               |
bsf     EECON1, WR    ; avvia scrittura   -------------
bsf     INTCON, GIE   ; riabilita interrupt
btfsc    EECON1,WR    ; polling attesa fine scrittura
 goto    $-1
bcf     EECON1, WREN  ; disabilita scrittura
```

Attenzione: La sequenza di scrittura evcidenziata NON deve essere alterata, né nella sequenza, né nella quantità di istruzioni: il suo scopo è quello di creare una chiave ad alta sicurezza che impedisce la scrittura indesiderata della EEPROM

Questa sequenza e il bit **WREN**, che è azzerato al reset, impediscono una scrittura accidentale durante condizioni anomale come Brown-out, Power Glitch o malfunzionamenti del software.

Il tempo di scrittura è dell'ordine dei 4-6ms (param. D122). Per questa ragione è presente la possibilità di una gestione in interrupt in modo tale da lasciare libero il processore per altre attività durante il tempo di scrittura.

Si potrà usare un polling sul flag **EEIF** invece di **WR**:

```
banksel  PIR2
bcf      PIR2, EEIF    ; cancella flag
......
```

```
bsf     INTCON, GIE   ; riabilita interrupt
banksel PIR2
btcfsc  PIR2, EEIF    ; attesa fine scrittura
 goto   $-1
bcf     PIR2, EEIF    ; cancella flag
banksel EECON1
bcf     EECON1, WREN  ; disabilita scrittura
```

In tutti i casi, **la routine, come scritta sopra, è bloccante**, ovvero ritorna dalla chiamata solo al termine della scrittura. Se questo non è adeguato all'applicazione, si possono usare altre soluzioni.

Con attesa in sleep.

E' possibile mandare il processore in sleep e farlo risvegliare dall'interrupt della EEPROM (flag **EEIF**):

```
banksel  PIR2
bcf      PIR2, EEIF    ; cancella flag
......
bsf     INTCON, GIE   ; riabilita interrupt
sleep                 ; attesa fine scrittura in sleep
nop
bcf     EECON1, WREN  ; disabilita scrittura
banksel PIR2
bcf     PIR2, EEIF    ; cancella flag
```

Questa tecnica potrà essere usata vantaggiosamente quando si implementa una o più scritture in un momento in cui il processore non deve svolgere altre attività, con una contemporanea riduzione del consumo di corrente.

Con fine scrittura in interrupt.

Si potrà attivare l'interrupt della EEPROM (bit **EEIE, PEIE, GIE**) e gestire l'evento dal vettore di interruzione. Questa tecnica è utile se il processore deve svolgere altre azioni critiche durante il tempo della scrittura.

Con tempo di scrittura esterno.

Ancora, è possibile scrivere, ma non attendere il completamento della scrittura.
Questa tecnica è utile se si scrive una sola locazione e prima della prossima scrittura trascorre un tempo sufficiente (min. 6ms) al completamento della precedente. In questo tempo il programma potrà svolgere altre attività.
Se, però, vanno ripetute più scritture consecutive, è opportuno scegliere una delle soluzioni viste prima, in quanto è possibile che la scrittura venga completata in un tempo minore del massimo indicato sul foglio dati; in tal caso è inutile inserire attese fisse quando l'analisi del bit **WR** o del flag **EEIF** consentono una gestione più efficace.

Una diversa soluzione per questo ultimo caso consiste nell'inserire il test di fine scrittura non al temine delle routine corrente, ma all'inizio di una successiva scrittura.
Se testiamo il flag di fine scrittura al termine della routine di scrittura, abbiamo tutto il tempo di attesa; se lasciamo la routine di scrittura completata e testiamo il flag prima della prossima scrittura, è probabile che sia trascorso tutto (o una parte) del tempo necessario alla scrittura e l'attesa sia ridotta o nulla:

```
       btfsc    EECON1,WR    ; polling attesa fine scrittura
        goto    $-1
       banksel EEADR
       movlw   address
       movwf   EEADR         ; indirizzo in cui scrivere
       .....
```

Nel caso in cui l'operazione di scrittura o lettura sia eseguita nel programma una volta o un numero di volte ridotto, si potranno scrive anche macro al posto di subroutine.

Lettura della memoria programma.

Come per la EEPROM, in alcuni chip anche la memoria programma può essere letta una locazione alla volta. L'accesso avviene con una sequenza analoga a quella necessaria per la EEPROM.

Va notato che, mentre il dato della EEPROM è disponibile nel ciclo successivo all'attivazione del bit di lettura **RD**, per la Flash l'operazione richiede due cicli, quindi è leggermente più lenta.

Per leggere una posizione di memoria del programma, l'utente deve:

- scrivere due byte dell'indirizzo **EEADR** e **EEADRH**
- settare il bit di controllo **EEPGD**
- settare il bit di avvio della lettura **RD**

Da osservare che occorre l'indirizzo completo, a 12bit (fino a 4k) e a 13bit (fino a 8k), che va scritto nella coppia di registri **EEADR** e **EEADRH**.

Una volta settato il bit di avvio della lettura, il controller Flash utilizzerà il secondo ciclo di istruzione per leggere i dati.
Ciò provoca che la seconda istruzione subito dopo la riga **bsf EECON1,RD** è da ignorare.
I dati sono disponibili nel ciclo successivo nel registro **EEDAT** e **EEDATH** e può essere letto come due byte dalle istruzioni seguenti.

```
      banksel EEADR
      movlw   addrh
      movwf   EEADRH        ; MSB indirizzo
      movlw   addrl
      movwf   EEADR         ; LSB indirizzo
      banksel EEACON1
      bsf     EECON1, EEPGD ; puntare alla memoria programma
      bsf     EECON1, RD    ; avvia lettura
; la memoria programma è letta nel secondo ciclo
      NOP
      NOP
      banksel EEDAT
; il dato è disponibile in EEDAT:H
```

Le due istruzioni **nop** impediscono all'utente di inserire una istruzione a due cicli dopo che il bit **RD** è stato settato.

Se per errore il bit **WR** è settato quando **EEPGD=1**, esso verrà immediatamente resettato e non si svolgerà l'operazione.

Rispetto alla lettura della EEPROM, quella della memoria programma, quindi, è leggermente più lenta.

Va notato che le operazioni di lettura, sia della EEPROM che della Flash non influiscono sulla sua durata. Sono solo le scritture a determinare un limite massimo di uso.

Scrittura delle memoria programma.

Per i chip che consentono la scrittura della memoria programma, questa azione è utile per implementare bootloader o per introdurre meccanismi di upgrade del firmware da remoto.

Attenzione: la scrittura della memoria programma durante l'esecuzione del programma, nei chip dove questo è possibile, è una azione rischiosa e va applicata solamente dove è ben chiaro cosa si sta facendo e gli algoritmi usati per la scrittura siano corretti e collaudati.

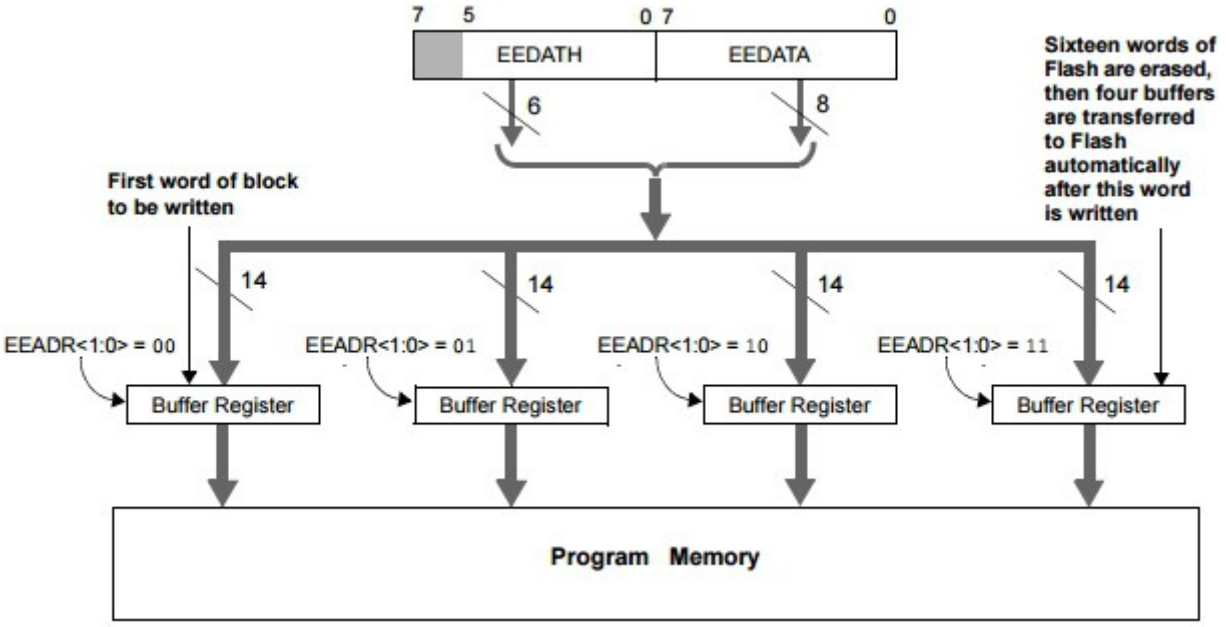
A differenza della EEPROM a cui si può accedere a livello di byte, per la scrittura della memoria programma si deve operare a blocchi.

La memoria programma Flash può essere scritta solo se l'indirizzo di destinazione è in un segmento di memoria che non è protetto da scrittura, come definito nei bit **WRT<1:0>** della ***Configuration Word***.

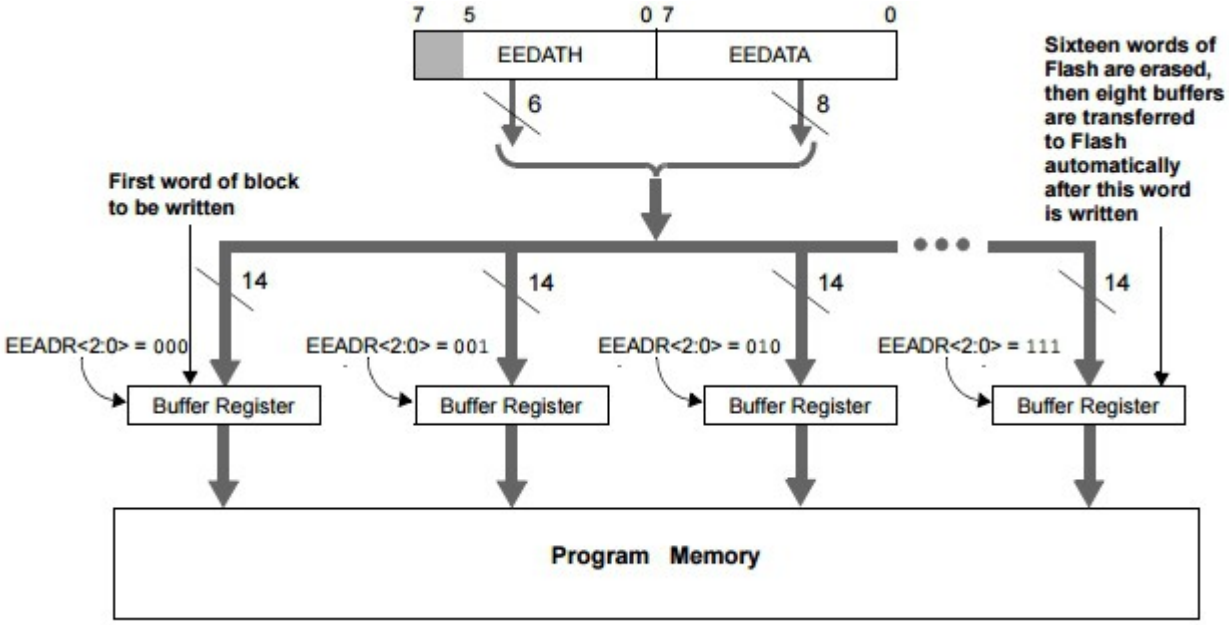
La memoria programma Flash deve essere scritta in blocchi di quattro parole per i chip con memoria di 4k.

Facciamo riferimento a “parole” (words) e non a bytes in quanto gli elementi della memoria programma sono a 14bit; questo richiede la presenza dei due registri **EEDATA** e **EEDATH**.

Un blocco consta di otto parole con indirizzi sequenziali, con un limite inferiore definito da un indirizzo, con **EEADR<2:0>=000**e terminante con **EEADR<2:0>=111**.
La scrittura del blocco viene eseguita previa cancellazione. L'operazione di scrittura è allineata ai limiti del blocco e non può verificarsi a cavallo di due blocchi.



E in blocchi di otto parole per i chip con 8k di Flash



Per scrivere, l'indirizzo di partenza deve prima essere caricato nei buffer (vedere le figure precedenti). Questo è effettuato scrivendo prima l'indirizzo di destinazione in **EEADR** e **EEADRH** e quindi scrivendo i dati a **EEDATA** e **EEDATH**.
Dopo che l'indirizzo e i dati sono stati impostati,, si può eseguire la sequenza di scrittura:

- settare il bit di controllo **EEPGD** di **EECON1**per puntare la memoria Flash
- scrivere la sequenza-chiave **55h**, poi **AAh**, in **EECON2**
- settare il bit di controllo **WR** del registro **EECON1**per avviare la scrittura

Devono essere scritte tutte le otto posizioni del registro del buffer con i dati voluti.

Per trasferire i dati dai registri di buffer alla memoria programma, **EEADR** e **EEADRH** devono indicare l'ultima posizione nel blocco a otto parole (**EEADR <2: 0> = 111**). Quindi deve essere eseguita la sequenza precedente.

L'utente deve seguire la stessa sequenza specifica per avviare la scrittura per ogni parola nel blocco di programma, scrivendo ogni parola di programma in sequenza (**000, 001, 010, 011, 100, 101, 110, 111**). Quando la scrittura è eseguita sull'ultima parola (**EEADR<2:0>=111**), il blocco di sedici parole viene automaticamente cancellato e vengono scritti i contenuti degli otto registri di buffer.

Dopo l'istruzione **bsf EECON1,WR**, il processore richiede due cicli per impostare l'operazione di cancellazione / scrittura. L'utente deve inserire due istruzioni **nop** dopo che il bit **WR** è stato settato.

Poiché i dati vengono scritti nei registri di buffer, la scrittura delle prime sette parole del blocco appare verificarsi immediatamente, però **il processore si arresta internamente per i tipici 4-5ms necessari durante il ciclo in cui avviene la cancellazione** (cioè, l'ultima parola del blocco).

Questa situazione non è una condizione di sleep: i clock e le periferiche continueranno a funzionare normalmente; solo il flusso delle istruzioni è arrestato per il tempo richiesto dalla scrittura.
Dopo il ciclo di scrittura delle otto parole, il processore riprenderà il funzionamento normale con la terza istruzione seguente la scrittura di **EECON1**.

I fogli dati indicano un esempio della sequenza completa di scrittura a otto parole. L'indirizzo iniziale viene caricato nella coppia di registri **EADR** e **EEADRH**; i dati delle otto parole vengono caricati tramite indirizzi indiretti.

```
;-----------------------------------------------------------
; Si assumono i seguenti presupposti:
; Un indirizzo di partenza valido (bit meno significativi = '000')
; è caricato nel buffer ADDRH:ADDRL
; ADDRH, ADDRL e DATADDR sono in memoria RAM dati
      banksel EEADRH
      movf    ADDRH,W      ; Load initial address
      movwf   EEADRH
      movf    ADDRL,W
      movwf   EEADR
      movf    DATAADDR,W   ; Load initial data address
      movwf   FSR
LOOP  movf    INDF,W       ; Load first data byte into lower
      movwf   EEDATA
      incf    FSR,F        ; Next byte
      movf    INDF,W       ; Load second data byte into upper
      movwf   EEDATH
      incf    FSR,F
      banksel EECON1
      bsf     EECON1,EEPGD ; program memory
      bsf     EECON1,WREN  ; abilita scrittura
      bcf     INTCON,GIE   ; disabilita tuti gli interrupt
      btfsc   INTCON,GIE   ; secondo AN576 - Per 16/17Cxx
       goto   $-2          ; non necessario chip meno datati.
;-----------------------------------------------------------------
; sequenza-chiave
```

```
      movlw   55h
      movwf   EECON2
      movlw   AAh
      movwf   EECON2
      bsf     EECON1,WR   ; attiva scrittura
       nop
       nop
;-----------------------------------------------------------------
      bcf      EECON1,WREN ; Disable writes
      bsf      INTCON,GIE  ; Enable interrupts (comment out if not
                           ; using interrupts)
      banksel  EEADR
      movf     EEADR,W
      incf     EEADR,F     ; Increment address
      andlw    0x0F        ; Indicates when sixteen words have been
                           ; programmed
      sublw    0x0F        ; 0x0F = 16 words
                           ; 0x0B = 12 words (PIC16F884/883/882 only)
                           ; 0x07 = 8 words
                           ; 0x03 = 4 words (PIC16F884/883/882 only)
      skpz                 ; Exit on a match,
       goto    LOOP        ; Continue if more data needs to be written
```

Avvertenza: per alcuni Midrange obsoleti, non considerati in questo corso, ad esempio 16F876/77, l'accesso alla memoria programma è leggermente diverso. Verificare sui fogli dati le modalità di scrittura da applicare.

Verifica della scrittura.

A seconda dell'applicazione, buona programmazione segue la pratica di verificare quanto scritto in EEPROM; questo evita possibili errori hardware, ad esempio un reset inaspettato o una perdita di sicurezza dovuta ad un elevato numero di cicli di scrittura.

```
      banksel EECON1
      bsf     EECON1, RD
      banksel EEDAT
      movf    EEDAT,W
      banksel EEdata
      xorwf   EEdata,w    ; compara con xor
      skpz
       goto   EEWRITE_ERR ; n – gestione errore
      ; s – continua
```

Una routine che scrive e verifica:

```
EEwrtver:             ; scrivi una locazione da EEPROM e verifica
      call    EEwrite
      bsf     EECON1, RD
      banksel EEDAT
      movf    EEDAT,w
      banksel EEdata
      xorwf   EEdata, W    ; compara con xor
      skpz
       goto   EEWRITE_ERR  ; n - gestione errore
; s – continua
```

Soluzioni pratiche.

Per rendere efficace la programmazione in Assembly, in generale è opportuno scrivere macro o subroutines che svolgano le azioni richieste dalle operazioni di scrittura e lettura della EEPROM e della Flash.

Questo isola il programmatore dalla necessità di riscrivere ogni volta le sequenze necessarie.
Ad esempio, sotto forma di subroutine:

```
EEread:                 ; leggi una locazione da EEPROM
; indirizzo da leggere in W
        banksel EEADR
        movwf   EEADR
        banksel EECON1
        bcf     EECON1, 7       ; memoria EEPROM dati
        bsf     EECON1, RD      ; attiva lettura
        banksel EEDAT
        movf    EEDAT, W         ; dato in W
        return

EEwrite:              ; scrivi una locazione da EEPROM
; indirizzo in W, dato da scrivere in EEdata
        banksel EEADR
        movwf   EEADR          ; indirizzo in cui scrivere
        movlw   EEdata
        movwf   EEDAT          ; dato da scrivere
        banksel EECON1
        bcf     EECON1, 7      ; EEPGD indirizza alla EEPROM
        bsf     EECON1, WREN   ; abilita scrittura
        bcf     INTCON, GIE    ; disabilita tutti gli interrupt
        movlw   55h            ; sequenza-chiave   -------------
        movwf   EECON2         ;                                |
        movlw   AAh            ; NON ALTERARE QUESTA SEQUENZA   |
        movwf   EECON2         ;                                |
        bsf     EECON1, WR     ; avvia scrittura   -------------
        bsf     INTCON, GIE    ; riabilita interrupt
        btfsc   EECON1,WR
         goto   $-1
        bcf     EECON1, WREN   ; disabilita scrittura
        retlw   0
```

Attenzione: molto importante!

La disabilitazione e sopratutto la riabilitazione di `GIE` va inserita solo se il programma fa uso dell'interrupt.

L'uso di `$-1` è ammissibile in quanto la routine è indirizzata all'uso con una famiglia di processori in cui la codifica dell'istruzione occupa una sola locazione.

La routine è stata scritta con una piccola variazione rispetto agli esempi riportati sui fogli dati, ovvero la sostituzione di **`bcf EECON1,EEPGD`** con **`bcf EECON1,7`** ; questo consente di usare la routine per tutti i chip Midrange che hanno l'accesso alla Flash (es. 16F690) oppure che hanno accesso solo alla EEPROM (es. 12F629).
Dove non c'è accesso alla Flash non è necessario uno switch tra Flash e EEPROM e il bit **`EEPGD`** non è implementato. Questo fa si che non compaia nella lista delle label *nomeprocessore.inc*: in fase di compilazione si genera un errore.
In ogni caso, nei Midrange il bit **`EEPGD`** si trova sempre nella posizione **`EECON1,7`**: se il bit è implementato, la sua scrittura a 0 indirizza l'operazione alla EEPROM e non alla Flash.
Se il bit non è implementato perchè non c'è possibilità di accedere alla Flash, la sua scrittura non avrà altro effetto che perdere un ciclo di istruzione. Quindi, la sostituzione è ammissibile.

Una soluzione molto più elegante è quella di verificare se la label è definita: non lo sarà per processori che non hanno accesso alla Flash e la riga tra **`#ifdef`** e **`#endif`** non sarà eseguita.:

```
  banksel EECON1
#ifdef    EEPGD
  bcf     EECON1, EEPGD   ; memoria EEPROM dati
#endif
  bsf     EECON1, RD      ; attiva lettura
```

Le subroutine vanno compilate per i singoli processori in quanto la disposizione dei registri di controllo della EEPROM è la più varia, a seconda del numero di banchi e del tipo di chip.

Riepilogo dei registri usati per l'accesso alla EEPROM.

Questi sono i registri interessati alle operazioni su EEPROM e Flash (dal foglio dati di 16F690):

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
0Bh/8Bh/ 10Bh/18Bh	INTCON	GIE	PEIE	T0IE	INTE	RABIE	T0IF	INTF	RABIF	0000 000x	0000 000x
0Dh	PIR2	OSFIF	C2IF	C1IF	EEIF	—	—	—	—	0000 ----	0000 ----
8Dh	PIE2	OSFIE	C2IE	C1IE	EEIE	—	—	—	—	0000 ----	0000 ----
10Eh	EEDATH(1)	—	—	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0	--00 0000	--00 0000
10Fh	EEADRH(1)	—	—	—	—	EEADRH3	EEADRH2	EEADRH1	EEADRH0	---- 0000	---- 0000
10Ch	EEDAT	EEDAT7	EEDAT6	EEDAT5	EEDAT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0	0000 0000	0000 0000
10Dh	EEADR	EEADR7	EEADR6	EEADR5	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0	0000 0000	0000 0000
18Ch	EECON1	EEPGD	—	—	—	WRERR	WREN	WR	RD	x--- x000	0--- q000
18Dh	EECON2	EEPROM Control Register 2 (not a physical register)								---- ----	---- ----

Da osservare (1) che **`EEDATH`** e **`EEADRH`** sono disponibili solo per i chip che hanno l'accesso alla Flash.
La tabella indica anche la situazione dei bit nei registri a seguito di reset per **POR/BOR** (caduta della tensione sotto i minimi ammissibili) o per altri reset che non comportano direttamente la mancanza di tensione (**WDT, MCLR**).

Possiamo ancora aggiungere che, trattando i registri di gestione della EEPROM, è necessario un uso intensivo del **`banksel`**, in quanto la posizione dei registri nei banchi può essere diversa da un chip all'altro.
Ad esempio, 16F690 e simili, con RAM su 4 banchi, i registri si trovano in banco 2 e 3:

EEDAT	10Ch	EECON1	18Ch
EEADR	10Dh	EECON2(1)	18Dh
EEDATH	10Eh		18Eh
EEADRH	10Fh		18Fh

In 12F629/675 e simili, con RAM su due banchi, sono concentrati in banco 1

	1Ah	EEDATA	9Ah
	1Bh	EEADR	9Bh
	1Ch	EECON1	9Ch
	1Dh	EECON2(1)	9Dh

Alcuni consigli per l'uso della EEPROM integrata

- E' necessario verificare che la Vdd resti entro i limiti nominali per tutto il tempo di durata della scrittura, dato che un eccessivo abbassamento della tensione potrebbe danneggiare il processo di scrittura, in quanto la pompa di carico interna non potrebbe fornire la tensione di programmazione necessaria.
  L'impiego del Brown Out Detector o di altri sistemi consente di evitare questo rischio, resettando il micro; l'errore potrà essere rilevato con la verifica della scrittura, con una adeguata gestione sia del reset che della scrittura della EEPROM.

- Occorre evitare di avere una interruzione durante la fase di scrittura della sequenza-chiave. Se avete interrupt attivi, ricordate, come negli esempi precedenti, di disabilitarli durante questa fase.

- Se la EEPROM è scritta un numero di volte elevato, la verifica dei dati dopo la scrittura è altamente consigliabile. In particolare, per migliorare la durata della EEPROM, mantenete bassa la temperatura del chip.

- L'area EEPROM è molto adatta per il salvataggio delle impostazioni utente. Se si cambia qualcosa, al riavvio del sistema le impostazioni sono sempre presenti.
  Se, però, si ha a che fare con applicazioni time-sensitive, l'attesa di 5ms per scrittura può non essere accettabile. Prima di utilizzare la EEPROM integrata è necessario valutare con attenzione questo limite di tempo.

- Altrettanto per le applicazioni in cui le scritture sono in numero elevato: se nella vita del dispositivo si prevede di superare il numero di scritture ammesse, è opportuno cercare altre soluzioni o implementare verifiche della tenuta della scrittura, con indicazione delle situazioni di errore.

La scrittura della EEPROM e della Flash.

Le criticità nell'uso della EEPROM, e ancor più della Flash, sono essenzialmente tre:

1. il tempo necessario per la scrittura, che, rispetto al ciclo istruzione, è sensibile
2. il limitato numero di scritture possibili prima del decadimento della affidabilità
3. l'obbligo di non avere reset che interrompano la scrittura, pena la mancata riuscita dell'operazione

Il primo problema richiede che sia valutato il tempo di scrittura. Se la lettura è abbastanza immediata, la scrittura richiede un tempo medio di 4ms, con un massimo di 6ms. Questo vuol dire 4000-6000us. A 4MHz di clock, il ciclo di una istruzione è 1us: nel tempo necessario per scrivere una locazione EEPROM o Flash si possono eseguire 4000-6000 istruzioni.
E' evidente che il programmatore debba studiare l'introduzione dei cicli di scrittura in modo tale da non influire sul resto dell'applicazione, cosa che, in generale, è possibile in tutte le applicazioni non critiche.

Il secondo problema richiede che la scrittura della EEPROM (e ancor più della Flash) sia da considerarsi ben diversamente dalla scrittura della RAM dati.
La EEPROM non può essere l'area dati di un sistema di data logging che richiede continui aggiornamenti; dovrà essere ben valutata la richiesta di scritture rispetto alla vita attesa del microcontroller ed eventualmente messe in atto strategie di verifica e uso di locazioni altrenative. Anche questo è possibile con un o sforzo ragionevole.

Il terzo problema è quello veramente grave: se, durante la scrittura di una locazione, cade la tensione di alimentazione o si verifica un reset, la scrittura quasi certamente è abortita.
La situazione si può monitorare con il bit WERR, ma questo non risolve il problema.
Se stiamo scrivendo una sola cella, è ragionevolmente facile ripristinare una nuova scrittura, ma se stiamo usando, come normale, più celle, o molte celle, risulta difficile, se non impossibile, sapere a che punto l'operazione di scrittura è andata a monte. Per quanto si possano implementare algoritmi sofisticati, la situazione è sostanzialmente la stessa: se la scrittura di un blocco di EEPROM è fallita, occorre ripetere completamente le operazioni. E se si tratta di dati accumulati sequenzialmente, certamente non è possibile recuperare la situazione precedente.

Ne deriva che **la scrittura di EEPROM**, **e ancor più della Flash programma, richiede**

- **l'implementazione di sistemi di verifica della tensione di alimentazione che non deve cadere sotto il minimo durante l'operazione**.
- L'impossibilità di verificarsi di eventi di reset incontrollati

Se la tensione di alimentazione cade sotto un certo minimo (tipicaente qualche decimo di volt sotto la minima tensione Vdd ammessa), la pompa di carico interna non fornisce la sufficiente tensione per la cancellazione/scrittura della cella EEPROM/Flash, che potrà essere malamente scritta e contenere un valore errato.

Occorre, quindi, tenere sotto controllo la Vdd, sia con le risorse interne (comparatori, ADC), sia con voltage detector esterni, in modo tale da impedire la scrittura se la tensione è in caduta.
In pratica, è consigliabile implementare una sorveglianza della tensione di rete a monte dello stabilizzatore che alimenta il sistema; questo potrà prevedere una caduta prossima della Vdd. Un interrupt permetterà di comunicare al microcontroller la situazione.

E' anche possibile che si utilizzi il tempo tra l'avviso della prossima caduta della rete e la reale caduta della Vdd per mettere in salvo dati. In questo caso, la presenza di adeguati elettrolitici sarà utile per sostenere la scrittura in corso.

Agire per scrivere EEPROM e Flash è una operazione che va fatta solamente se si è garantita una Vdd sufficiente e l'assenza di reset. Questo può richiedere modifiche anche non semplici tanto all'hardware che al software.
Se non si è assicurata questa situazione, è da evitare la scrittura di EEPROM e Flash se non sotto la supervisione dell'utente: in un sistema automatico avremmo solamente l'indicazione che la scrittura è fallita e probabilmente nessun modo per recuperare la situazione se non far ripartire il programma da zero.

Per quanto riguarda i reset, sappiamo che possono avere varie sorgenti. Reset “soft” come quello prodotto dall'overflow del watchdog timer (WDT) non dipendono dalla tensione di alimentazione, ma, in ogni caso, effetuano una ripartenza del programma dal vettore di reset, il che richiede nel programma una gestione dell'evento.

Nel caso di reset “hard”, si tratta di azioni dipendenti dall'abbassamento della tensione di alimentazione, come il Power On Reset (POR) o il reset creato da Brown Out Detector (BOR).
In questi casi, occorre identificare la causa del reset ed agire di conseguenza.
E' , ovviamente, da escludere, che si utilizzi il reset MCLR durante la scrittura della EPROM o della Flash. Peraltro, questo segnale, nei PIC non obsoleti, può essere perfino disabilitato dalla configurazione.

Se non si è assicurata questa situazione ed è, invece, **presente il rischio di una Vdd bassa o di un reset inaspettato, è opportuno evitare di accedere in scrittura a EEPROM e Flash**.

HEF.

Gli ***Enhanced Midrange*** ed alcuni chip classificati come Midrange (16F673, 16F72x, 16F88x) dispongono, al posto della EEPROM, di una area di memoria denominata **HEF**, acronimo di High Endurance Flash, ovvero Flash a Lunga Durata.

Tuttavia, HEF ha caratteristiche un poco diverse dalla EEPROM.

Qui alcune pagine sull'argomento.

Alcuni chip (ad es. 16F72x) non dispongono di EEPROM, ma hanno la possibilità di leggere la memoria programma.
Per questo sono implementati i seguenti registri:

- **PMCON1**
- **PMDATL**
- **PMDATH**
- **PMADRL**
- **PMADRH**

Il valore scritto in **PMADRH:L** determina quale posizione di memoria di programma viene letta.
L'operazione di lettura verrà avviata impostando il bit **RD** del registro **PMCON1**.

Il controller di memoria del programma richiede due istruzioni per completare la lettura, cosicchè la seconda istruzione dopo il set del bit **RD** verrà ignorata.
Per evitare conflitti con l'esecuzione del programma, è consigliabile inserire due nop.
Quando la lettura viene completata, il risultato viene inserito nella coppia **PMDATH:L.**

Ecco un esempio di riferimento:

```
banksel  PMADRL
movf     MS_PROG_ADDR, W
movwf    PMADRH           ;MS Byte of Program Address to read
movf     LS_PROG_ADDR, W
movwf    PMADRL           ;LS Byte of Program Address to read
banksel  PMCON1
bsf      PMCON1, RD       ;Initiate Read
nop
nop
banksel  PMDATL
movf     PMDATL, W        ;W = LS Byte of Program Memory Read
movwf    LOWPMBYTE
movf     PMDATH, W        ;W = MS Byte of Program Memory Read
movwf    HIGHPMBYTE
```

Altri chip, in bilico tra i Midrange e gli Enhanced Midrange, come 12F32x e 12LF1522 dispongono della possibilità di scrivere la Flash come HEF.
Si rimanda ai relativi fogli dati per le caratteristiche e gli esempi di programmazione.

Altre risorse.

- Microchip AN601 EEPROM Endurance Tutorial
- Microchip AN1019 EEPROM Endurance Tutorial
- Microchip AN602 How to get 10 Million Cycles out of your Microchip Serial EEPROM
- Microchip AN536 Basic Serial EEPROM Operation
- Microchip AN537 Everything a System Engineer needs to know about Serial EEPROM
- Microchip AN562 Using the Microchip Endurance Predictive Software
- Microchip AN576 Techniques to Disable Global Interrupts
- Microchip AN851 A FLASH Bootloader for PIC16 and PIC18 Devices
- Microchip AN1673 Using the PIC16F1XXX High-Endurance Flash (HEF) Block
- Microchip 8-bit Bootloader
- Open Source PIC Bootloaders