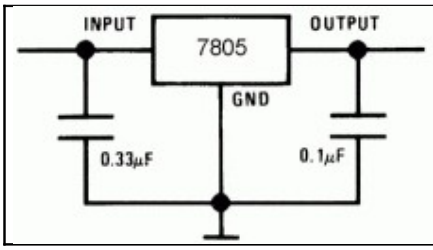
Esercitazioni PIC Baseline

17A - Vdd

Lo scopo di queste pagine è fornire alcune informazioni sulla tensione di alimentazione dei PIC Baseline che sono solitamente trascurate:

La Vdd

Questo termine indica la tensione di alimentazione del chip, che per i PIC Baseline può andare da 2V a 5.5V.
Questi sono limiti operativi (foglio dati, param. D001): il valore massimo non deve essere superato, pena il danneggiamento irreversibile del componente ed non si deve scendere al di sotto del valore minimo, pena un funzionamento irregolare.
In effetti, la Vdd può andare da 0 a 6.5V (**valori assoluti**!), ma al di sopra dei 6Vsi ha la certezza della rapida distruzione del chip.
Per contro, al di sotto dei 2V non ci sono problemi di sicurezza, ma viene messa in crisi la conservazione dei dati in RAM e il funzionamento delle periferiche integrate.
Occorre, quindi, che non si travalichino i limiti 2-5.5V.



La soluzione generalmente usata è quella di un regolatore lineare a tre terminali, campo in cui è presente ancora, largamente, il celebre (ma vecchio) 7805 che rende in uscita 5V con una entrata tra 7 e 35V.
Il suo punto di forza è la semplicità del circuito applicativo.

I valori indicati per i condensatori sono quelli minimi consigliati dal foglio dati.

Consideriamo ora un particolare che non è così evidente: vediamo migliaia di esempi di circuiti basati su microcontroller con la Vdd = 5V. E se ne può trarre l'idea che sia questo il valore da usare.

In effetti, la tensione di 5V è stata standardizzata tempo fa per alimentare i circuiti TTL e, con essi, si è così ampiamente diffusa da risultare quella tipica microprocessori e microcontroller, i quali, anche se realizzati in tecnologie MOS, si adeguano ai livelli logici tipici dei TTL, per mantenere la compatibilità con le migliaia e migliaia di altri circuiti integrati.
Se i microprocessori hanno limiti di alimentazione molto stretti, dato che si tratta di componenti tipicamente "solo CPU", in cui contano le prestazioni e che fanno parte di sistemi complessi, dato che memorie e periferiche sono esterne al chip, i microcontroller, in cui tuto è integrato nel componente e che sono pensati per ambiti di impiego differenti, hanno limiti molto più elastici; come abbiamo detto all' inizio, per i PIC va bene qualsiasi tensione tra 2 e 5V. In questo range il microcontroller funzionerà come ci si aspetta.

Possiamo allora alimentarlo anche con una delle altre tensioni "standard", ovvero 3.3V oppure 3V, tensioni per le quali esistono regolatori lineari di applicazione del tutto analoga al 7805.
In questo senso nulla vieta di alimentare il circuito con due o tre batterie da 1.5V in serie o con una cella al Litio 3-4.2V.

In pratica, una alimentazione a batteria ha il solo svantaggio che la tensione scende con l'uso; si rende necessario mantenere sotto controllo questa tensione, sopratutto se si usano accumulatori ricaricabili, onde evitare una scarica troppo profonda.

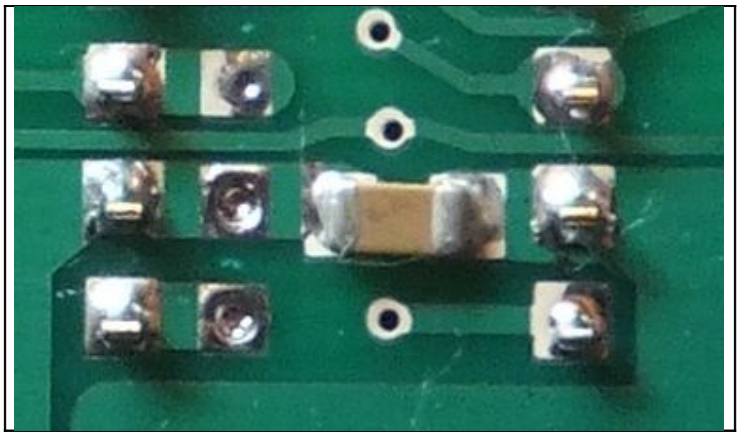
Quindi, togliamo pure dalla mente **l' immagine del PIC + 7805: non è un obbligo**.

Se per quanto riguarda il funzionamento del programma, nel senso della sua esecuzione da parte del microcontroller, la tensione di alimentazione è indifferente se non per la massima frequenza di clock possibile, non lo è certo per quanto riguarda le periferiche collegate.
In particolare, il classico LED che abbiamo usato nelle varie esercitazioni, richiede una resistenza in serie per limitare la corrente e questa dovrà avere un valore diverso (ved. Esercitazione 1) a seconda della tensione. Dato che dalla corrente dipende la luminosità, calcolando una resistenza per una Vdd=5V e poi alimentato il circuito a 3V avremo una riduzione della luminosità del LED. Se il circuito sarà stabilmente alimentato a 3V, occorrerà ricalcolare la resistenza. E si può avere anche i caso in cui periferiche collegate, ad esempio display LCD, abbiano un limite di funzionamento a 5V e, alimentati a 3V, si rifiutino di funzionare correttamente. Questo non esclude la possibilità di avere nello stesso circuiti parti alimentate a tensioni diverse, usando soluzioni di interfaccia tra le varie sezioni.

In generale, **più che il valore assoluto, quello che importa è la qualità di questa alimentazione**: a parte i problemi di accoppiamenti indesiderati, se, in pratica, il microcontroller accetta tensioni di alimentazione non certo perfette, va tenuto presente che, se si utilizza la tensione di alimentazione come riferimento per la conversione AD e per i comparatori, diventa estremamente importante non tanto il suo valore nominale (per il quale basta adeguare i calcoli negli algoritmi) o la sua precisione (che può essere anch'essa corretta da programma), ma **la sua stabilità** e l' assenza di ripple e impulsi di disturbo. In particolare, nei Baseline è obbligo utilizzare la Vdd come tensione di riferimento del modulo ADC e, anche se si tratta di una conversione a soli 8 bit, ogni problema sull'alimentazione si riflette sul risultato e il tentativo di correggere la situazione usando algoritmi di media su più misure non è sempre risolutivo.

Non è questo l' ambito per una discussione approfondita sul problema dell'alimentazione: si trovano con facilità ampie documentazioni, articoli, schemi e consigli a cui rimandiamo.

Qui aggiungiamo solo che un elemento indispensabile che mai deve mancare è **un condensatore ceramico multistrato, a bassa induttanza, collegato tra i pin di alimentazione il più vicino possibile al chip**, meglio de direttamente sotto lo zoccolo. Vediamo qui la soluzione adottata sulla **LPCuB**, utilizzando un componente SMD:

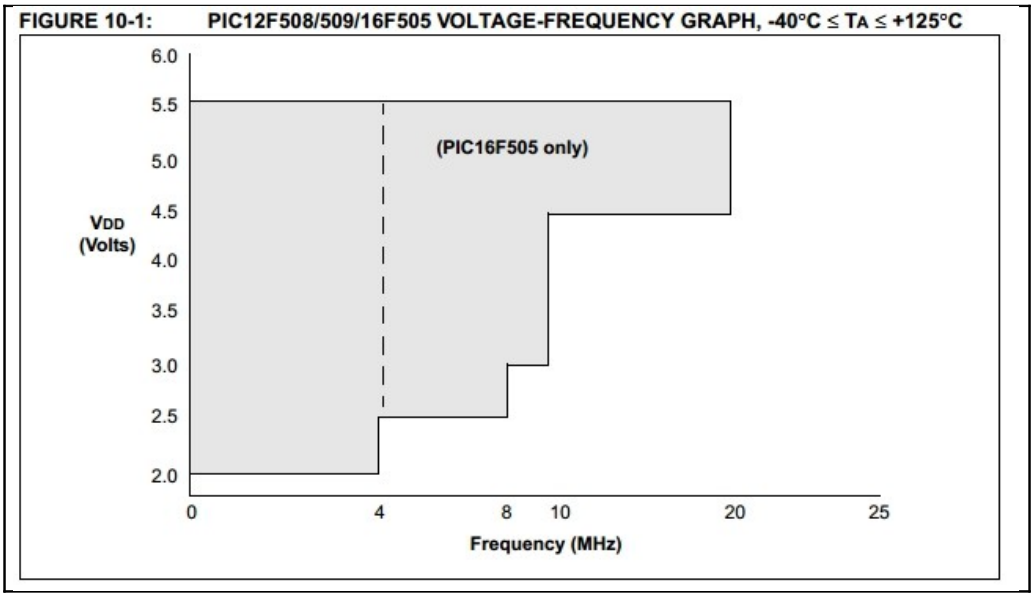


Ovviamente poi nel circuito andranno inseriti condensatori di disaccoppiamento dove occorre, ma questi sono argomenti che riguardano la progettazione hardware, più che quella software.
Va però ben tenuto presente che se l'hardware non è a posto, il software non funziona (e viceversa…).

Dunque, per i PIC:

1. **Non è determinante il valore della Vdd**, che può andare tra 2 e 5V o poco più.
2. Lo può essere, invece, per le periferiche collegate al microcontroller.
3. Se l' applicazione non utilizza la Vdd come riferimento analogico, non ha importanza la sua variazione, sempre tenendo presente i limiti dei punti 1 re 2. Mentre la qualità (assenza di disturbi) è sempre bene sia curata al massimo.
4. Ma se usiamo funzioni analogiche (ADC, comparatori) **la stabilità e la qualità** della Vdd è essenziale, più del suo valore assoluto, il quale sarà correggibile con opportuni algoritmi inseriti nel programma.

Va considerato, inoltre, come abbiamo visto parlando del clock, che la massima frequenza di lavoro è collegata alla tensione di alimentazione in modo diretto:



Minore è la Vdd, minore sarà la frequenza massima raggiungibile. Vediamo ad esempio dal grafico che frequenze di clock superiori a 10MHz sono possibili con sicurezza solo se laVddè maggiore di 4.5V.

La frequenza di lavoro è anche legata al fattore consumo: minore è la frequenza e minore sarà la corrente assorbita.

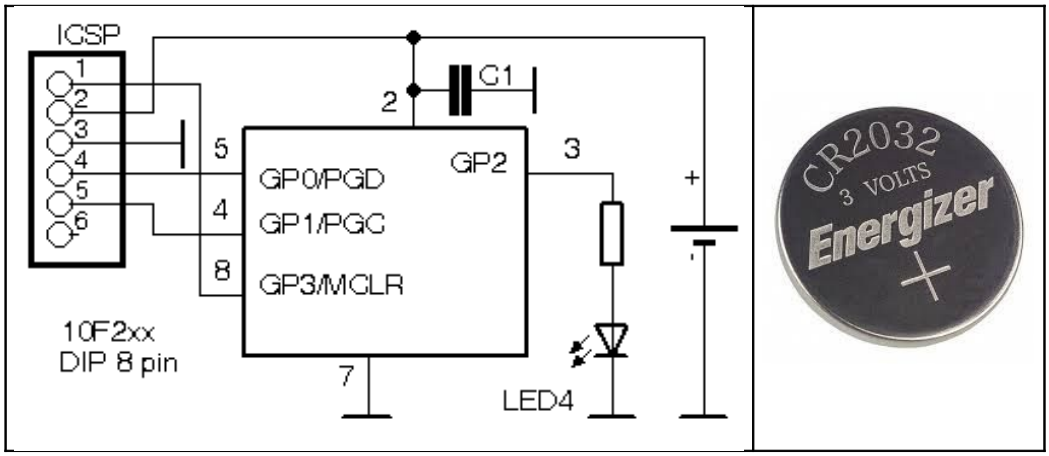
Dunque, abbassando tensione e frequenza, abbassiamo il consumo di corrente.
Ne deriva che una soluzione low power potrà essere realizza implementando:

- Clock minore possibile
- Alimentazione alla minima tensione possibile
- Ampio uso della modalità sleep

Vediamo una applicazione.

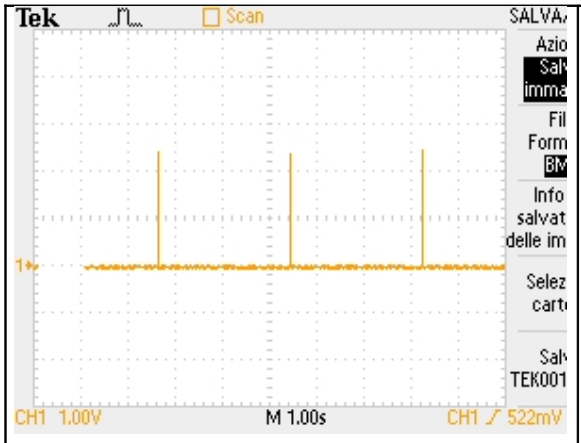
Una applicazione

Possiamo realizzare un circuito basato su questi principi:



Un PIC10F2xx viene alimentato a Vdd=3V con una batteria CR2032; si tratta del classico elemento a disco usato per sostenere la memoria CMOS nei personal computer e quindi assai diffuso e facile da recuperare, ma va benissimo qualsiasi altra batteria tra 2 e 4.5V.

Per quanto riguarda i tempi di commutazione, ricordiamo che non è possibile agire sul clock, dato che questi chip possono operare solamente con quello interno a 4MHz; quindi non è possibile avere un clock a frequenza più bassa. Se usassimo le solite procedure di tempo, il chip sarebbe sempre in funzione a 4MHz e questo produrrebbe un consumo eccessivo.
A questo si ovvia utilizzando la modalità di sleep, utilizzando il WDT come un temporizzatore.

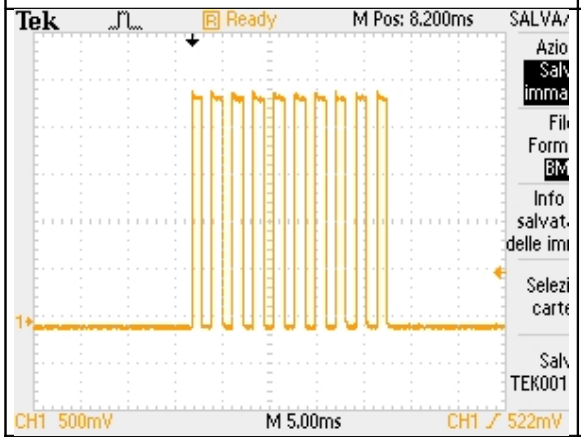


Il LED viene acceso per circa 20ms una volta ogni due secondi.

Nei periodi di pausa, il chip è mandato in sleep per ridurre il consumo.

Ne risulta un oscillogramma come quello a lato (base dei tempi : 1s/div).

Si nota come l'intervallo del WDT sia,in questo caso, di oltre 2.4ms circa.



Per ridurre ulteriormente la corrente media, il periodo di accensione è composto da una serie di impulsi di 1ms.

Nell' oscillogramma a lato la base dei tempi è passata a 5ms/div.

Data la rapidità, non si percepisce sfarfallio e l'occhio riconosce un flash unico, però l' oscilloscopio cattura la situazione reale..

Nel programma, la durata della routine di ritardo **`Delay1ms`** determina il ciclo on/off , mentre il parametro **`step`** stabilisce quante volte viene ripetuto; variandoli, si potranno sperimentare combinazioni diverse.
Con i valori attuali si hanno 10 impulsi da 1ms ciascuno.

Come detto, il tempo di due secondi circa è prodotto non da una routine di conteggio, ma dall' intervento del WDT sullo sleep. Nel complesso, per una corrente del LED,negli istanti di accensione,superiore a 20mA, la corrente media è di circa 2.2uA! Una batteria da 200mAh dovrebbe poter durare migliaia di ore.
Con un LED ad alta luminosità, il lampeggio è ben visibile anche al sole, mentre di notte si può percepire bene a un centinaio di metri di distanza.
La resistenza in serie al LED serve a limitare la corrente. Si trova comunemente la formula di calcolo:

Rled = (Vdd – Vled) / Iled

Dove ***Vled*** è la tensione di conduzione del LED. Se abbiamo ***Vdd=3V***, ***Vled=2V*** e ***Iled=10mA***:

Rled = (3 – 2) / 0.010 = 100ohm

Però, con bassi valori di tensione Vdd e alti valori di tensione di conduzione del LED, va tenuta presente anche la caduta di tensione ***Voh*** sul MOSFET dell'I/O, che non è 0V. Quindi il calcolo corretto sarà:

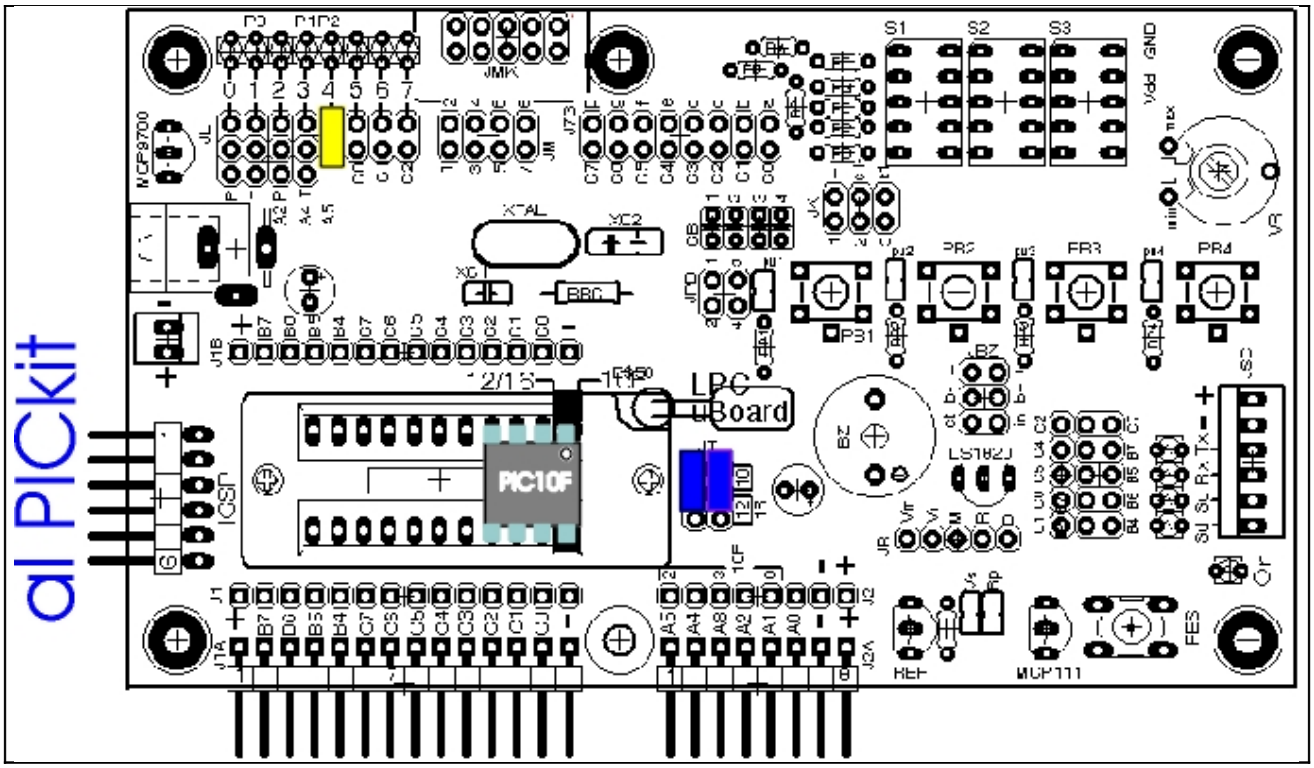
Rled = (Vdd – Vled – Voh) / Iled

Così, se ***Voh=0.4V*** abbiamo:

Rled = (3 – 2 – 0.4) / 0.01 = 60ohm

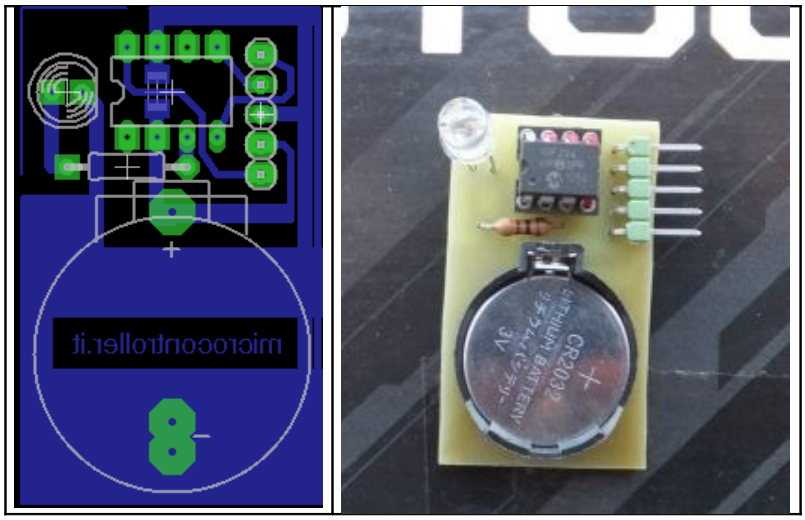
Che potrà essere normalizzato al valore comunemente reperibile di 56ohm.
Va considerato anche il fatto che la tensione di batteria tende a ridursi con l'uso e quindi a ridurre la corrente nel LED; questo ridurrà leggermente la luminosità col tempo.
Dato, però, che i port possono portare nominalmente 25mA e che la corrente al LED viene erogata a impulsi, con un basso duty cycle, possiamo usare una resistenza più bassa di quella calcolata, ad esempio 33ohm, che permette il passaggio di 18mA circa. Con LED a bassa corrente (2-5mA) potrà essere attorno ai 100ohm; per un calcolo preciso del valore, potete consultare queste pagine.
Comunque, la resistenza dipende dal LED, dalla luminosità che vogliamo ottenere e dal consumo di corrente che possiamo accettare, ovviamente senza superare i 25mA gestibili dall'uscita digitale.
Di sicuro non è possibile accendere LED la cui tensione sia maggiore di quella della batteria; quindi, con la batteria scelta non possiamo pensare di usare con questo circuito LED blu o bianchi che hanno una tensione di conduzione più alta di 3V.

Il circuito può essere sperimentato sulla **LPCuB**:



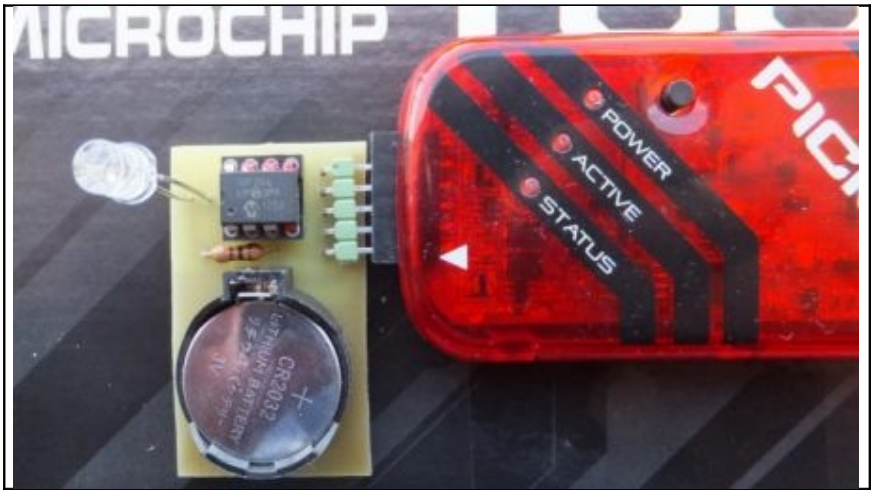
Qui possiamo variare il periodo degli impulsi di accensione ed il loro numero, a seconda di quello che si vuole ottenere: minore è il tempo al di fuori dello sleep, maggiore è il consumo, ma maggiore è la luminosità ottenuta. Minore è il tempo di accensione o il duty cycle, minore è la luminosità percepita del LED, ma minore sarà il consumo e maggiore la durata della pila.
Si potrà far variare anche il tempo di intervento del WDT portandolo a 1s (prescaler 1:64), o anche meno, ottenendo una accelerazione dei lampeggi (maggiore corrente media assorbita).

La scheda di sviluppo contiene molti componenti che assorbono corrente, come il LED verde di ON e il potenziometro VR. Una volta verificato il funzionamento si potrà anche portare il tutto su un piccolo stampato per avere un oggetto finito:

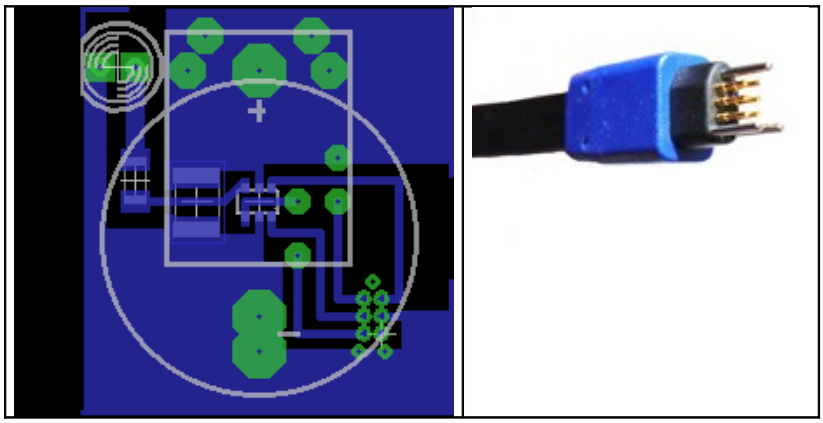


Dono usati componenti comuni, per montaggio su foro, a parte il condensatore in parallelo ai pin di alimentazione del chip (0.1-4.7uF).

Il circuito prevede una presa ICSP per poter collegare il Pickit anche qui e operare modifiche al firmware senza rimuovere il chip. Questa presa dispone di soli 5 pin per ragioni di spazio; viene escluso il pin 6 che non è utilizzato per la programmazione di questi chip:



Volendo miniaturizzare, si potrà usare il package SOT-23, montato sul lato opposto della batteria. Anche qui possiamo prevedere comunque una presa ICSP usando il **Tag-connect**:



Il circuito può essere usato come spia per apparati vari, parte di giocattoli o per individuare oggetti o ostacoli al buio o come semplice gadget; oppure, comandando più LED, può essere la base di una luce di riconoscimento per ciclisti o pedoni notturni. Più LED potranno essere pilotati usando più pin.

La stessa applicazione, con le solite piccole modifiche potrà girare su qualsiasi Baseline, e, in generale, su qualsiasi PIC.

Misura della Vdd attraverso il modulo ADC

Possiamo utilizzare il modulo ADC per valutare la tensione di alimentazione del microcontroller. Si presenta in molte applicazioni la necessità di valutare questa tensione: uno dei casi più comuni è quello dei sistemi alimentati a batteria, dove occorre tenerne sotto controllo la tensione per richiedere la sostituzione quando scarica oppure, nel caso di accumulatori, per evitarne una scarica eccessiva.

Questo è facilmente implementabile in tutti i chip che dispongono di modulo ADC, utilizzandone un canale e un ingresso. La soluzione è evidente, ma occupa un pin e solitamente ne richiede uno ulteriore per una tensione di riferimento esterna, dato che non si può utilizzare la Vdd come riferimento perchè essa è variabile con lo stato di scarica delle batterie.
Sfortunatamente, i Baseline hanno come riferimento per la conversione analogico-digitale proprio e solo la Vdd. Quindi non è possibile collegare un ingresso analogico a questa tensione, dato che varierebbe con essa anche il riferimento e quindi non sarebbe possibile alcuna valutazione di variazioni (dato che il risultato sarebbe sempre identico).
Si potrà utilizzare **il comparatore**, ma è una soluzione che impiega pin, cosa poco gradita nei chip a basso numero di I/O.

Possiamo, però, utilizzare il riferimento interno a 0.6V, il quale risulta costante per qualsiasi variazione della Vdd nel range di funzionamento ammesso, effettuando una misura "inversa".
Quando effettuiamo una conversione AD, il risultato è pari a:

risultato conversione = valore da convertire / (Vref / numero degli step)

Ad esempio, se abbiamo disponibile una conversione a 8 bit, sono possibili 256 step. Con una tensione di riferimento di 5V, questo vuol dire che ogni step vale 5/256 = 19.531mV per step. Ovviamente, per una misura molto precisa, si cercherà di avere disponibile una tensione di riferimento ben stabile, magari multiplo del numero degli step. Ad esempio, con una Vref da 2048mV ogni step vale 2048/256 = 8mV.
Se la conversione rende 64h (100 decimale), il valore letto è di 100 * 8 = 800mV.

Tutto questo dovrebbe essere chiaro: abbiamo una tensione di riferimento stabile ed una tensione da misurare variabile.
Ma osserviamo la cosa da un opposto punto di vista, ovvero quello di avere il valore da convertire stabile e la tensione di riferimento variabile.
Se abbiamo come riferimento la Vdd a 5V (5000mV) e misuriamo la tensione del riferimento a 0.6V, si ha:

$$V_{ref}/\ step = 5000/256 = 19.531mV/step$$

e la conversione rende:

$$ADC = Vin / (V_{ref}/\ step) = 0.6 / 19.531 = 30.7\ decimale \rightarrow 1Eh.$$

Poniamo ora che la Vdd, usata con Vref+, scenda a 4V; di conseguenza la risoluzione sarà:

$$V_{ref}/\ step = 4000/256 = 15.625\ mV/step.$$

Quindi la lettura del riferimento interno, **che resta costante**, renderà:

$$ADC = Vin / (V_{ref} / step) = 0.6 / 15.625 = 38.4dec \rightarrow 26h$$

Se la tensione Vdd scende a 3V:

$$V_{ref} / step = 3000/256 = 11,71875\ mV/step.$$

Quindi la lettura di 0.6V renderà:

$$ADC = Vin / (V_{ref}/step) = 0.6 / 11,71875 = 51.2\ dec \rightarrow 33h$$

I valori decimali sono arrotondati in difetto al numero intero, dato che l' Assembler utilizza questo principio. Quindi la precisione è limitata, anche perché la tensione del riferimento interno è bassa e il convertitore rende solamente 8 bit.

In generale, nel caso del convertitore AD a 8 bit (256 step) e la tensione di misura a 0.6V (600mV):

$$ADC = 600 / ((V_{dd}*1000) / 256))$$

Da cui :

$$V_{dd} = 0.6 * (256 / ADC) = 153.6 / ADC$$

All'atto pratico possiamo dire che, con il variare della tensione usata come riferimento per la misura, il risultato della conversione di un valore costante, minore della tensione di riferimento, dà **un valore inversamente proporzionale alla tensione di riferimento** stessa.

Possiamo stendere una tabella:

Vdd	ADC		Vdd	ADC		Vdd	ADC		Vdd	ADC
5.5	1B		4.1	25		3.1	31		2.1	48
5.4	1C		4.0	26		3.0	33		2.0	4C
5.2	1D		3.9	27		2.9	34			
5.1	1E		3.8	28		2,8	36			
4.9	1F		3.7	29		2.7	38			
4.7	20		3.6	2A		2.6	3A			
4.6	21		3.5	2B		2.5	3D			
4.5	22		3.4	2D		2.4	3F			
4.3	23		3.3	2E		2.3	42			
4.2	24		3.2	2F		2.2	45			

Utilizzando un foglio di calcolo è facile tracciare la tabella che relazioni la Vdd con il valore ottenuto dalla conversione nella misura della Vref. Utilizzando questa tabella si potrà derivare una lookup table per la misura della Vdd o il valore limite sotto il quale non si deve scendere. In alternativa si potrà implementare un algoritmo di calcolo, ma solitamente la tabella è la via più semplice, anche considerando che una misura indiretta di questo genere tipicamente ha lo scopo di monitorare lo stato della tensione delle batterie che alimentazione il circuito e segnalare lo stato di scarica.

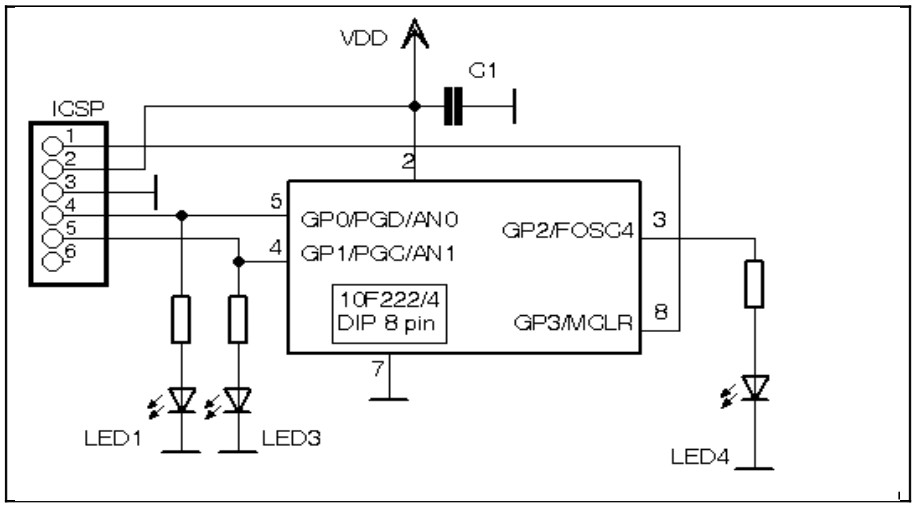
Questa misura "inversa" ha una precisione limitata rispetto a quella ottenibile con un metodo "diretto", ma per l' impiego previsto, è più che adeguata e consente di valutare differenze di una decina di millivolt o minori sulla Vdd. Inoltre, rispetto ad una misura "diretta" con il convertitore ADC, presenta notevoli vantaggi:

- non viene utilizzato un pin esterno, cosa indispensabile per chip con basso numero di pin
- non richiede un partitore esterno che, nelle applicazioni low power, è una aggiunta poco gradita
- non richiede una tensione di riferimento esterna per la precisione della misura, dato che essa è "intrinseca" al metodo di misura stesso

Per quanto riguarda questo ultimo punto, la precisione della tensione fornita dal riferimento interno, come detto, è minore di quella che potrebbe dare un voltage reference esterno, genere LM4040, REF-01, ecc.: il suo valore dipende dalle caratteristiche costruttive e del chip, con una variazione possibile tra 0.5V e 0.7V. Quindi, volendo effettuare misure precise, si richiede una calibrazione.

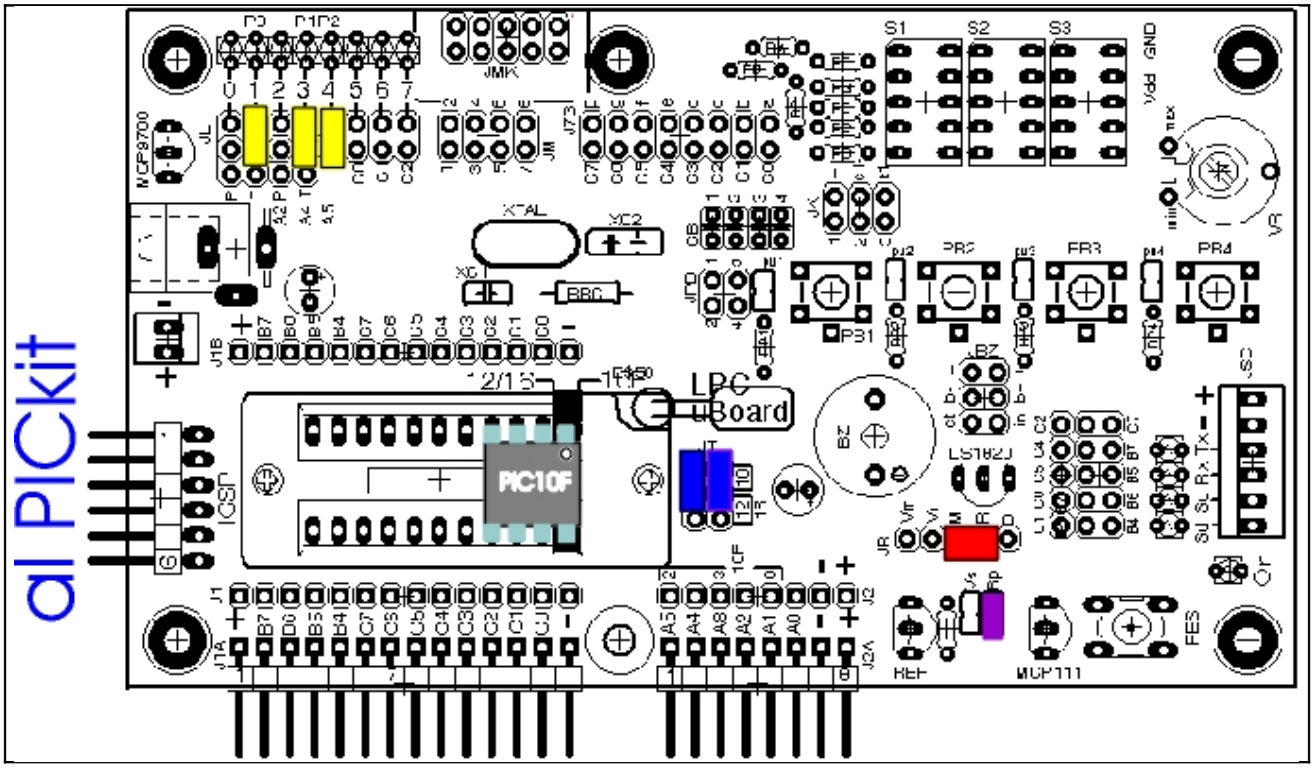
Una applicazione di prova

Rivolgiamoci inizialmente al più piccolo PIC con modulo ADC, il 10F220 o 10F222, ed utilizziamo tre LED per indicare la situazione della Vdd.



Osserviamo che nessun pin è utilizzato per la misura analogica, in quanto il modulo ADC viene collegato internamente al chip alla sorgente di riferimento a 0.6V. I pin sono impiegati per comandare 3 LED che segnalano il valore misurato della tensione di alimentazione.

E sulla **LPCuB**:



Non servono altro che i jumper “gialli” per collegare gli I/O ai LED e, ovviamente, quelli “blu” per il processore. Il tasto RES non è utilizzato e i jumper “rosso” e “viola” non sono indispensabili.

Il programma.

Vogliamo accendere i tre LED in modo da indicare tre limiti nella tensione di alimentazione, in questo modo:

Vdd	ADC	LED		
		1	3	4
Vdd< 3.3	adc>2Eh	x	-	-
3.3<Vdd<4.5	2Eh>adc>22h	-	x	-
Vdd>4.5	adc<22h	-	-	x

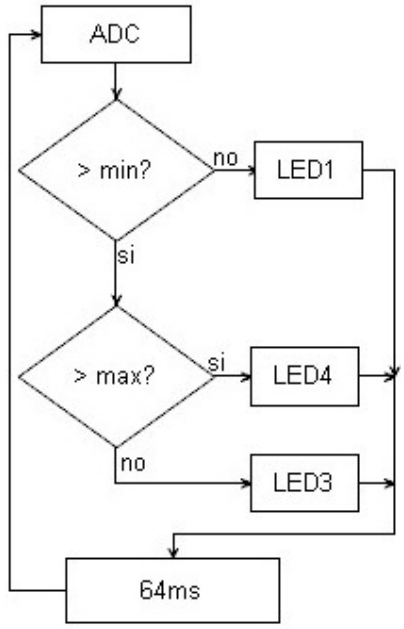
Se facciamo variare la Vdd osserveremo che i LED commutano al raggiungere delle soglie imposte.

Per ottenere la misura indiretta, colleghiamo l’ ingresso del modulo ADC con il riferimento interno.

Non serve alcun ingresso analogico:

```
; ADC per misura riferimento interno
    movlw  b'00111101'
           ; 00------  no ingressi analogici
           ; --11----  INTOSC/4
           ; ----11--  riferimento interno
           ; -------1  abilita ADC
    movwf  ADCON
```

Dopo di che eseguiamo la conversione come abbiamo visto nell' esercitazione relativa e compariamo il risultato con i limiti previsti, secondo questo flowchart:




```
 ; limiti
 CONSTANT  min = 0x2E     ; 3.0V
 CONSTANT  max = 0x22     ; 4.5V

; ADRES > min? (Vdd>3.3V ?)
     CFLJA  ADRES, min, next1
; ADRES < min
     movlw  b'001'        ; LED 1 acceso
     movwf  GPIO
     goto   mainloop

; ADRES > max? (Vdd>4.5V ?)
next1    CFLJA  ADRES, max, next2
; min<ADRES<max
     movlw  b'010'        ; LED 3 acceso
     movWf  GPIO
     goto   mainloop

; ADRES>max
next2
     movlw  b'100'        ; LED 4 acceso
     movwf  GPIO
     goto   mainloop
```

Una macro **CFLJA** rende la cosa alquanto più comoda che non la serie di opcode sottintesi:

```
;*****************************************************************
; CFLJA Compare File to Literal and Jump if Above
; Compara il contenuto di un registro con un numero e va a indirizzo
; se file > literal
CFLJA  MACRO file,lit,addr
       movlw    (255-lit)    ; W = (~lit)
       addwf    file,W       ; W = file + W = file + (~lit)
       skpnc                 ; C=0 se file <= lit, esci
        goto    addr         ; C=1 se file > lit, vai a indirizzo
       ENDM
```

Impieghiamo la somma **addwf** invece della sottrazione **subwf**, con il complemento a 1 del numero fisso con cui si effettua la comparazione.
Questa macro, assieme ad altre, è offerta in un file ***compmacro.asm*** che viene incluso nel sorgente.
Ricordiamo ancora una volta che, delle macro dichiarate con l' inclusione, entrano a far parte della

compilazione solamente quelle che vengono poi utilizzate, mentre le altre restano “virtuali” senza occupare memoria programma.
Le successive conversioni AD sono intervallate da una routine di ritardo di 64ms ottenuta con l’ ausilio di Timer0. In una applicazione reale, qui sarà possibile inserire altre operazioni che il micro deve svolgere.

Ovviamente, sarà possibile verificare il funzionamento del programma solamente disponendo di una Vdd variabile.

 Va tenuto presente che la conversione AD riporta il valore misurato in base alla Vref ed alla tensione di ingresso.
Se la Vdd, che, qui, è il riferimento della conversione, è afflitta da ripple o disturbi e questi si sovrappongono al valore nominale durante il campionamento effettuato dal modulo ADC, il risultato ottenuto sarà errato e afflitto da variazioni apparentemente casuali.

La via per correggere il problema è quella di disporre di un Vdd adeguata. Se questo non è possibile, occorrerà implementare un algoritmo che calcoli una media di più campionamenti (anche se questo metodo non è sempre risolutivo).

Una applicazione più complessa.

In base a quanto abbiamo visto finora, possiamo realizzare un circuito che ci permetta di leggere su un display il valore della tensione Vdd misurata.
Per questo, sfruttiamo quanto abbiamo appreso nella esercitazione 11A, dove abbiamo visto come sia possibile trasferire un valore da un registro a due cifre a 7 segmenti.

Abbiamo, quindi, da gestire tre processi "simultanei", che, per quanto detto nell'esercitazione 10A, in effetti facciamo scorrere sequenzialmente:

1. conversione AD
2. trasposizione del risultato sul display
3. gestione del multiplex del display

Dalla conversione AD otteniamo un valore esadecimale, ma possiamo trasformarlo in una indicazione in volt attraverso una lookup table, realizzata in base alla tabella vista qui sopra.
Nella tabella mettiamo in relazione il risultato della conversione con il valore corrispondente della tensione:

```
  ......
   addwf   PCL,f                ; punta PC
;                #      ADC
   retlw   0x57   ;0    1B
   retlw   0x55   ;1    1C
   retlw   0x53   ;2    1D
   retlw   0x51   ;3    1E
   retlw   0x50   ;4    1F
   retlw   0x48   ;5    20
   retlw   0x47   ;6    21
   retlw   0x45   ;7    22
   retlw   0x44   ;8    23
  ......
```

Dobbiamo considerare i valori tra 1Bh (derivato dalla misura con una Vdd=5.7V) a 4Ch (derivato da una Vdd=2V). Si tratta di 50 valori, ma non è questo il problema, dato che la tabella si può estendere fino a 256 elementi (l' indice di ricerca nella tabella è W, che, essendo a 8 bit, va da 00 a FFh). Il problema reale è doppio: dobbiamo far partire la tabella non da 1Bh, ma da 0 e dobbiamo evitare che, per una qualsiasi ragione, l' indice superi la fine della tabella.
Se l'indice puntasse oltre la fine della tabella, verrebbero eseguiti i codici contenuti in quelle locazioni, con risultati imprevedibili.
Per questo inseriamo all' ingresso della tabella un filtro che impedisce l' entrata di valori troppo bassi o troppo alti, rinviandoli ad una segnalazione di errore:

```
; lookup table per convertire ADC da hex in volt
volttbl_r:
   CFLJB   Vtemp, min, aderL   ; se ADC<minimo, errore LL
   CFLJA   Vtemp, max, aderH   ; se ADC>massimo, errore HH   retlw
  ......
```

Se il valore in ingresso è minore del minimo della tabella, si esce dalla stessa con un errore. Così pure se è superiore al massimo.
Per partire da 0 con l' indice, basta sottrarre al valore in ingresso della tabella quello del minimo:

```
    movlw   min                 ; parti da 0 nella tabella
    subwf   ADRES,w
```

Così, ad esempio, se la conversione rende 1Bh, sottraendo il minimo, che è 1Bh, si avrà in W un indice pari a 0, che punterà il primo della tabella. Se il valore è 1Fh, sottraendo il minimo si avrà 4, che corrisponde alla quinta riga della tabella. E così via.

Ad ogni valore in uscita dall' ADC facciamo corrispondere nella tabella il valore in volt espresso con due cifre, ma in base esadecimale. Queste cifre ci serviranno per introdurle, una alla volta, in una seconda lookup table che le trasforma nelle maschere per l' accensione dei segmenti, come abbiamo visto nella esercitazione 5A:

```
; segment data table - display catodo comune
segtbl_r
        andlw   0x0F            ; solo nibble basso
        addwf   PCL,f           ; punta PC
        retlw   b'00111111'     ; "0" -|-|F|E|D|C|B|A
        retlw   b'00000110'     ; "1" -|-|-|-|-|C|B|-
        retlw   b'01011011'     ; "2" -|G|-|E|D|-|B|A
        retlw   b'01001111'     ; "3" -|G|-|-|D|C|B|A
        retlw   b'01100110'     ; "4" -|G|F|-|-|C|B|-
        retlw   b'01101101'     ; "5" -|G|F|-|D|C|-|A
        retlw   b'01111101'     ; "6" -|G|F|E|D|C|-|A
        retlw   b'00000111'     ; "7" -|-|-|-|-|C|B|A
        retlw   b'01111111'     ; "8" -|G|F|E|D|C|B|A
        retlw   b'01101111'     ; "9" -|G|F|-|D|C|B|A
        retlw   b'00111000'     ; "L" -|-|F|E|D|-|-|- exit per Ah
        retlw   b'01110110'     ; "H" -|G|F|E|-|C|B|- exit per Bh   retlw
  ......
```

Vediamo di capire il meccanismo.

1. La conversione AD rende ad esempio il valore **1Fh**.
2. Lo riduciamo a **1Fh-1Bh = 4**
3. **W = 4** punta la riga `retlw  0x50  ;4  1F`
4. Quindi usciamo dalla tabella con **W=50h**
5. Immettiamo nella tabella dei segmenti il nibble basso, ovvero 0
6. Otteniamo la prima riga `retlw  b'00111111'`
7. Riportando questa sul port di uscita acenderemo i segmenti corrispondenti alla cifra 0.
8. Passiamo il nibble alto, ovvero 5, e otteniamo la riga `retlw  b'01101101'`
9. Riportando questa sul port di uscita accenderemo i segmenti corrispondenti alla cifra 5.

In pratica, l'oggetto dei `retlw` della prima tabella è il valore in decimale che il display andrà a visualizzare attraverso la seconda tabella: abbiamo effettuato la conversione senza utilizzare formule matematiche, che avrebbero impegnato un numero di istruzioni molto più consistente.

Resta ancora da vedere come funzionano le segnalazioni di errore: le macro di comparazione iniziali rinviano a due uscite della tabella:

```
aderL   retlw   0xAA    ; display LL
aderH   retlw   0xBB    ; display HH
```

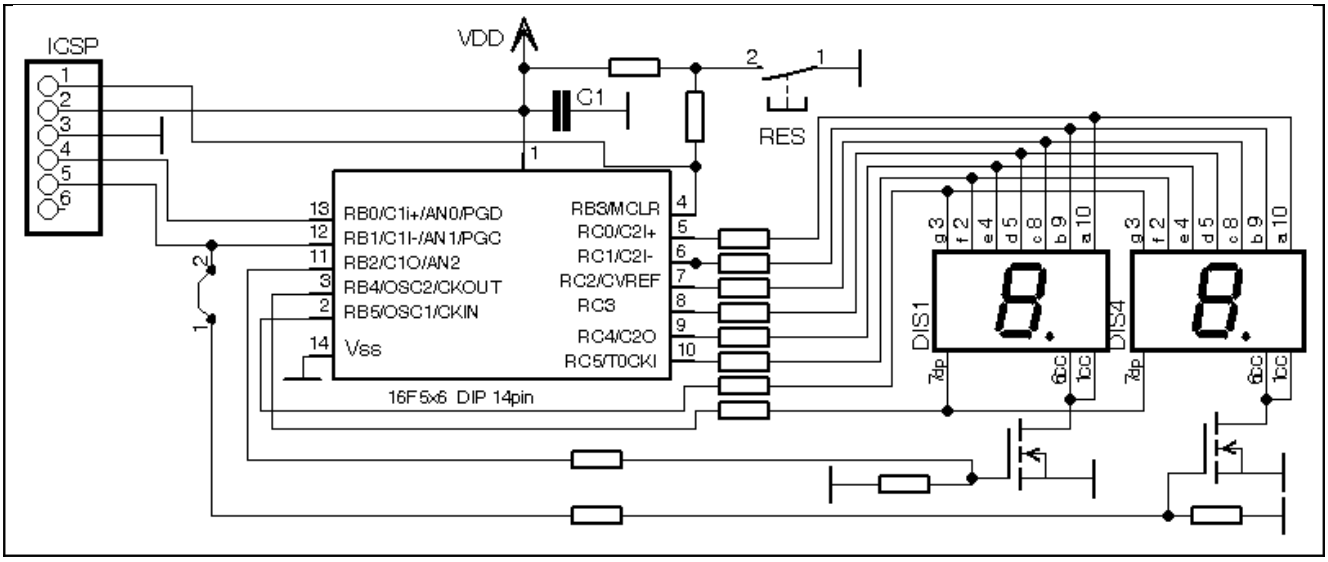
Se il valore convertito è troppo basso, si esce con **W=AAh**. Ah è il numero esadecimale successivo a 9 e quindi la tabella dei segmenti punta alla riga `retlw  b'00111000'` che accenderà i

segmenti corrispondenti alla lettera **L**.
Ugualmente, un valore oltre il massimo farà uscire con **W = BBh** e nella tabella segmenti questo corrisponde alla maschera per accendere la lettera **H**.

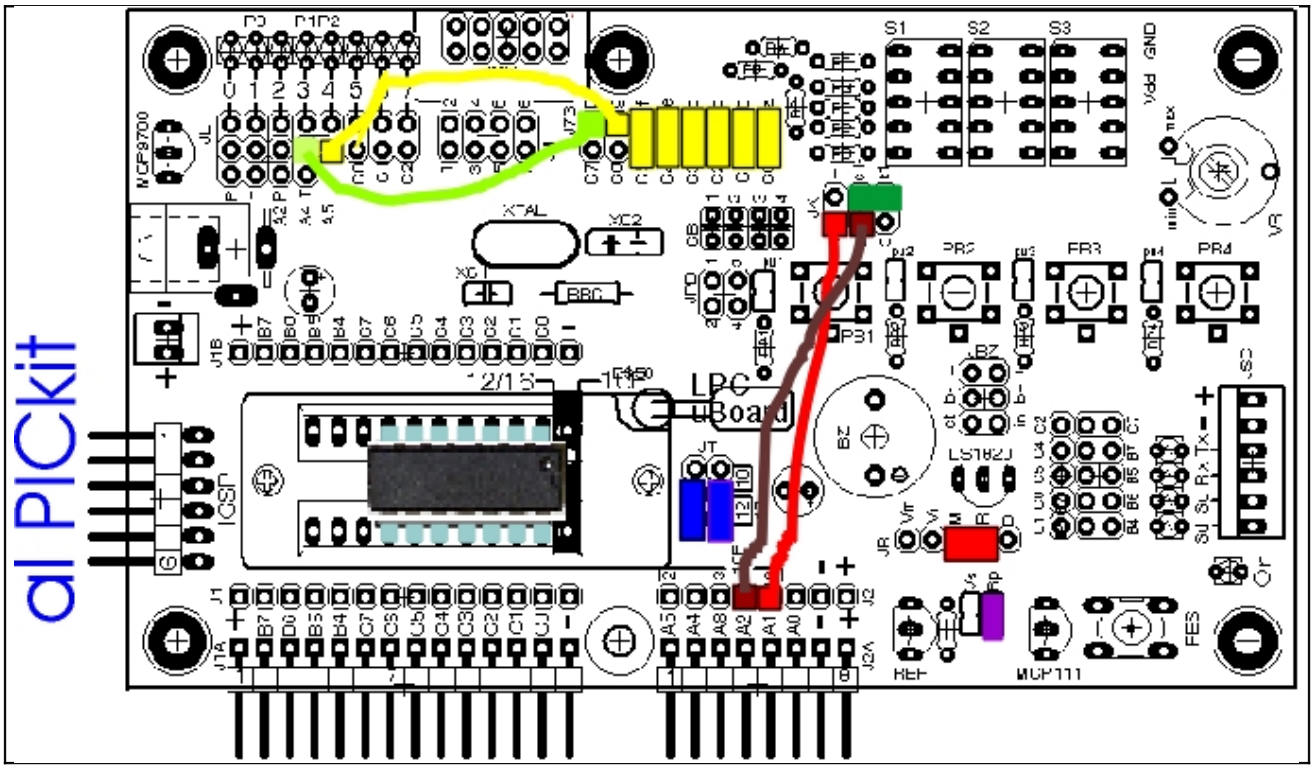
Dal punto di vista circuitale ci occorre un PIC che abbia:

- pin sufficienti per comandare i segmenti
- e il modulo ADC

per cui la scelta cade obbligatoriamente su 16F506/526, per il quale utilizziamo questo schema:



E sulla **LPCuB**:



I jumper “gialli” collegano RC5:0 ai segmenti f:a
Il jumper volante “giallo “ collega RB5 al segmento g

Il jumper volante “verde “ collega RB4 al punto decimale
Il jumper volante “marrone” collega RB2 al gate del MOSFET delle unità (S2)
Il jumper volante “rosso” collega RB1 al gate del MOSFET delle decine (S1)
Il jumper “verde” collega il gate del MOSFET che comanda la cifra S1
Il pulsante RES è utilizzato e quindi i jumper “rosso” e “viola” sono necessari.

Durante la programmazione del chip occorre staccare il jumper volante “rosso”, ovvero scollegare dalla presa ICSP il gate del MOSFET di S1, perché il carico su RB1/PGC impedirebbe l’ operazione.

Il programma.

Ne abbiamo già tratteggiato i punti essenziali.
Se osserviamo il sorgente (***18A_526.asm***) vediamo che ricalca esattamente la struttura di quanto visto nell’ esercitazione 11A : il timer 0 costituisce la base dei tempi che condiziona il funzionamento del multiplex.

Qui utilizziamo un ciclo di 4ms circa; l’uso di riferimenti simbolici anche per le costanti consente di modificare con rapidamente e semplicemente questo tempo (potremmo usare anche 2ms, come nella versione dell’esercitazione 11A, dato resterebbe comunque tempo più che sufficiente per la conversione AD e la gestione delle tabelle).

Per quanto riguarda l’ADC, effettuiamo la conversione ogni secondo circa, usando un contatore per totalizzare 256 periodi da 4ms:

```
; conversione AD
        decfsz   cntr,f      ; acquisisci solo ogni tempo * bittmr
         goto    dslp_3      ; 256 * 4ms = 1s circa
        movlw    tempo       ; ricarica contatore
        movwf    cntr
```

La scelta è dettata da queste considerazioni:

1. la variazione della tensione di alimentazione dovuta al decadere della batteria è un fenomeno abbastanza lento. Non serve monitorare la Vdd a livello di ms, ma di secondi.
2. il risultato della conversione AD è soggetto ad oscillazioni quasi certe sull’ultima cifra. Se avessimo un ripetersi molto rapido delle conversioni, tale da fornire al display l’ultima cifra variabile, l’ effetto sarebbe quello di non poterla leggere.

In effetti, la conversione AD è soggetta di per se ad un errore di 1/2LSB ed oltre a questo, stiamo operando su valori che rendono la metà degli 8 bit disponibili, dato che la tensione in misura (0.6V) è molto minore del riferimento (2-5.5V). Osserviamo nella tabella come il risultato diventi più preciso col diminuire della differenza tra tensione di riferimento e tensione da misurare: nella prima parte della tabella la differenza di 1 nel risultato della conversione, a causa della misura inversa, deriva dalla variazione di decine di millivolt nella Vdd. Se ci troviamo con una Vdd affetta da un minimo ripple e/o ad un valore intermedio tra due in tabella, il risultato si troverà a “saltare” tra due valori vicini. Questo può essere minimizzato solamente applicando un qualche algoritmo di media, ma è riducibile anche rallentando le conversioni in modo che una eventuale variazione dell’ ultima cifra sia riportata solo a scadenze dell’ ordine dei secondi.

Quindi, se ci troveremo con il diplay che passa da 4.8 a 5.0 è perchè probabilmente la Vdd ha un valore intermedio. Una maggiore definizione sarebbe possibile usando un ADC con un numero maggiore di bit e un riferimento interno più alto, come quelli disponibili nei PIC18F.
Nel prototipo usato per l' esercitazione è stata introdotta una correzione del valore in uscita dalla conversione: il contenuto del registro ADRES viene copiato in una locazione temporanea Vtemp, che sarà usata per le azioni successive, e qui viene incrementato di 1:

```
        movf    ADRES,w
        movwf   Vtemp
        incf    Vtemp,f     ; correzione +1
```

L'AN1072 di Microchip, indicata in Bibliografia, fornisce molti dettagli sull' argomento, anche se è riferito a Midrange/Enhanced Midrange e PIC18 che hanno moduli ADC molto più complessi e performanti che non quelli dei Baseline..

Per una presentazione migliore, accendiamo anche il punto decimale del digit delle decine:

```
        bsf     segmdp      ; aggiungi punto decimale
        DECon               ; accendi decine
```

che sarà spento assieme al digit stesso (se non lo facessimo, anche il digit delle unità avrebbe il dp acceso):

```
        DECoff              ; spegni decine
        bcf     segmdp      ; spegni punto decimale
```

Le tabelle sono posizionate nei primi 256 bytes della pagina 1:

```
TABLES  CODE  0x200
```

Ricordiamo **i limiti imposti dai Baseline per la gestione di tabelle e chiamate di subroutines**, che abbiamo già trattato.
Tabelle e subroutines sono accessibili in via indiretta con salti dalla pagina 0, mentre `pagesel` garantisce la corretta indicizzazione.
La posizione delle tabelle in pagina 1 consente di avere le istruzioni principali in pagina 0, che resta praticamente tutta disponibile.

Il sorgente è scritto in una forma rilocabile. L'uso intensivo di simboli e di librerie di macro consente la stesura di un sorgente abbastanza lineare e semplice.

Anche in questo caso occorre una Vdd variabile per alimentare il circuito.

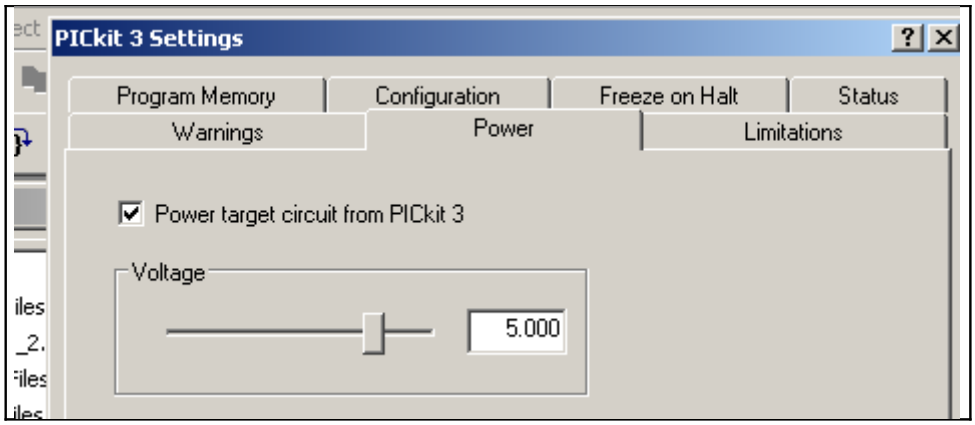
Variare la Vdd.

Per testare gli esempi precedenti occorre una tensione variabile di alimentazione per il microcontroller.

Una prima via è quella di disporre di un alimentatore regolabile almeno tra 2 e 5V, per fornire la Vdd al circuito. Ottimo il modulo **UBS-12** , progettato per lavorare con la **LPCuB**.

Se non avete disponibili né l' uno, né l' altro, esiste comunque una via alternativa: **utilizzare la tensione fornita dal Pickit**.

Abbiamo visto all' inizio del corso che i circuiti degli esempi possono essere alimentati direttamente dal Pickit. E la tensione che questo fornisce è regolabile, sia don il tool come programmer che come debugger. Dalla Dalla voce ***Settings*** del relativo sotto menu, dobbiamo abilitare la funzione Power, dove è possibile selezionare il valore della tensione generata con un cursore a scorrimento:



La tensione è variabile in teoria tra 2.75V e 5.5V. In pratica, la massima tensione non supererà il valore fornito dalla porta USB che alimenta il Pickit e che è tipicamente 5V (ma può essere anche un 5% meno).

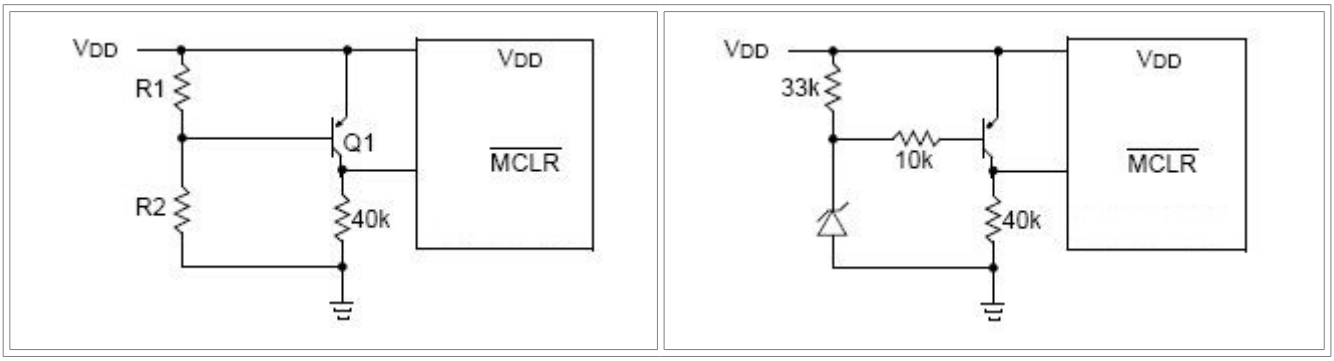
Brown Out Protection

Restando nell'ambito della tensione di alimentazione, occorre citare la protezione contro il Brown-out. Questa funzione non ha a che vedere con il programma, ma è un supporto hardware indispensabile per parare possibili problemi sull' alimentazione.

Con **Brown Out** si identifica una condizione della tensione di alimentazione che scende sotto il valore minimo di sicurezza previsto per il funzionamento del chip o del circuito.
In questo caso, la tensione non arriva a zero e nemmeno al valore che permette il funzionamento del circuito di reset (<0.8V) che, quindi, non interviene.
I Baseline possono operare tra 2 e 5.5V sulla Vdd e, come abbiamo visto in relazione al clock, minore è la tensione, minore sarà la frequenza operativa possibile. Inoltre, al di sotto di 1.5V sulla Vdd non è garantita la tenuta dei dati in RAM. Ne risulta che se la Vdd scende anche brevemente al di sotto di 2V, senza arrivare vicino a 0, è probabile che il chip smetta di funzionare correttamente e in modo non prevedibile.
Queste condizioni si verificano nel caso di brevi mancanze di rete che non riescono ad essere compensate dagli elettrolitici dell' alimentatore oppure con l' uso di batterie che a fine carica o difetti possono scendere al di sotto dei 2V.

Per questa ragione, dai Midrange in su, i PIC incorporano un circuito detto ***Brown Out Protection*** che ha lo scopo di resettare il chip se l' alimentazione è difettosa nel senso indicato.
Però, **i Baseline non dispongono di questo modulo** e, nel caso in cui si abbia a che fare con una alimentazione che può presentare problemi, occorre aggiungere un circuito esterno, collegato al pin di reset (**MCLR**).

Microchip consiglia alcuni circuiti semplici:



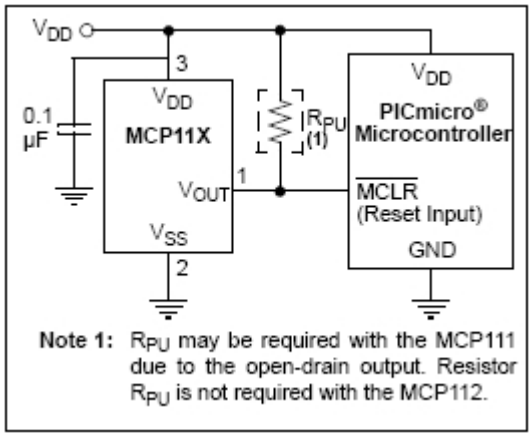
Si tratta di due trigger che collegano alla Vdd il pin **MCLR** fino a che essa è maggiore di una determinata soglia.
Nel primo circuito, ***Q1*** viene spento quando la Vdd scende al di sotto della tensione di conduzione del transistori:

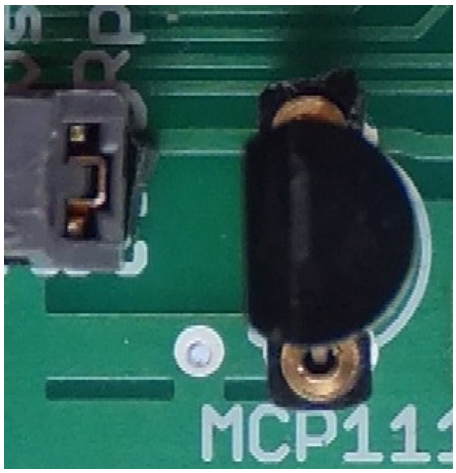
$$Vdd * R1/(R1+R2)$$

Questo valore è attorno a 0.7V, ma dipende molto dal transistor (che sarà un qualsiasi PNP, genere BC807, BC307, ecc) e va cercato sperimentalmente. Per una maggiore precisione, il secondo circuito utilizza un diodo zener. In tal caso si avrà il blocco del transistor per $Vdd < Vz + 0.7V$.
In entrambi i casi si tratta di circuiti che puntano essenzialmente al basso costo, mentre la soglia di intervento va verificata in funzione delle caratteristiche dei semiconduttori usati.

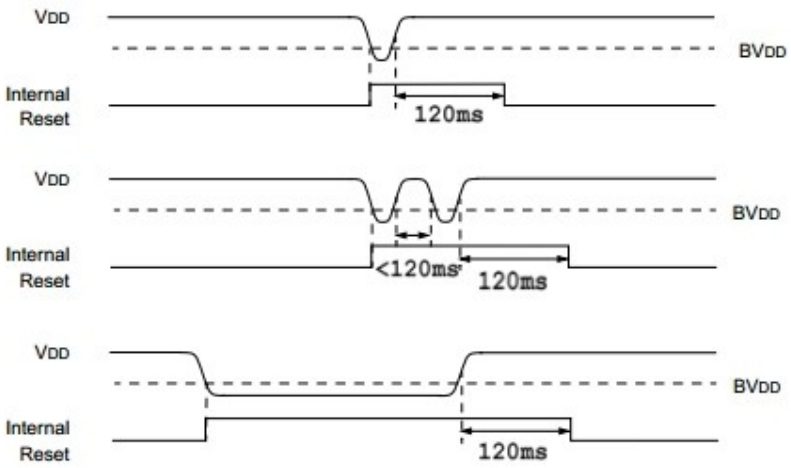
Dal punto di vista di una realizzazione professionale, risulta molto più sensato l' impiego di un ***Voltage Supervisor***, ovvero di un circuito integrato che consenta in un solo componente una elevata

precisione di intervento, due elementi che compensano il maggior costo. In particolare sono adeguati quelli della serie MCP11x di Microchip. La scheda **LPCuB** è prevista per l' inserzione di uno di questi componenti su un apposito zoccolo.





Usando un **MCP111** si dovrà abilitare il pull-up con ul jumper ***Rp***, mentre usando un **MCP112** questo non sarà necessario. Questi integrati sono costruiti per diverse tensioni (1.90V, 2.32V, 2.63V, 2.90V, 2.93V, 3.08V,
4.38V e 4.63V) e quindi si adattano alle più varie condizioni.
La scelta del valore di intervento dipenderà dalla tensione di lavoro minima prevista; ad esempio, se il circuito prevede una Vdd di 5V ben stabilizzata, un intervento a 4.63V, al di sotto del 4.75V (5V-5%) è adeguato. Se si alimenta il circuito con due batterie da 1.5V o una litio da 3V, si potrà scegliere un intervento tra 2.32 e 2.63V a seconda di quale tensione minima i componenti del circuito potranno sostenere e di quella dell'acccumulatore, per evitare una sovra scarica.
inoltre, va considerato anche quanto collegato al microcontroller; ad esempio, se una periferica esterna può lavorare fino ad una Vdd di 2.5V, anche se il micro arriva a 2V (e anche meno..), occorrerà provvedere con una soglia almeno a 2.63Vper garantirne un funzionamento sicuro.
MCP11x non introducono un ritardo nel rilascio del pin di uscita quando la tensione ritorna al di sopra del valore di intervento; il ritardo tra il rilascio del reset e l' avviamento del micro è quello stabilito dalle caratteristiche del chip.



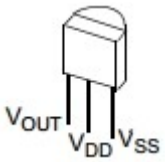
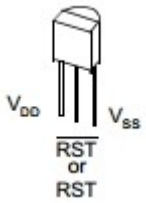
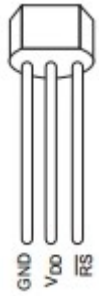
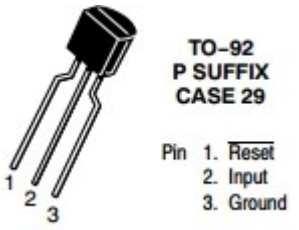
Invece, i modelli MCP10x, 12x e 13x aggiungono 120ms. Il comportamento è quello indicato a lato.

L' inserzione di un ritardo potrebbe essere positiva nel caso in cui si preveda la possibilità di un rapido ripetersi della sottotensione, come nella seconda immagine.

O, anche, quando una estensione del tempo di reset si rende necessaria per altri componenti collegati al microcontroller.

Da notare che MCP10x, 12x e 13x hanno una piedinatura differente da MCP11x e quindi l' inserzione sullo zoccolo non è diretta, ma occorre rifarsi al foglio dati del componente utilizzato.

Giusto per avere una idea delle possibili variazioni:

			
MCP111/2, MCP10xD, TC54	**MCP10xH**	**TC32M, TL7757**	**MC33064, DS1812, AMS26, TS831**

Se si utilizza uno dei tantissimi altri componenti analoghi (TS831, DS1233, TC32M, TC54, EC95809, DS1812, DS1818, MC33064/33164, TL7757, AMS26, MN1380, ecc) è necessario consultare il foglio dati relativo per il corretto inserimento sulla scheda e per la presenza o meno del pull-up.
Possono essere scelti modelli con l' uscita push-pull, ma, in tal caso, non si potrà utilizzare il pulsante di reset (che cortocircuiterebbe il MOSFET alto). Sono quindi da preferire quelli open drain, in cui la chiusura del pulsante di reset non crea problemi.

L' inserzione del Voltage Supervisor non comporta alcuna modifica nel software, se non nel fatto che una analisi delle cause di Reset rivelerà condizioni uguali a quelle dovute all' intervento di un livello basso sul pin MCLR.

Per verificare quanto detto, sarà necessario installare un voltage detector, ad esempio MCP111, nella versione per 3.08V (ma va bene qualsiasi altra). Occorre poi collegare la LPCuB ad un alimentatore variabile tra 2 e 5V. Caricato uno qualsiasi dei programmi precedenti, se la tensione di alimentazione è maggiore di quella di intervento del voltage detector, il circuito funzionerà regolarmente; abbassando la tensione il programma continuerà a funzionare fino a che questa non scende al di sotto della soglia, momento in cui il programma si arresterà e rimarrà in condizioni di reset fino al ritorno di una tensione superiore.

BIBLIOGRAFIA

Qualche link ad application notes di possibile interesse

Using voltage regulator to convert 5-12V range to 3.3V
Improving the Transient Immunity - Freescale
Hardware design guideline power supply and voltage measurement – ST
External Brown-out Protection – Atmel
TRIAC+Microcontroller Safety Precautions – ST
Protecting Microcontroller Systems against Power-Supply Imperfections – Philips
Transformerless Power Supply – Microchip
PRINTED CIRCUIT BOARD DESIGN NOTES – Silabs
AN1072 Measuring VDD Using the 0.6V Reference - Microchip

17A_10F.asm

```
;*****************************************************************
; 17A_10F.asm
;-----------------------------------------------------------------
;
;    Titolo        :  Corso Assembly & C - Esercitazione 17A
;                     Lampeggia un LED collegato a GP2.
;                     ton=20ms  toff=2,3s in sleep per basso consumo.

;    PIC           :  10F200/2/4/6
;    Supporto      :  MPASM
;    Versione      :  1.0
;    Data          :  01-05-2013
;    Ref. hardware :
;    Autore        :  afg
;
;
;  Impiego pin :
;  ----------------
;     10F200/2/4/6 @ 8 pin DIP       10F200/2/4/6 @ 6 pin SOT-23
;
;                 |__\/__|                 *______|
;            NC -|1    8|- GP3        GP0 -|1    6|- GP3
;           Vdd -|2    7|- Vss        Vss -|2    5|- Vdd
;           GP2 -|3    6|- NC         GP1 -|3    4|- GP2
;           GP1 -|4    5|- GP0              |______|
;                |______|
;
;                              DIP SOT
;    NC                         1:
;    Vdd                        2:  5: +
;    GP2/T0CKI/FOSC4/[COUT]     3:  4: Out LED alla Vss
;    GP1/[CIN-]/ICSPCLK         4:  3:
;    GP0/[CIN+]/ICSPDAT         5:  1:
;    NC                         6:
;    Vss                        7:  2: -
;    GP3/MCLR/VPP               8:  6:
;
;    [] solo 204/6
;
;*****************************************************************
;            DEFINIZIONE DI IMPIEGO DEI PORT
;GPIO map
; |  3  |  2  |  1  |  0  |
; |-----|-----|-----|-----|
; | in  | LED |     |     |
;
;#define    GPIO,GP0   ;
;#define    GPIO,GP1   ;
#define    LED GPIO,GP2   ; LED tra pin e Vss
;#define    GPIO,GP3   ;
;
;#################################################################
; Definizione del processore
 #ifdef __10F200
```

```
        LIST      p=10F200
        #include <p10F200.inc>
 #endif
 #ifdef __10F202
        LIST      p=10F202
        #include <p10F202.inc>
 #endif
 #ifdef __10F204
        LIST      p=10F204
        #include <p10F204.inc>
#define  proccomp
 #endif
 #ifdef __10F206
        LIST      p=10F206
        #include <p10F206.inc>
#define  proccomp
 #endif
        radix     dec           ; default numeri decimali

;*****************************************************************
;                           CONFIGURAZIONE
;
; Si WDT, no CP, pin4=GP3;
  __config  _CP_OFF & _MCLRE_OFF  & _WDT_ON

;*****************************************************************
;                           RAM
    CBLOCK 0x10
    d1,d2,d3              ; counter per temporizzazione
    endc

step EQU  .10             ; ripetizioni lampeggio

;*****************************************************************
;                            RESET ENTRY
;
; Reset Vector
RESET_VECTOR   ORG     0x00

        clrwdt
; calibrazione oscillatore interno
        andlw   0xFE
        movwf   OSCCAL
; disabilita T0CKI per avere GP2 libero
; prescaler 1:128 al WDT
; OPTION def '11111111'
;             1------- GPWU disabilitato
;             -1------ GPPU disabilitato
;             --0----- clock interno (no T0CKI)
;             ---1---- done
;             ----1--- prescaler al WDT
;             -----111 1:128  2.3s circa
     movlw  b'11011111'
     OPTION

 #ifdef  proccomp   ; solo 10F204/6
; disabilita comparatore
```

```
     movlw  b'11110111'
     movwf  CMCON0
 #endif

; GP2 out
     movlw   b'11111011'
     tris    GPIO

     movlw   step
     movwf   d3
ll   bsf     LED
     call    Delay1ms
     bcf     LED
     call    Delay1ms
     decfsz  d3
      goto   ll

     sleep


; subroutines
Delay1ms
        movlw   0xC6
        movwf   d1
        movlw   0x01
        movwf   d2
Delay1ms_0
        decfsz  d1, f
         goto   $+2
        decfsz  d2, f
         goto   Delay1ms_0
        goto    $+1
        nop
        retlw   0

;*****************************************************************
;                              THE END
    END
```

17A_10FVdd.asm

```
;*****************************************************************
; 17A_10FVdd.asm
;-----------------------------------------------------------------
;
;    Titolo        :  Corso Assembly & C - Esercitazione 17A
;                     Valutazione della Vdd attraverso la misura
;                     indiretta della tensione di riferimento del
;                     modulo ADC.

;    PIC           :  10F220/222
;    Supporto      :  MPASM
;    Versione      :  1.0
;    Data          :  01-05-2013
;    Ref. hardware :
;    Autore        :  afg
;
;
;  Impiego pin :
;  ----------------
;     10F220/222 @ 8 pin DIP       10F220/222 @ 6 pin SOT-23
;
;                 |__\/__|                  *------|
;           NC -|1    8|- GP3          GP0 -|1    6|- GP3
;          Vdd -|2    7|- Vss          Vss -|2    5|- Vdd
;          GP2 -|3    6|- NC           GP1 -|3    4|- GP2
;          GP1 -|4    5|- GP0                |______|
;               |______|
;
;                        DIP  SOT
;    NC                   1:     nc
;    Vdd                  2:  5: ++
;    GP2/T0CKI/FOSC4      3:  4: Out LED
;    GP1/ICSPCLK/AN1      4:  3: Out LED
;    GP0/ICSPDAT/AN0      5:  1: Out LED
;    NC                   6:     nc
;    Vss                  7:  2: --
;    GP3/MCLR/VPP         8:  6:
;
;*****************************************************************
;            DEFINIZIONE DI IMPIEGO DEI PORT
;GPIO map
; |  3  |  2  |  1  |  0  |
; |-----|-----|-----|-----|
; |     |LED4 |LED3 |LED1 |
;
#define    LED1  GPIO,GP0   ; LED tra pin e Vss
#define    LED3  GPIO,GP1   ;
#define    LED4  GPIO,GP2   ;
;#define    GPIO,GP3   ;
;
;#################################################################
 #ifdef __10F220
       LIST      p=10F220     ; definizione del processore
       #include <p10F220.inc>
```

```
 #endif
 #ifdef __10F222
        LIST      p=10F222     ; definizione del processore
        #include <p10F222.inc>
 #endif

        radix     dec          ; default numeri decimali

;*****************************************************************
;                           CONFIGURAZIONE
;
; 4MHz, no WDT, no CP, pin4=GP3;
  __config  _IOFSCS_4MHZ  & _CP_OFF & _MCLRE_OFF  & _WDT_OFF

;*****************************************************************
;                           RAM

;*****************************************************************
;                           COSTANTI
;limiti
 CONSTANT  min = 0x33   ; 3.0V
 CONSTANT  max = 0x22   ; 4.5V

;*****************************************************************
;                          MACRO
; set macro per comparazione
 #include C:\PIC\Library\Baseline\compmacro.asm

;*****************************************************************
;                            RESET ENTRY
;
; Reset Vector
RESET_VECTOR   ORG     0x00


;*****************************************************************
;                         MAIN PROGRAM
;
Main:
; calibrazione oscillatore interno
        andlw   0xFE
        movwf   OSCCAL

; disabilita T0CKI per avere GP2 libero
; prescaler 1:256 al Timer0
; OPTION def '11111111'
;             1------- GPWU disabilitato
;             -1------ GPPU disabilitato
;             --0----- clock interno (no T0CKI)
;             ---1---- done
;             ----0--- prescaler al Timer0
;             -----111 1:256
     movlw  b'11011111'
     OPTION

; ADC per misura riferimento interno
            ; 00------  no ingressi analogici
```

```
            ; --11----  INTOSC/4
            ; ----11--  riferimento interno
            ; ------0-  GO/Done
            ; -------1  abilita ADC
     movlw  b'00111101'
     movwf  ADCON0

; GP out
     movlw  b'11111000'
     tris   GPIO

mainloop:
; ritardo 64ms
     movlw  .6
     movwf  TMR0
lll  movf   TMR0,w
     bnz    lll

; avvia conversione
     bsf    ADCON0,GO
adlp btfsc  ADCON0,NOT_DONE
      goto  adlp

; ADRES > min? (Vdd>3.3V ?)
     CFLJA  ADRES, min, next1
; ADRES < min
```

```
     movlw  b'001'        ; LED 1 acceso
     movwf  GPIO
     goto   mainloop

; ADRES > max? (Vdd>4.5V ?)
next1
     CFLJA  ADRES, max, next2
; min<ADRES<max
     movlw  b'010'        ; LED 3 acceso
     movWf  GPIO
     goto   mainloop

; ADRES>max
next2
     movlw  b'100'        ; LED 4 acceso
     movwf  GPIO
     goto   mainloop


;*****************************************************************
;                              THE END
```

```
        END
```

17A_526.asm

```
;*****************************************************************
; 17A_526.asm
;-----------------------------------------------------------------
;
;    Titolo         : Corso Assembly & C - Esercitazione 17A
;                     Misura indiretta della Vdd con risultato ADC
;                     in volt su due cifre
;
;    PIC            : 16F506/526
```

```
;    Supporto      : MPASM
;    Versione      : V.519-1.0
;    Data          : 01-05-2013
;    Ref. hardware :
;    Autore        :  afg
;
;-------------------------------------------------------------------------
;
;  Impiego pin :
;  ----------------
;       16F506/526 @ 14 pin
;
;                    |__\/__|
;              Vdd -|1   14|- Vss
;              RB5 -|2   13|- RB0
;              RB4 -|3   12|- RB1
;         RB3/MCLR -|4   11|- RB22
;              RC5 -|5   10|- RC0
;              RC4 -|6    9|- RC1
;              RC3 -|7    8|- RC2
;                   |______|
```

```
;
;    Vdd                    1: ++
;    RB5/OSC1/CLKIN         2: Out  segm g
;    RB4/OSC2/CLKOUT        3:
;    RB3/!MCLR/VPP          4: MCLR
;    RC5/T0CKI              5: Out  segm f
;    RC4/C2OUT              6: Out  segm e
;    RC3                    7: Out  segm d
;    RC2/CVref              8: Out  segm c
;    RC1/C2IN-              9: Out  segm b
;    RC0/C2IN+             10: Out  segm a
;    RB2/C1OUT/AN2         11: Out  gate unità
;    RB1/C1IN-/AN1/ICSPC   12: Out  gate decine
;    RB0/C1IN+/AN0/ICSPD   13:
;    Vss                   14: --
;
;*************************************************************************
; scelta del processore
  #ifdef  __16F526
         LIST      p=16F526
         #include <p16F526.inc>
  #endif
```

```
  #ifdef  __16F506
        LIST      p=16F506
         #include <p16F506.inc>
  #endif

        radix  dec

;*****************************************************************
;                           CONFIGURAZIONE
  #ifdef  __16F526
; Oscillatore interno, 4MHz, no WDT, no CP, MCLR
 __config  _IntRC_OSC_RB4 & _IOSCFS_4MHz & _WDTE_OFF & _CP_OFF &
_CPDF_OFF & _MCLRE_ON
  #endif
  #ifdef  __16F506
; Oscillatore interno, 4MHz, no WDT, no CP, MCLR
  __config  _IntRC_OSC_RB4EN & _IOSCFS_OFF & _WDT_OFF & _CP_OFF &
_MCLRE_ON
  #endif

;*****************************************************************
;                           MEMORIA RAM
; general purpose RAM
```

```
    UDATA                   ; area RAM
temp    res 1               ; temporaneo
Vtemp   res 1               ; temporaneo ADC
cntr    res 1               ; counter


 ;*****************************************************************
;=================================================================
;            DEFINIZIONE DI IMPIEGO DEI PORT
;
;PORTC map
;|  5  |  4  |  3  |  2  |  1  |  0  |
;|-----|-----|-----|-----|-----|-----|
;|segmf|segme|segmd|segmc|segmb|segma|
;
#define     segma  PORTC,0    ; segmenti f:a
#define     segmb  PORTC,1    ;
#define     segmc  PORTC,2    ;
#define     segmd  PORTC,3    ;
#define     segme  PORTC,4    ;
#define     segmf  PORTC,5    ;
```

```
;PORTB map
;|  5  |  4  |  3  |  2  |  1  |  0  |
;|-----|-----|-----|-----|-----|-----|
;|segmg|     |     | AN2 |GATE1|GATE2|
;
;#define           PORTB,0    ;
#define     GATE1  PORTB,1    ; gate MOSFET cifra decine
#define     GATE2  PORTB,2    ; gate MOSFET cifra unità
;#define           PORTB,3    ; MCLR
#define     segmdp PORTB,4    ; segmento dp
#define     segmg  PORTB,5    ; segmento g
;
;#####################################################################
;                          COSTANTI
;
#define  bit05k  TMR0,1 ; bit TMR0 per 512us
#define  bit1k   TMR0,2 ;             1024us
#define  bit2k   TMR0,3 ;             2048us
#define  bit4k   TMR0,4 ;             4096us
#define  bit8k   TMR0,5 ;             8192us
```

```
#define bittmr   bit4k  ; 4096us
tempo   EQU 0           ; numero di ccli tra le conversioni
                        ; 256 x 4096us = 1s

; limiti risultato ADC
 constant min = 0x1B
 constant max = 0x4C

;#########################################################################
;                          MACRO LOCALI
; comando gate MOSFET
UNITon   MACRO
        bsf GATE2
          ENDM
UNIToff  MACRO
        bcf GATE2
          ENDM
DECon   MACRO
        bsf GATE1
          ENDM
```

```
DECoff  MACRO
        bcf GATE1
          ENDM

; macro generali
 #include C:\PIC\Library\Baseline\compmacro.asm

;*****************************************************************
;                          RESET ENTRY
;
; Reset Vector
        CODE     0x00

; calibrazione oscillatore interno
        movwf   OSCCAL

; salto per lasciare spazio ai vettori indiretti per le subroutines
        pagesel start
        goto    start

;-----------------------------------------------------------------
```

```
; vettori subroutines
ADConv:                   ; conversione AD
        pagesel ADConv_r
        goto    ADConv_r
volttbl:                  ; lookup table per le tensioni
        pagesel volttbl_r
        goto    volttbl_r
segtbl:                   ; lookup table maschere dei segmenti
        pagesel segtbl_r
        goto    segtbl_r

;*****************************************************************
;                        MAIN PROGRAM
Main     CODE

start:
; disabilita comparatori per liberare la funzione digitale
        bcf     CM1CON0, C1ON
        bcf     CM2CON0, C2ON

; disabilita T0CKI da RB5, prescaler 1:256 al Timer0
```

```
          ;      b'11010111'
          ;        1-------     GPWU abilitato
          ;        -1------     GPPU abilitato
          ;        --0-----     clock interno
          ;        ---1----     falling
          ;        ----0---     prescaler al Timer0
          ;        -----111     1:256
        movlw   b'11010111'
        OPTION

        clrf    PORTB     ; clear latch del port
        clrf    PORTC

; Tutti i port utili come out
        movlw   0
        tris    PORTB
        tris    PORTC

; ADC per misura riferimento interno
                ; 00------  no ingressi analogici
                ; --11----  INTOSC/4
```

```
                ; ----11--  riferimento interno
                ; ------0-  GO/Done
                ; -------1  abilita ADC
        movlw   b'00111101'
        movwf   ADCON0

        movlw   tempo     ; contatore cadenza conversioni
        movwf   cntr

        clrf    TMR0      ; inizializza timer

;-----------------------------------------------------------------------
; ciclo display
; digit unità
displp  btfss   bittmr     ; 2.048ms ?
         goto   displp     ; no - attesa
        movf    Vtemp,w    ; si - carica risultato conversione
        andlw   0x0F       ; solo nibble basso
        pagesel segtbl
        call    segtbl     ; lookup table
        pagesel $
```

```
        movwf   temp        ; salva in temporaneo
        movwf   PORTC       ; passa al port segm f:a
        btfsc   temp,6      ; comanda segmento g
         bsf    segmg
        btfss   temp,6
         bcf    segmg
        UNITon              ; cifra unità accesa
dslp_1  btfsc   bittmr      ; 2.048ms ?
         goto   dslp_1      ; no - attesa
dslp_2  UNIToff             ; si - spegni unità
; digit decine
; swap Vtemp in W, nibble basso<->nibble alto
        swapf   Vtemp,w
        andlw   0x0F        ; solo nibble basso
        pagesel segtbl
        call    segtbl      ; lookup table
        pagesel $
        movwf   temp
        movwf   PORTC
        btfsc   temp,6
         bsf     segmg
```

```
        btfss   temp,6
         bcf     segmg
        bsf     segmdp      ; aggiungi punto decimale
        DECon               ; accendi decine

; conversione AD
        decfsz  cntr,f      ; acquisisci solo ogni tempo * bittmr
         goto   dslp_3      ; 256 * 4ms = 1s circa
        movlw   tempo       ; ricarica contatore
        movwf   cntr
        pagesel ADConv
        call    ADConv      ; conversione AD
        movf    ADRES,w
        movwf   Vtemp
        incf    Vtemp,f     ; correzione +1
        pagesel volttbl
        call    volttbl     ; tabella per valore in volt
        pagesel $
        movwf   Vtemp       ; salva per elaborazioni successive

; fine tempo ?
```

```
dslp_3  btfss   bittmr      ; 2.048ms ?
         goto   dslp_3      ; no - attesa
        DECoff              ; si - spegni decine
        bcf     segmdp      ; spegni punto decimale
        goto    displp      ; loop


TABLES  CODE  0x200
;#####################################################################
;             AREA TABELLE E SUBROUTINES in PAGINA1
;---------------------------------------------------------------------
; lookup table per convertire ADC da hex in volt
volttbl_r:
   CFLJB   Vtemp, min, aderL  ; se ADC<minimo, errore LL
   CFLJA   Vtemp, max, aderH  ; se ADC>massimo, errore HH
   movlw   min                ; parti da 0 nella tabella
   subwf   Vtemp,w
   addwf   PCL,f              ; punta PC
;               #     ADC
   retlw   0x57  ;0    1B
   retlw   0x55  ;1    1C
```

```
retlw   0x53  ;2   1D
retlw   0x51  ;3   1E
retlw   0x50  ;4   1F
retlw   0x48  ;5   20
retlw   0x47  ;6   21
retlw   0x45  ;7   22
retlw   0x44  ;8   23
retlw   0x43  ;9   24
retlw   0x42  ;10  25
retlw   0x40  ;11  26
retlw   0x39  ;12  27
retlw   0x38  ;13  28
retlw   0x37  ;14  29
retlw   0x37  ;15  2A
retlw   0x36  ;16  2B
retlw   0x35  ;17  2C
retlw   0x34  ;18  2D
retlw   0x33  ;19  2E
retlw   0x33  ;20  2F
retlw   0x32  ;21  30
retlw   0x31  ;22  31
```

```
retlw   0x31  ;23  32
retlw   0x30  ;24  33
retlw   0x30  ;25  34
retlw   0x29  ;26  35
retlw   0x28  ;27  36
retlw   0x28  ;28  37
retlw   0x27  ;29  38
retlw   0x27  ;30  39
retlw   0x26  ;31  3A
retlw   0x26  ;32  3B
retlw   0x26  ;33  3C
retlw   0x25  ;34  3D
retlw   0x25  ;35  3E
retlw   0x24  ;36  3F
retlw   0x24  ;37  40
retlw   0x24  ;38  41
retlw   0x23  ;39  42
retlw   0x23  ;40  43
retlw   0x23  ;41  44
retlw   0x22  ;42  45
retlw   0x22  ;43  46
```

```
   retlw   0x22  ;44  47
   retlw   0x21  ;45  48
   retlw   0x21  ;46  49
   retlw   0x21  ;47  4A
   retlw   0x20  ;48  4B
   retlw   0x20  ;49  4C

aderL retlw  0xAA   ; display LL
aderH retlw  0xBB   ; display HH

;-----------------------------------------------------------------------
; segment data table - display catodo comune
segtbl_r
        andlw  0x0F          ; solo nibble basso
        addwf  PCL,f         ; punta PC
        retlw  b'00111111'   ; "0" -|-|F|E|D|C|B|A
        retlw  b'00000110'   ; "1" -|-|-|-|-|C|B|-
        retlw  b'01011011'   ; "2" -|G|-|E|D|-|B|A
        retlw  b'01001111'   ; "3" -|G|-|-|D|C|B|A
        retlw  b'01100110'   ; "4" -|G|F|-|-|C|B|-
        retlw  b'01101101'   ; "5" -|G|F|-|D|C|-|A
        retlw  b'01111101'   ; "6" -|G|F|E|D|C|-|A
        retlw  b'00000111'   ; "7" -|-|-|-|-|C|B|A
        retlw  b'01111111'   ; "8" -|G|F|E|D|C|B|A
        retlw  b'01101111'   ; "9" -|G|F|-|D|C|B|A
        retlw  b'00111000'   ; "L" -|-|F|E|D|-|-|- exit per Ah
        retlw  b'01110110'   ; "H" -|G|F|E|-|C|B|- exit per Bh

;-----------------------------------------------------------------------
; Misura tensione
ADConv_r:
; attesa riduzione rumore
        DLY10US
; avvia conversione
        bsf     ADCON0,GO
adlp    btfsc   ADCON0,NOT_DONE
         goto   adlp
        retlw   0

;*****************************************************************
;                            THE END
        END
```