Esercitazioni PIC Baseline

11A – Il modulo ADC

Lo scopo dell' esercitazione è capire come utilizzare il modulo convertitore Analogico/Digitale.

L' acquisizione di segnali analogici da parte di un dispositivo digitale richiede la conversione di una grandezza continua in una discontinua. Questa funzione è richiesta perchè la maggior parte dei sensori di grandezze fisiche (temperatura, umidità, peso, accelerazione, intensità luminosa, ecc.) rende valori in tensione variabili in certo range, mentre il processore deve lavorare su dati di tipo digitale.
L' operazione di conversione è effettuata da circuiti chiamati convertitori **Analogico-Digitali** (ADC - *Analog to Digital Converter*), che possono appartenere a molte tipologie costruttive.

Nei microcontroller si trovano comunemente moduli ADC del genere ad approssimazione successive, con una risoluzione che va da 8 a 14 bit. Con "risoluzione" si intende che la tensione in ingresso, che varia tra un minimo ed un massimo continui, sarà resa in uscita dalla conversione in forma binaria, con un determinato numero di cifre; maggiore è questo numero, maggiore sarà la precisione con cui il valore digitale si avvicina a quello analogico.
Un ADC a 10 bit significa che ha la capacità di rilevare 1.024 (2^{10}) valori analogici distinti; un ADC a 8 bit rende 256 (2^{8}) valori.

Il range della tensione in ingresso può variare tra Vss e Vdd, in modo da non danneggiare il circuito di ingresso del pin; se si devono convertire tensioni di valori al di fuori di questa gamma (tensioni inferiori a Vss, ovvero tensioni negative o superiori a Vdd) occorre utilizzare dei circuiti esterni (partitori, condizionatori, amplificatori) per adeguare la tensione in ingresso ai limiti del microcontroller.

1. **I pin utilizzati come ingressi del modulo ADC NON possono essere usati per altre funzioni**.

2. Nei PIC, la filosofia costruttiva è quella che tende al minimo consumo e questo è dato dai pin configurati come analogici. Questo fa si che **nei default del POR l' impostazione analogica è quella pre fissata per tutti i pin che la supportano**.

   Necessitando di altre funzioni condivise sul pin, come quella di I/O digitale, occorre disabilitare da programma la funzione analogica.

La tecnica di conversione richiede la carica di un condensatore interno, di cui si rileva, con un comparatore e un contatore binario, il tempo di scarica, valutato con un clock che può essere derivato dall'oscillatore principale. Il contenuto del contatore è il numero che viene restituito al completamento della conversione. Questo momento è segnalato da un bit apposito nel registro di

controllo dell' ADC.
Il convertitore AD ha un certo consumo e, per la politica della minimizzazione dell' energia assorbita dal chip, è spento al default e richiede il settaggio di un bit per entrare in funzione.
Solitamente sono disponibili più canali di ingresso del modulo ADC, che sono selezionabili uno alla volta col registro di controllo.
Solo alcuni Baseline integrano il convertitore:

PIC	Pin	Canali ADC
10F220	6	2
10F222	6	2
12F510	8	3
16F506	14	3
16F526	14	3
16F527	20	8
16F570	20	8

Come si vede, più pin ha il dispositivo, più ingressi analogici possono essere resi disponibili.

Il modulo ADC dei Baseline è a 8 bit ed ha una struttura semplificata rispetto alle altre famiglie. In particolare, non esiste la possibilità di usare una tensione di riferimento esterna per la conversione, che **ha come riferimenti la tensione di alimentazione Vss e Vdd**. Ne consegue che la misura effettuata dal convertitore varia con il variare della Vdd. Quindi questa, per assicurare una risultato corretto, deve essere quanto più possibile stabile e priva di rumore.
Il modulo ha un generatore interno a 0.6V che serve come riferimento assoluto ed il cui impiego vediamo nella esercitazione relativa alla Vdd.

La conversione rende un valore esadecimale che dipende dalla tensione misurata, dalla tensione di riferimento e dalla definizione del convertitore:

Risoluzione ADC / Tensione di riferimento = Lettura ADC / Tensione misurata

Per una conversione a 8 bit che usa la Vdd a 5V come riferimento, se la tensione in ingresso vale 2V:

256 / 5 = lettura ADC / 2.5

lettura ADC = (256 /5) * 2.5 = 128d -> 7Fh

Data la natura digitale del valore in uscita dalla conversione, una variazione del valore analogico in ingresso sarà rilevata solamente se supera la risoluzione del convertitore, ovvero il valore tra uno step ed il successivo. Questo vuol dire che variazioni della tensione in ingresso inferiori a :

Tensione di riferimento / Risoluzione ADC

non saranno rilevabili. Nel caso di una conversione a 8 bit che usa una Vdd di 5V come riferimento, si tratta di :

5 / 256 = 0.01953 V

Al di sotto di questo limite non ci saranno differenze nel risultato della conversione; per poter valutare variazioni minori di 19mV occorrerà un convertitore AD con una risoluzione maggiore.

Da quanto sopra dovrebbe essere chiaro che il risultato della conversione non è un valore "assolutamente" preciso, ma è una approssimazione, proprio per la stessa natura dell' operazione di conversione. A questo si sommano gli errori dovuti al convertitore e alle fluttuazioni della tensione di riferimento e di quella da misurare. Ne risulta che i bit meno significativi del risultato possono essere affetti da errore, la cui correzione solitamente si opera o non tenendone conto oppure utilizzando algoritmi di media (ad esempio arithmetic mean, olympic mean, ecc.) su diversi campionamenti successivi.

Il modulo ADC dei Baseline

Prendiamo in considerazione i **Baseline con 2 o 3 canali analogici**. I chip a più di 14 pin sono Baseline, come già accennato, un poco anomali, più simili ai Midrange.

Il controllo del modulo ADC per 10F220/2, 12F510 e 16F506/526 è demandato ad un SFR chiamato **`ADCON0`** (***ADc CONtrol register 0***) :

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-0	R/W-0
ANS1	ANS0	ADCS1	ADCS0	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

La funzione dei singoli bit è la seguente:

bit7:6 **ANS1:0** Selezione dei canali analogici di ingresso abilitati:

ANS1:0	Canale configurato come analogico
00	nessuno
01	AN2
10	AN0 e AN2
11	AN0, AN1, AN2

Questo indica due cose importanti:

1. se desideriamo che nessun pin sia collegato ad una funzione analogica, occorre impostare da programma i due bit7:6 di ADCON0 a livello 0. Come indicato prima, per default al POR il valore di questi bit è 11, ovvero tutti i pin disponibili come analogica sono impostati come tali.
   Va ancora osservato che i bit ANS1:0 a livello 1 attivano la funzione analogica sui pin indipendentemente da qualsiasi altra impostazione sia stata definita prima. L' unica eccezione è quella degli ingressi dei comparatori, con cui il modulo ADC condivide la funzione.
   Quindi, se vogliamo usare i pin AN0, AN1, AN2 come I/O digitali occorrerà sicuramente disabilitare la funzione analogica ADC:

```
; disabilita ADC
      bcf       ADCON0,ANS1
      bcf       ADCON0,ANS2
```

2. Possono essere selezionate alcune combinazioni di ingressi: nel caso indicato (16F506), AN1 o AN2 da soli non possono essere attivi, così pure qualsiasi altra combinazione non presente in tabella. Questo comporta che nell' assegnazione delle funzioni ai pin si debba verificare la situazione del microcontroller che si intende usare. Ad esempio:

```
; abilita AN0 e AN2
      bsf       ADCON0,ANS1
      bcf       ADCON0,ANS2
```

Ovviamente, per i PIC10F220/222 sono selezionabili solo due canali.

bit5:4 **ADCS1:0** Selezione del clock per la conversione:

ADCS1:0	Clock
00	*Fosc/16*
01	*Fosc/8*
10	*Fosc/4*
11	*INTOSC/4*

La velocità con cui è eseguita la conversione dipende dal clock che è applicato al modulo ADC e questo è selezionato dai bit indicati. Si tratta di utilizzare o un clock generato con componenti esterni (LP,XT,HS,EC ExtRC) oppure l'oscillatore interno. Nel caso in cui l' oscillatore interno sia anche quello principale, la scelta **`Fosc/4`** e **`INTOSC/4`** sono la medesima cosa.

Il modulo ADC, abbiamo detto, è del tipo ad approssimazioni successive e richiede, per completare la conversione, un tempo di **13** ***Tad***, dove ***Tad*** è pari al ciclo di istruzione. Quindi, per 4MHz di clock, Tad=1us. Questo vuol dire che la conversione è completata in **13us**.
Se il clock è **8MHz**, ***Tad*=500ns** e la conversione occupa **6.5us**.
Dove è possibile utilizzare un clock esterno maggiore, dato che ***Tad*** deve essere compreso tra 500ns

e 50us, occorre utilizzare un fattore di divisone diverso da /4. Così, con un clock esterno di 20MHz, occorre una divisione /16:

***Fosc*= 20MHz *Fosc/4* = 5MHz -> *Tad* = 200ns**

che è troppo basso. Allora ricorriamo a:

***Fosc/16* = 1.25MHz -> *Tad* = 800ns**

che rientra nei limiti previsti. Qui, la conversione è completata in 10.4us.
Da notare che l' aumento del clock principale può non ridurre il tempo di conversione: il modulo ADC è un "modulo", ovvero un elemento che lavora separatamente dal processore, a cui è sincronizzato dal clock, ma che richiede tempi propri per il completamento delle sue attività.
In ogni caso, non è consigliabile scendere al di sotto dei limiti indicati dal folio dati per ***Tad*** nel tentativo di abbreviare il tempo della conversione, perchè il risultato potrebbe essere errato e in modo non sistematico.

bit3:2 **CHS1:0** Selezione del canale in ingresso:

ADCS1:0	Clock conversione
00	AN0
01	AN1
10	AN2
11	riferimento interno 0.6V

Abbiamo detto che il modulo ADC dispone di più canali in ingresso; questi corrispondono ai pin AN0,1,2 e sono multiplexati all' ingresso del modulo. Dato che, ovviamente, è possibile convertire solo un segnale alla volta, la scelta del canale di ingresso viene fatta con i bit3:2 del registro di controllo.

La scelta 11 non corrisponde ad alcun canale, ma collega l'ingresso del modulo ADC con il riferimento interno che è usato per la conversione. A che serve misurare il valore di una tensione di riferimento, quindi assai stabile e di valore ben definito? Questo diventa importante in apparati alimentati a batteria senza stabilizzatori intermedi (sappiamo che i PIC possono funzionare senza problemi tra 2 e 5.5V); se la tensione scende a mano a mano che la batteria si esaurisce, varia la Vdd che è utilizzata come riferimento superiore per la conversione, che risulterà errata. Campionando la tensione di riferimento diventa possibile correggere i risultati della conversione. Inoltre serve a valutare in via indiretta la tensione di alimentazione Vdd, come vedremo più avanti.

bit1 **GO/DONE** Bit di avvio conversione/flag di fine conversione:

GO/DONE	Scrittura	Lettura
1	avvia conversione	

0		conversione completata

Questo bit ha una funzione un poco particolare, trattandosi sia di un bit di comando, sia di un flag di segnalazione:

- in scrittura, se posto a 1, avvia la conversione
- in lettura, diventa 0 quando la conversione è completata

Dato che i Baseline non gestiscono interrupt, occorre verificare in polling il flag per individuare il completamento della conversione:

```
; avvia conversione
      bsf       ADCON0,GO
; attesa fine conversione
eclp  btfsc     ADCON0,NOT_DONE
       goto     eclp
....
```

Osserviamo che, in apparenza, il **bsf** e il **btfsc** fanno riferimento a due bit diversi. Però, il bit/flag di avvio della conversione è definito nel file ***nomeprocesore.inc*** con diversi alias:

```
;----- ADCON0 Bits ----------------------------------------------------
ADON         EQU H'0000'
GO_NOT_DONE  EQU H'0001'

GO           EQU H'0001'
CHS0         EQU H'0002'
CHS1         EQU H'0003'
ADCS0        EQU H'0004'
ADCS1        EQU H'0005'
ANS0         EQU H'0006'
ANS1         EQU H'0007'

NOT_DONE     EQU H'0001'
```

Può essere, quindi, più significativo (anche se non certo obbligatorio) usare una label o l'altra a seconda che si tratti dell'azionamento del bit di comando o dell'analisi del flag di fine conversione.

bit0 **ADON** Bit di accensione del modulo ADC:

GO/DONE	ADC
0	disabilitato
1	abilitato

Il modulo ADC può essere "acceso" e "spento" da programma. Questo è utile per ridurre il consumo di energia, abilitando il modulo solo quando serve.

 **Avvertenza** : spegnere il modulo ADC con il bit **ADON** è una cosa diversa dalla disabilitazione delle funzioni analogiche associate ai pin.
Portando a 0 il bit **ADON**, il modulo ADC è disabilitato, ma le impostazioni fissate con gli altri bit del registro **ADCON0** restano valide.
Quindi, per eliminare dai pin le funzioni analogiche non richieste, si deve agire sui bit **ADC1:0**, come indicato prima.

Per ricapitolare, la gestione della conversione AD comporta le seguenti fasi:

1. definire quali pin sono da destinare all' ingresso analogico con **ADC1:0**
2. definire il clock della conversione con **ADCS1:0**
3. selezionare il canale che si vuole leggere con **CHS1:0**
4. abilitare il modulo con **ADON**
5. Aggiungiamo anche un certo ritardo per permettere la stabilizzazione e il sample della tensione di ingresso da parte del convertitore
6. avviare la conversione con **GO/DONE**
7. attendere la fine della conversione testando il bit

In istruzioni:

```
; setup ADC
      movlw b'01110010'
;             01------  AN2
;             --11----  INTOSC/4
;             ----10--  CH2
;             ------1-  Abilita ADC
      movwf   ADCON0
; attesa per stabilizzazione
      delay10us         ; almeno 10us
; avvia conversione
      bsf     ADCON0,GO
; attesa fine conversione
eclp  btfsc    ADCON0,NOT_DONE
       goto    eclp
```

Alternativamente, il setup può essere formulato come catena di OR:

```
; setup ADC
      movlw   b'01'<<ANS0|b'11'<<ADCS0|b'10'<<CHS0|1<<ADON
      movwf   ADCON0
```

che, però, è un po' laborioso.

Il risultato ottenuto dalla conversione si trova in un apposito registro **ADRES**. Il valore contenuto indica la tensione misurata:

Vin = ADRES * Vref /256

Nel nostro caso, Vref è la Vdd. Si vede bene la necessità di avere una tensione di alimentazione stabilizzata quanto più si desidera un risultato corretto della conversione; non importa tanto il valore assoluto, dato che, conoscendolo, si potrà valutare correttamente il risultato, quanto l' assenza di oscillazioni e ripple che falserebbero la misura in modo difficilmente aggiustabile.

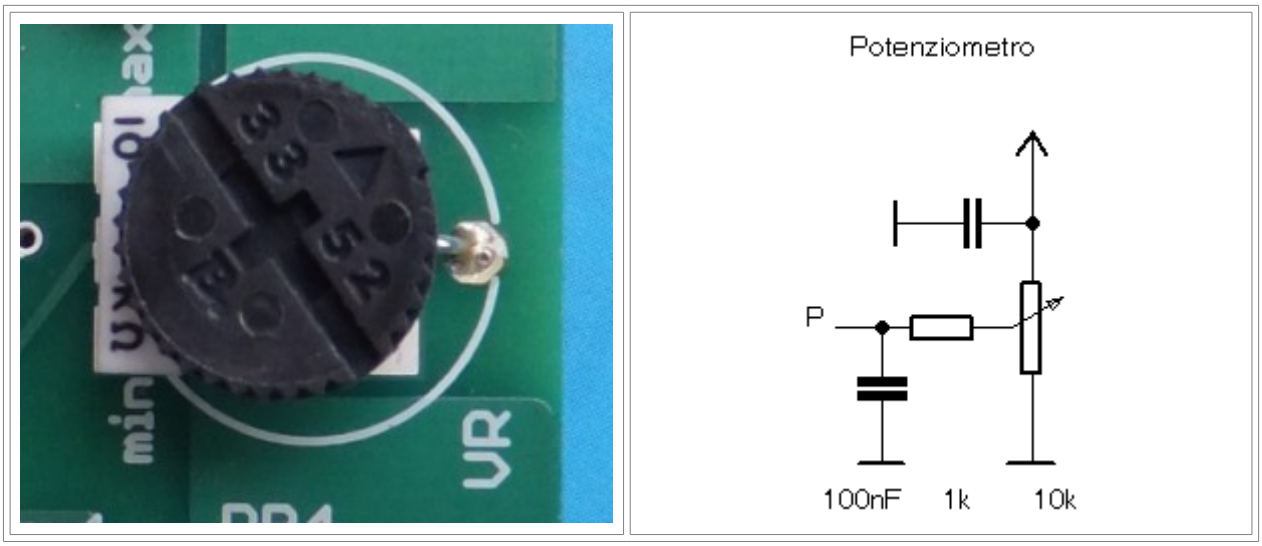
Alla fine della conversione, se occorre ridurre il consumo energetico, di disabiliterà il modulo ADC, azzerando il bit **ADON**.

Uso dell' ADC per valutare una tensione.

Iniziamo a verificare il funzionamento dell' ADC su un PIC minimale. Nella famiglia 10F2xx, troviamo 10F222 e 10F224 che dispongono del convertitore con due canali in ingresso.
Possiamo realizzare un "calibro" che indica se una tensione è al di fuori di un certo raggio.
Data la bassa quantità di I/O disponibili, usiamo due LED per l' indicazione:

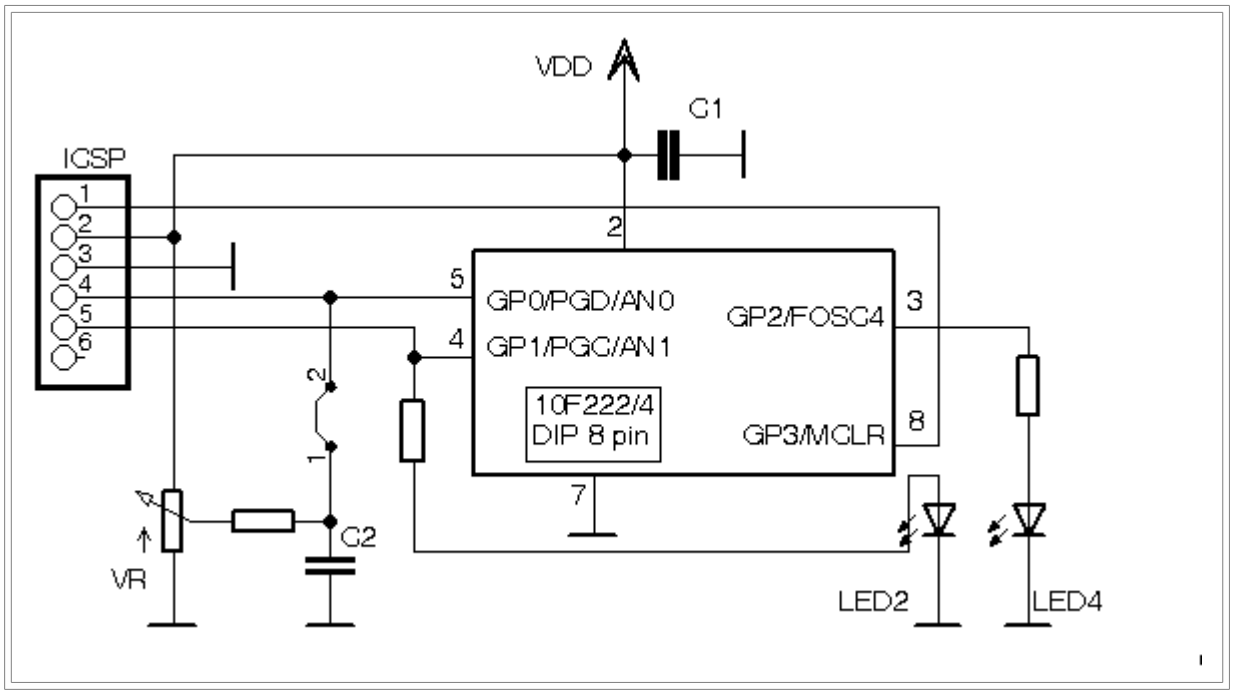
LED1	LED4	Tensione
on	**off**	bassa
on	**on**	giusta
off	**on**	alta

Come fonte del segnale utilizziamo il potenziometro che la scheda **LPCuB** mette a disposizione. Questo è collegato tra Vdd e Vss e quindi il cursore scorre su tutti i valori compresi in questa gamma.



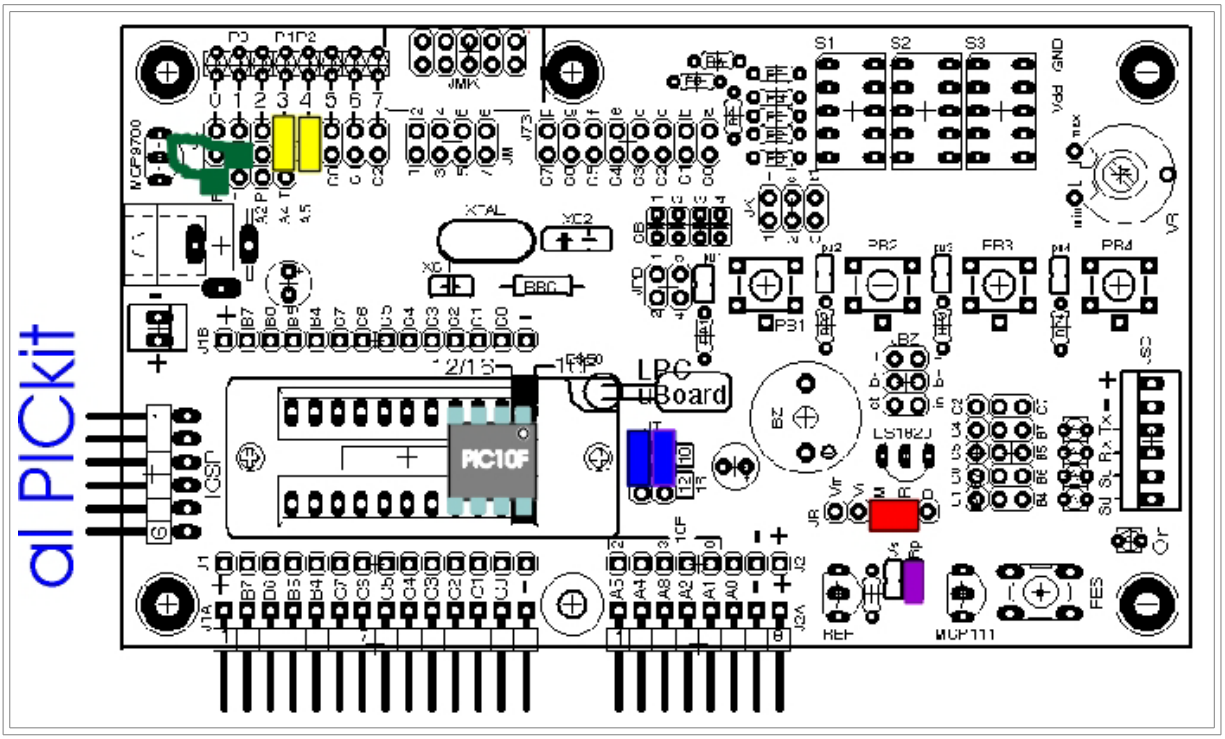
Il cursore ha in serie una resistenza di valore limitato (da 100ohm a 1k) con funzione di protezione del pin di ingresso del microcontroller da eventuali fenomeni elettrostatici e che, assieme ad un condensatore da 100nf costituisce un filtro passa basso per limitare eventuali disturbi impulsivi.

Il circuito elettrico è semplice:



La tensione in ingresso al canale **`AN0`** viene derivata dal potenziometro VR.

Sulla LPCuB bastano pochi jumper:



I jumper "gialli" collegano i LED.
Il jumper "verde" collega il cursore del potenziometro.

Il jumper "verde" va tolto durante la programmazione del chip, dato che **AN0** è condiviso su **GP0** assieme alla linea di scambio dei dati **PGD** che viene usata dalla connessione ICSP.

Inoltre, **il Pickit va staccato durante il funzionamento** per evitare di avere problemi a causa dei sui circuiti interni che verrebbero a trovarsi in parallelo all' ingresso analogico.

Il pulsante di Reset non viene usato nell' esercitazione e può rimanere collegato o scollegato a scelta.
Se è collegato, ricordarsi sempre di non premerlo durante la fase di programmazione, dato che l' ingresso MCLR condivide la tensione di programmazione Vpp.

Il programma.

Possiamo fissare i limiti di tensione per i quali devono accendersi o spegnersi i LED, usando una forma percentuale sul valore della tensione di riferimento:

```
#####################################################################
; COSTANTI
; limiti di tensione
min EQU (255/3)     ; limite minimo 33% di Vdd  (1.66V circa per Vdd=5V)
max EQU ((255/3)*2) ; limite massimo 66% di Vdd (3.33V circa per Vdd=5V)
```

Possiamo comunque esprime in ogni altro modo valido i limiti e variarli a seconda dell'applicazione.
I PIC10F220/222 hanno un modulo ADC con un controllo maggiormente semplificato rispetto a quanto desritto sopra. In particolare, dato che possono operare solamente con il clock interno, non è necessaria la scelta el clock di conversione. Quindi ADCON0 si presenta così:

R/W-1	R/W-1	U-0	U-0	R/W-1	R/W-1	R/W-0	R/W-0
ANS1	ANS0	—	—	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

Dato il basso numero di pin disponibili, i canali analogici sono solo due e sono impostabili singolarmente:

ANS1:0	Funzione
00	GP1 e GP0 digitali
01	GP0 analogico, GP1 digitale
10	GP0 digitale, GP1 analogico
11	GP1 e GP0 analogici

Si ricorda che, per default al POR, entrambi gli ingressi sono attivati (ANS1:0 = 11) e che per accedere alla funzione digitale occorre portarli a 0.
I bit 5:4, usati per la selezione del clock di conversione, qui non hanno alcuna funzione (bit non implementati).
Per quanto riguarda la selezione del canale da misurare:

CHS1:0	Funzione
00	AN0/GP0
01	AN1/GP1
10	Vref 0.6V
11	Vref 0.6V

I bit di abilitazione e il bit/flag di start/fine conversione sono identici a quanto detto sopra.

In base a questo, configuriamo il registro di controllo del modulo ADC per avere l' ingresso AN0:

```
; configura ADC per AN0 e abilita
    movlw   b'01000001'
    movwf   ADCON0
```

Aggiungiamo una attesa di almeno 10us per permette la stabilizzazione delle impostazioni e la carica del condensatore interno di sample&hold sul canale analogico scelto ed avviamo la conversione:

```
; attesa stabilizzazione
     DLY10US

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

Quando il flag **NOT_DONE** va a livello 1 la conversione è conclusa.
Utilizziamo il risultato comparandolo con i limiti per gestire opportunamente i LED

Il sorgente può essere compilato per 10F220 e 10F222 (default).

Ruotando il potenziometro, per il primo terzo della corsa sarà acceso solo LED3; nella parte centrale saranno accesi LED3 e LED4. Per l'ultimo terzo sarà acceso solo LED4.

Se la tensione da misurare è superiore alla tensione di alimentazione del microcontroller, occorre inserire un **partitore** in modo che il valore a fine corsa del potenziometro non superi la Vdd.

T & t

Durante la scrittura di questo sorgente, nel taglia-e-incolla, è sfuggita la linea **radix dec** che cambia l'indicazione della base numerica. Ne è risultato che l'Assembler ha utilizzato la base hex, che è il default.
Successivamente, nella definizione dei valori dei limiti massimo e minimo, confidando nella onnipresente inserzione della richiesta di radice decimale, la cosa è stata espressa così:

```
; limiti di tensione
min EQU (255/3)     ; limite minimo 33% di Vdd
max EQU ((255/3)*2) ; limite massimo 66% di Vdd
```

Il risultato della compilazione è stato un atteso **BUILD SUCCEEDED**, con la generazione del file eseguibile, ma la solita attenzione al resto del messaggio ha rilevato una linea inattesa:

```
Warning[202] C:10A_10F.ASM 143 : Argument out of range. Least significant bits used.
```

Che cosa è successo?
il valore 255, in mancanza di radice decimale, è stato inteso come 255h e il risultato della formula relativa a **max** è stato 18Eh.

*(255h /3) *2 = C7h * 2 = 18Eh*

Questo supera il numero inseribile in 8 bit (larghezza del dato) e il compilatore ha utilizzato solo la parte bassa (8Eh), producendo l'eseguibile, ma avvertendo del problema. Il programma non avrebbe funzionato correttamente.
Nel file ***.lst*** troviamo la controparte della segnalazione:

```
Warning[202]: Argument out of range. Least significant bits used.
0015    0C8E     00141    movlw max
```

Inserendo la definizione della radice voluta, il problema viene corretto, dato che 255 sarebbe inteso come decimale:

*(255 / 3) *2 = * 2 = 170 -> AAh*

Nel file ***11A_10F.asm*** la riga di definizione della radice numerica è commentata, in modo che la compilazione ricrei la situazione di errore. Basterà togliere il **;** per correggere la situazione (ricordiamo ancora una volta che il contenuto delle linee di commento, che iniziano con il ; non viene considerato durante la compilazione).
Il caso ricorda che, anche se è stata specificata (o si pensa sia stata specificata...) una radice, è sempre buona norma indicare i numeri in modo corretto:

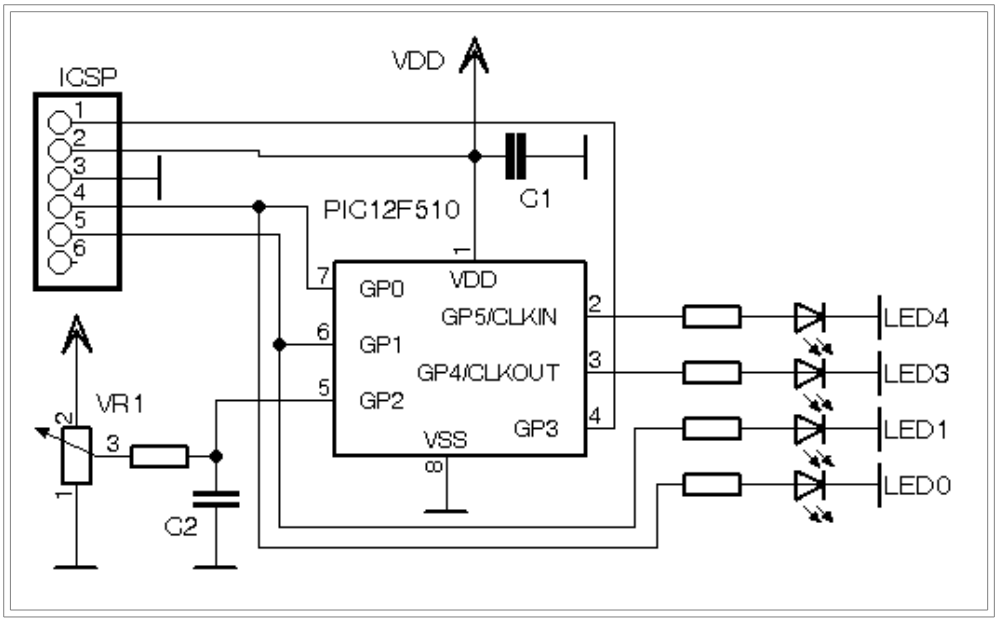
```
; limiti di tensione
min EQU (.255/3)     ; limite minimo 33% di Vdd
max EQU ((.255/3)*2) ; limite massimo 66% di Vdd
```

e, sopratutto, è assolutamente necessario leggere TUTTO il messaggio che MPASM invia alla fine della compilazione, non accontentandosi del solo **`BUILD SUCCEEDED`** , che può non essere sufficiente a confermare la correttezza del risultato.

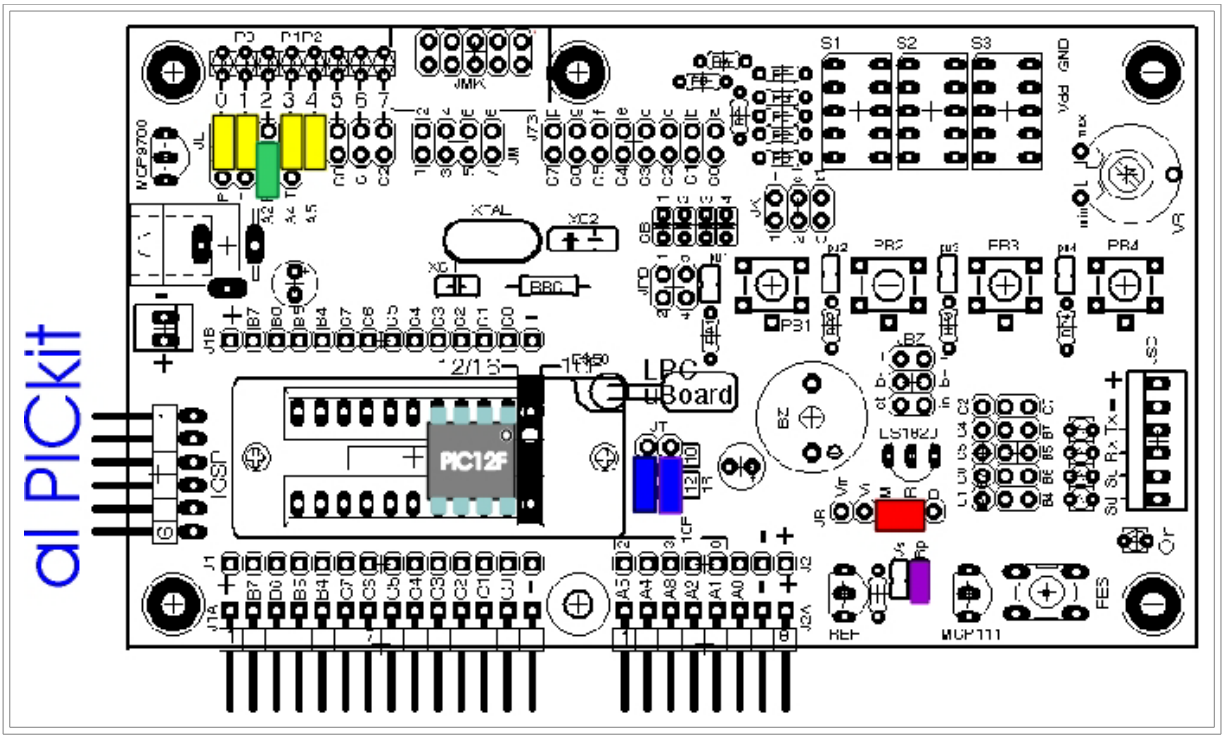
Visualizzare il dato risultante dalla conversione su 4 LED

Possiamo realizzare in pochi minuti un semplice esempio attinente al tema: **vogliamo visualizzare il risultato della conversione AD, in forma binaria, su alcuni LED**.

Come fonte del segnale utilizziamo ancora il potenziometro che la scheda **LPCuB** mette a disposizione e come processore usiamo un 12F510, con 6 I/O, il che permette di comandare 4 LED, che saranno accesi in proporzione al risultato della conversione.
Per avere 4 I/O disponibili come uscite, occorre utilizzare una sola analogica. Il chip permette di selezionare come singolo ingresso da dedicare all'ADC un solo pin, che è **`AN2`**/**`GP2`**:



Sulla **LPCuB**:



I jumper "gialli" collegano i LED a **GP0/1/4/5**.
Il jumper "verde" collega il cursore del potenziometro a **GP2**.

Il programma.

Abbiamo due problemi: il primo è quella che abbiamo già visto, ovvero allineare i LED saltando **GP3**, che non può essere usato come uscita, e anche **GP2** che utilizzato dall' ingresso analogico.
Il secondo riguarda la rappresentazione del risultato della conversione AD , che è su 8 bit, su soli 4 LED.
Questo si supera facilmente con alcune considerazioni; innanzitutto osserviamo che con 4 LED possiamo rappresentare numeri binari da 0 a F (da 0000 a 1111), mentre il numero rappresentato su 8 bit varia tra 0 e FF (da 00000000 a 11111111).
Possiamo usare allora solamente i 4 bit più alti, scartando il nibble basso; avremo una presentazione binaria del risultato pari a:

display = risultato / 16

Avendo a che fare con numeri binari, una divisione per 2 è realizzabile senza alcuna difficoltà con uno shift a destra. Dividere per 16 corrisponde a 4 shift successivi:

```
; divide risultato per 16
      movf        risultato, w
      andlw       0xF0                ; elimina nibble basso
      movwf       risultato
      clrc                            ; azzera carry per shift
      rrf         risultato,f
      rrf         risultato,f
      rrf         risultato,f
```

```
      rrf       risultato,f
```

Però la cosa si risolve con ben maggiore efficienza semplicemente con uno swap:

```
; divide risultato per 16
      swapf     risultato, w      ; scambia nibble alto<->nibble basso
      andlw     0x0F              ; elimina nibble alto
      movwf     risultato
```

Occorre poi escludere **GP2** e **GP3** e comandare la loro posto **GP4** e **GP5**:

```
; sposta bit 2-3 in 4-5
        btfsc   temp,2
         bsf    temp,4
        btfsc   temp,3
         bsf    temp,5
        movf    temp,w
        movwf   GPIO
```

Il dato viene passato direttamente al GPIO senza usare shadow in quanto tutti i bit sono scritti in una unica operazione.
Il Timer0 viene utilizzato per cadenzare una operazione ogni 130ms circa.

Il fatto di non considerare i bit più bassi del risultato della conversione rende il display dei LED molto stabile, anche se occorre un certo angolo di rotazione per avanzare o arretrare il contatore binario.

SWAPF

L' istruzione opera su un byte scambiando il nibble alto con quello basso.
Ad esempio:

```
       movlw      b'00001111'
      movwf      target             ; target = 00001111
      swapf      target,f           ; scambia nibble alto<->nibble
basso
; ora target vale 11110000
```

La conversione vera è propria rispecchia quanto detto in precedenza:

```
; setup ADC
      movlw b'01110010'
;             01------  AN2
;             --11----  INTOSC/4
;             ----10--  CH2
```

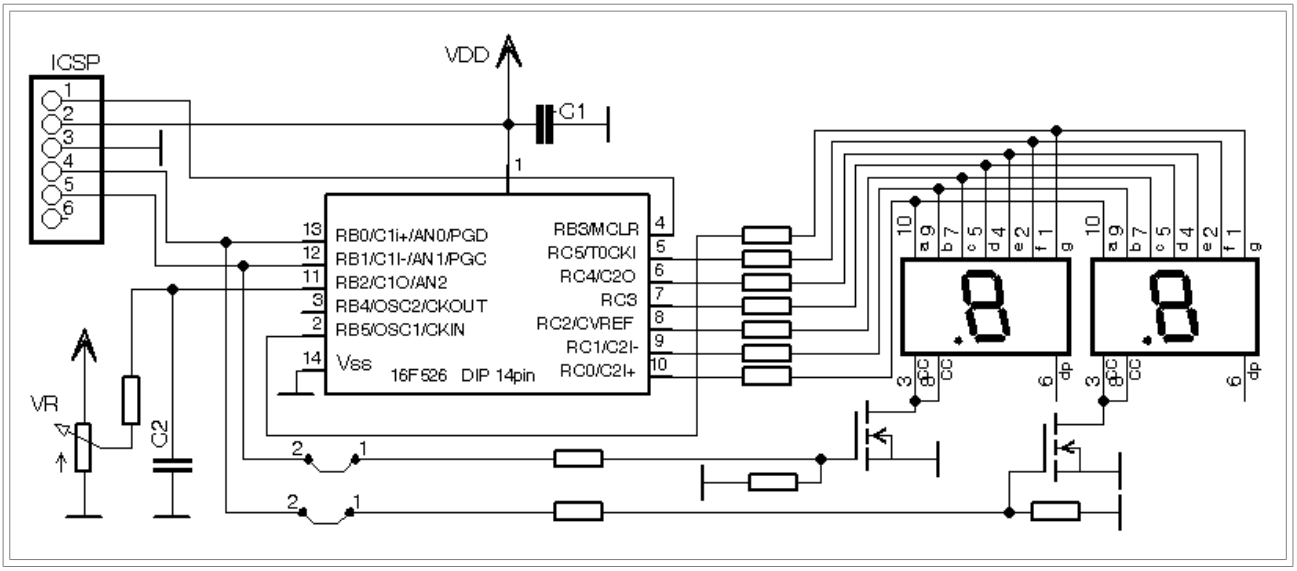
```
;             ------1-  Abilita ADC
      movwf   ADCON0
; avvia conversione
      bsf     ADCON0,GO
; attesa fine conversione
w lp  btfsc   ADCON0,NOT_DONE
       goto   wlp
; salva risultato conversione
      movf    ADRES,w
      movwf   risultato
```

Con le solite poche modifiche si potrà portare il sorgente su 16F506/526. Anche 10F220/2 hanno il modulo ADC, ma si avrebbero solo due I/O liberi.

Ruotando il potenziometro, si modificherà la cifra binaria presentata dai LED, con tutti spenti se il cursore è ruotato verso la Vss (min.) e tutti accesi se è ruotato verso la Vdd (max).

Lettura del dato analogico su 2 display a 7 segmenti

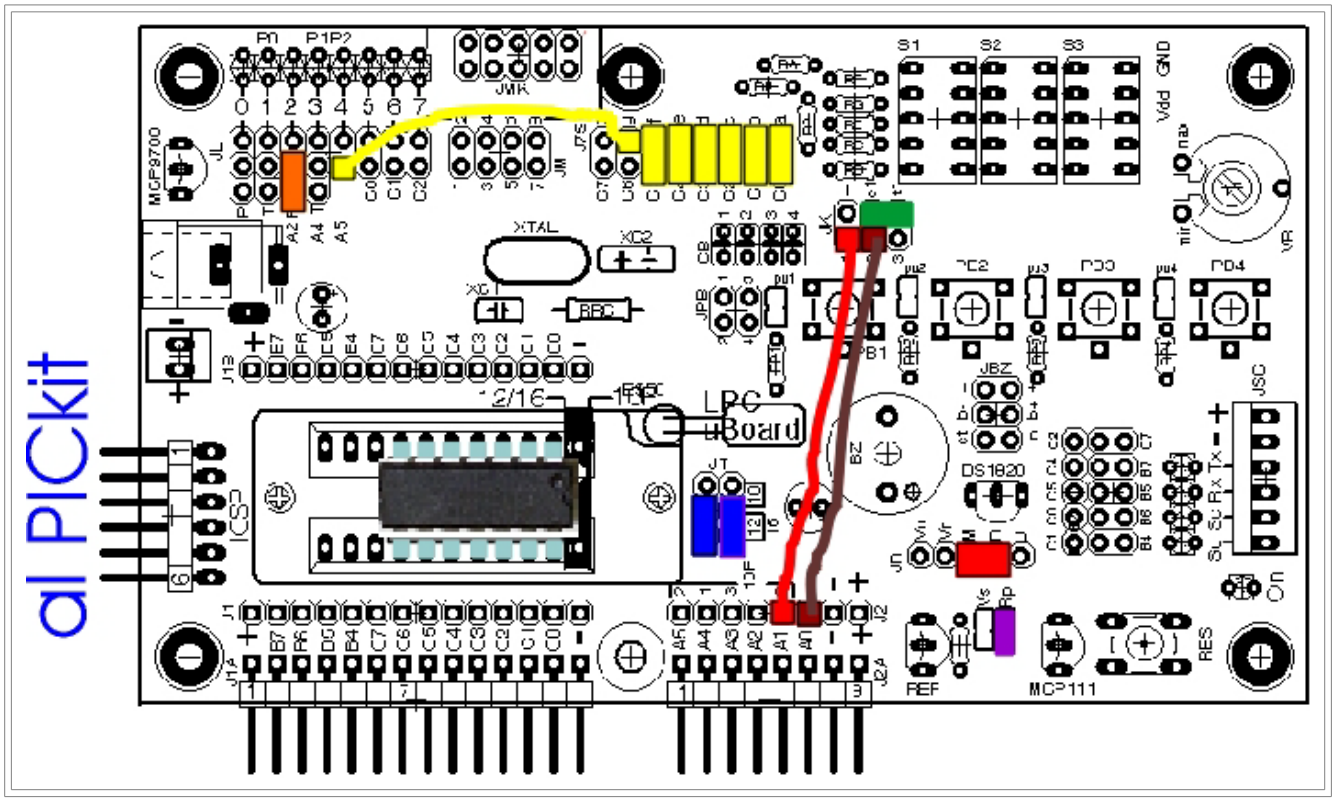
Usando un chip a 14 pin (16F506/526) possiamo avere la lettura del dato convertito sui display a 7 segmenti.



Disponendo di 2 display potremo rappresentare l' intero numero del risultato della conversione, che varia tra 00 e FFh.

I jumper in serie alle linee che si derivano da **RB0** e **RB1** indicano la necessità di scollegarle durante la programmazione on-board.

Sulla **LPCuB**:



I jumper "gialli" collegano i segmenti.
Il jumper "arancio" collega il cursore del potenziometro a **RB2**.
I jumper volanti "marrone" e "rosso" collegano gli I/O di comando dei gate dei MOSFET. Questi jumper saranno scollegati durante la programmazione del chip.
il jumper "verde" collega il gate del primo MOSFET.
I jumper del reset ("rosso" e "viola") non sono necessari, dato che il programma non prevede la funzione.

Il programma.

Molto semplice, utilizza vari elementi visti in precedenza e in particolare:

- il sistema di lookup table per comandare i segmenti
- la modalità multiplex per gestire le due cifre

Lo stato di un bit del Timer0 che conta free run viene usato per la cadenza del multiplex, nella quale è inserita la conversione AD.
Il dato ottenuto dalla conversione, sotto forma di byte esadecimale, viene passato, suddiviso in due nibble, direttamente alla lookup table per ottenere le cifre corrispondenti.
Ruotando il potenziometro, cambia la tensione applicata all' ingresso del modulo ADC e quindi l' indicazione del display che varia tra 00 e FFh.

E' normale uno sfarfallio della cifra delle unità, dato che non è inserita nel programma alcuna correzione delle unità: la conversione risente di qualsiasi minima variazione sia della Vdd, usata come riferimento, sia della tensione di ingresso, afflitta da disturbi dati dalle tensioni indotte dei vicini circuiti in cui scorrono le tensioni impulsive del multiplex, oltre alla normale precisione che è ±½LSB.
Per evitare lo sfarfallio si può ricorrere ad un campionamento più lento oppure all'inserimento di algoritmi di media di più valori.

11A_510F.asm

```
;*****************************************************************
; 11A_510.asm
;-----------------------------------------------------------------
;
;    Titolo        : Corso Assembly & C - Esercitazione 11A
;                    Risultato ADC su 4 LED
;
;    PIC           : 12F510
;    Supporto      : MPASM
;    Versione      : V.519-1.0
;    Data          : 01-05-2013
;    Ref. hardware :
;    Autore        :  afg
;
;-----------------------------------------------------------------
;
;  Impiego pin :
;  ----------------
;       12F510  @ 8 pin
;
;                   |__\/__|
;              Vdd -|1    8|- Vss
;              GP5 -|2    7|- GP0
;              GP4 -|3    6|- GP1
;         GP3/MCLR -|4    5|- GP2
;                   |______|
;
;    Vdd                      1: ++
;    GP5/OSC1/CLKIN           2: Out  LED4
;    GP4/OSC2                 3: Out  LED3
;    GP3/!MCLR/VPP            4: MCLR
;    GP2/T0CKI/AN2/C1OUT      5: ingresso analogico
;    GP1/ICSPCLK/AN1/C1IN-    6: Out  LED1
;    GP0/ICSPDAT/AN0/C1IN+    7: Out  LED0
;    Vss                      8: --
;
;*****************************************************************
;
        LIST      p=12F510
         #include <p12F510.inc>

        radix     dec

;*****************************************************************
;                           CONFIGURAZIONE
; oscillatore interno, 4MHz, no WDT, no CP, MCLR
  __config  _IntRC_OSC & _IOSCFS_OFF & _WDT_OFF & _CP_OFF & _MCLRE_ON

;*****************************************************************
;                           MEMORIA RAM
; general purpose RAM
        UDATA          ; inizio area RAM
temp    res 1          ; temporaneo
```

```
;*********************************************************************
;                            MACRO

  #include C:\PIC\Library\Baseline\basemacro.asm

;*********************************************************************
;=====================================================================
;            DEFINIZIONE DI IMPIEGO DEI PORT
;
;GPIO map
;|  5  |  4  |  3  |  2  |  1  |  0  |
;|-----|-----|-----|-----|-----|-----|
;|LED4 |LED3 |MCLR | AN2 |LED1 |LED0 |
;
;#define      GPIO,GP5    ; LED4
;#define      GPIO GP4    ; LED3
;#define      GPIO,GP3    ; MCLR
;#define      GPIO,GP2    ; AN2
;#define      GPIO,GP1    ; LED1
;#define      GPIO,GP0    ; LED0

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

; calibrazione oscillatore interno
        movwf   OSCCAL
        pagesel Init
        goto    Init

;*********************************************************************
;             TABELLE E SUBROUTINES IN PAGINA 0

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

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

Init:
; disabilita comparatori per liberare la funzione digitale
        bcf     CM1CON0, C1ON

; disabilita T0CKI da RB5, prescaler 1:256 al Timer0
         ;      b'11111111'
         ;        1-------     GPWU abilitato
```

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

        clrf    GPIO        ; clear latch del port

; Tutti i port utili come out
        movlw   0
        tris    GPIO

; configura ADC per AN2, INTOSC/4 e abilita
        movlw    b'01111011'
        movwf    ADCON0

        clrf     TMR0       ; inizializza timer

;-----------------------------------------------------------------
; ciclo lettura / display
; digit unità
displp  movf    TMR0,w      ; 65ms ?
        skpz                ; si - ADC
         goto   displp      ; no - attesa

; conversione AD
        pagesel ADConv
        call    ADConv
        pagesel $

; display risultato conversione
; allinea bit risultato con i GPIO disponibili
        swapf   ADRES,w     ; scambia nibbles in W
        andlw   b'00001111' ; solo bit3:0
        movwf   temp        ; salva

 ; sposta bit 2-3 in 4-5
        btfsc   temp,2
         bsf    temp,4
        btfsc   temp,3
         bsf    temp,5
        movf    temp,w
        movwf   GPIO

; attesa
dlp1    movf    TMR0,w      ; fine stato TMR0=0 ?
        bnz     displp      ; si - loop
         goto   dlp1        ; no - attesa

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

11A_10F.asm

```
;*****************************************************************
; 11A_10F.asm
;-----------------------------------------------------------------
;
;    Titolo        : Corso Assembly & C - Esercitazione 11A
;                    Impiego ADC
;
;    PIC           : 10F220/2
;    Supporto      : MPASM
;    Versione      : V.20x-1.0
;    Data          : 01-05-2013
;    Ref. hardware :
;    Autore        :  afg
;
;-----------------------------------------------------------------
;
;  Impiego pin :
;  ----------------
;     10F220/2 @ 8 pin DIP         10F220/2 @ 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: AN0
;    NC                        6:  nc
;    Vss                       7:  2: --
;    GP3/MCLR/VPP              8:  6:
;
;
;*****************************************************************
;=================================================================
;            DEFINIZIONE DI IMPIEGO DEI PORT
;
;GPIO map
; |  3  |  2  |  1  |  0  |
; |-----|-----|-----|-----|
; |     | LED4| LED3| AN0 |
;
;#define        GPIO,GP0    ;
#define    LED3 GPIO,GP1    ;
#define    LED4 GPIO,GP2    ;
;#define        GPIO,GP3    ;

;
;*****************************************************************
```

```
;#####################################################################
; scelta del processore
 #ifdef __10F222
        LIST      p=10F222
        #include <p10F222.inc>
 #endif
 #ifdef __10F224
        LIST      p=10F224
        #include <p10F224.inc>
 #endif

        ;radix dec   ;<<<--- da correggere !!!

;#####################################################################
;                         CONFIGURAZIONE
;
 ; No WDT, no CP, pin4=GP3
  __config  _IOFSCS_4MHZ & _MCPU_OFF & _WDT_OFF & _CP_OFF & _MCLRE_OFF

;#####################################################################
;                         MEMORIA RAM
;
;#####################################################################
;                         COSTANTI
; limiti di tensione
min EQU  (255/3)        ; limite minimo 33% di Vdd
max EQU  ((255/3)*2)    ; limite massimo 66% di Vdd

;#####################################################################
;                         MACRO
; DLY10US - Attesa 10us
; PIC Baseline con FOSC = 4MHz
DLY10US  MACRO
    goto  $+1           ; 5 x 2us
    goto  $+1
    goto  $+1
    goto  $+1
    goto  $+1
         ENDM

;#####################################################################
;=====================================================================
;                          RESET ENTRY
;
; Reset Vector
        ORG     0x00

 ; calibrazione oscillatore interno
        andlw   0xFE        ; assicura no FOSC4
        movwf   OSCCAL

; pre clear output
        clrf    GPIO

; disabilita T0CS per liberare GP2
; OPTION default  11111111
;                 1-------     !GPWU disable
```

```
;                -1------     !GPPU disable
;                --0-----     disabilita T0CS
;                ---1----     falling edge
;                ----1---     prescaler al WDT
;                -----111     rate 1:256
        movlw   b'11011111'
        option

; GP2:1 come out
        movlw   b'00001001'
        TRIS    GPIO

; configura ADC per AN0 e abilita
mainloop:
        movlw    b'01000001'
        movwf    ADCON0

; attesa stabilizzazione
        DLY10US

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

; compara risultato con i limiti
; limite minimo
        movlw    min           ; w=ADRES-min
        subwf    ADRES,w
        bnc      minexit       ; se C=0, ADRES<=min
; limite massimo
        movlw    max           ; w=ADRES-max
        subwf    ADRES,w       ; se C=1 max=>ADRES
        bc       maxexit
; ok - nel range voluto - gestione LED
inrange bsf      LED3          ; in range - entrambi i LED accesi
        bsf      LED4
        goto     mainloop
; al di sotto del minimo - gestione LED
minexit bcf      LED4          ; <= minimo - LED3 acceso
        bsf      LED3
        goto     mainloop
; al di sopra del massimo - gestione LED
maxexit bsf      LED4          ;>= massimo - LED4 acceso
        bcf      LED3
        goto     mainloop

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

11A_526.asm

```
;*****************************************************************
; 11A_526.asm
;-----------------------------------------------------------------
;
;    Titolo        : Corso Assembly & C - Esercitazione 11A
;                    Risultato ADC in hex 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: AN2
;    RB1/C1IN-/AN1/ICSPC    12: Out  gate unità
;    RB0/C1IN+/AN0/ICSPD    13: Out  gate decine
;    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
        CBLOCK 0x10          ; inizio area RAM
      temp                   ; temporaneo
        ENDC

 ;*****************************************************************
;=================================================================
;            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     GATE2  PORTB,0    ; gate MOSFET cifra unità
#define     GATE1  PORTB,1    ; gate MOSFET cifra decine
;#define           PORTB,2    ; AN2
;#define           PORTB,3    ; MCLR
;#define           PORTB,4    ;
#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

;#####################################################################
;                          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

;#####################################################################
;                          RESET ENTRY
;
; Reset Vector
        ORG     0x00

; calibrazione oscillatore interno
        movwf   OSCCAL
        goto    Main

;#####################################################################
;             TABELLE E SUBROUTINES IN PAGINA 0

; segment data table - display catodo comune
segtbl  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'01110111'   ; "A" -|G|F|E|-|C|B|A
        retlw  b'01111100'   ; "b" -|G|F|E|D|C|-|-
        retlw  b'00111001'   ; "C" -|-|F|E|D|-|-|A
        retlw  b'01011110'   ; "d" -|G|-|E|D|C|B|-
        retlw  b'01111001'   ; "E" -|G|F|E|D|-|-|A
        retlw  b'01110001'   ; "F" -|G|F|E|-|-|-|A

; Misura tensione
ADConv:
; attesa riduzione rumore
```

```
        DLY10US
; avvia conversione
        bsf     ADCON0,GO
adlp    btfsc   ADCON0,NOT_DONE
         goto   adlp
        retlw   0

;
;#####################################################################
;                       MAIN PROGRAM
Main:
; disabilita comparatori per liberare la funzione digitale
        bcf     CM1CON0, C1ON
        bcf     CM2CON0, C2ON

; disabilita T0CKI da RB5, prescaler 1:256 al Timer0
         ;      b'11111111'
         ;        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

; configura ADC per AN2, INTOSC/4 e abilita
        movlw    b'01111011'
        movwf    ADCON0

        clrf     TMR0      ; inizializza timer

;---------------------------------------------------------------------
; ciclo display
; digit unità
displp  btfss   bittmr     ; 4ms ?
         goto   displp     ; no - attesa
        movf    ADRES,w    ; si - carica risultato conversione
        andlw   0x0F       ; solo nibble basso
        call    segtbl     ; lookup table
        movwf   temp       ; salva in temporaneo
        movwf   PORTC      ; scrivi sul 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     ; 4ms ?
         goto   dslp_1     ; no - attesa
```

```
dslp_2  UNIToff             ; si - spegni unità
; digit decine
; swap ADRES in W, nibble basso<->nibble alto
        swapf   ADRES,w
        andlw   0x0f        ; solo nibble basso
        call    segtbl
        movwf   temp
        movwf   PORTC
        btfsc   temp,6
         bsf     segmg
        btfss   temp,6
         bcf     segmg
        DECon               ; accendi decine
        call    ADConv      ; conversione AD
dslp_3  btfss   bittmr      ; 4ms ?
         goto   dslp_3      ; no - attesa
        DECoff              ; si - spegni decine
        goto    displp      ; loop

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