Esercitazioni PIC Midrange

Inserto_e – il modulo ADC

Un buon numero di Midrange dispone del modulo ADC, ovvero del convertitore analogico-digitale integrato.

Lo scopo di questo modulo è quello di trasformare valori di tensione analogici in corrispondenti valori digitali. Ad esempio, l'uscita analogica di un sensore di temperatura o pressione o un generico valore di tensione potrà essere convertito in dato digitale per essere elaborato dal processore.

I numeri binari risultanti dalla conversione possono essere di lunghezza differente: nei PIC a 8 bit si parla di 8, 10 o 12bit; più bit contiene il risultato, maggiore è la risoluzione.


Alcuni Midrange datati hanno la risoluzione della conversione AD pari a 8bit.
Qui intendiamo parlare di dispositivi più recenti in cui **la risoluzione è 10bit**.

Questo vuol dire che l'uscita del convertitore sarà 0 per 0V di ingresso e 3FFh (1023 decimale) per una tensione di ingresso pari a Vdd o alla tensione di riferimento.


La tensione massima applicabile agli ingressi non deve essere inferiore alla Vss e non deve superare la Vdd

Dove è necessario misurare tensioni maggiori occorrerà interporre un partitore e per tensioni molto basse o negative si utilizzerà una adeguata interfaccia analogica.

La conversione è effettuata con il metodo delle approssimazioni successive, un bit alla volta, sotto il controllo di un **clock**: dalla sua frequenza dipende la durata della conversione.


Il tempo di completamento della conversione di un bit è indicato come *Tad*, ma per 10bit di risultato, il tempo complessivo sarà *11Tad* (1 *Tad* per l'avvio della conversione più 1 *Tad* per ogni bit).

Ad esempio, se il ***Tad*** è 2us, la conversione sarà completata in 22us.

Il clock necessario all'avanzamento della conversione è derivabile da due fonti:

- il clock del processore (***Fosc***). Essendo il clock primario bloccato durante la condizione di sleep, non è possibile usare questa selezione se si si intende far operare il modulo ADC in questa condizione;
- oppure un oscillatore RC interno dedicato (***Frc***) da circa 500kHz, che è attivo anche durante la condizione di sleep, permettendo così il completamento della conversione anche in questa modalità e rendendo l'evento una sorgente di wakeup, oltre che di interrupt.

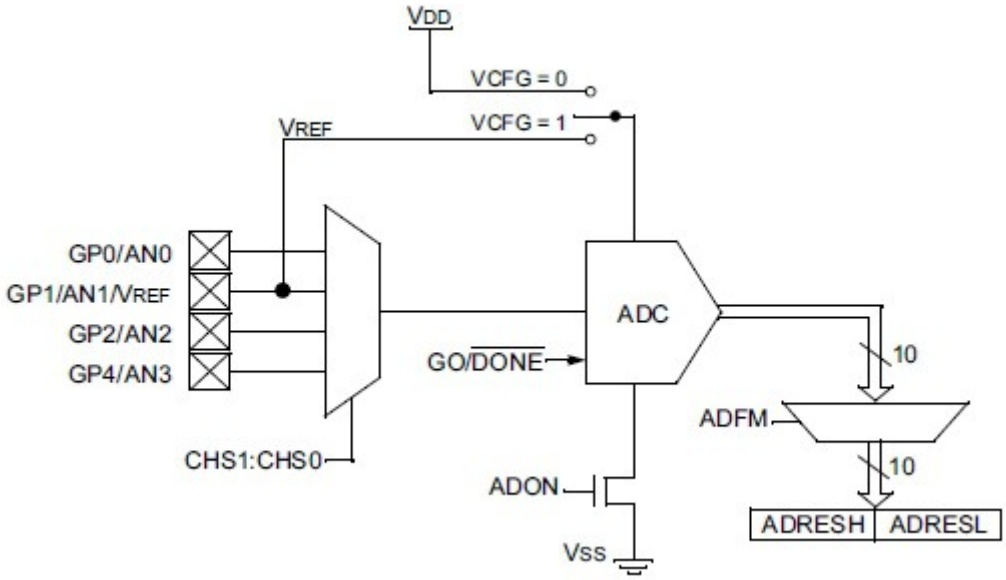
Un particolare: se si seleziona il clock ***Frc***, la conversione sarà avviata dopo l'esecuzione di una istruzione. Questo è necessario quando la conversione deve avvenire nella condizione di sleep.

Il convertitore, unico, dispone di più ingressi selezionabili da programma, il cui numero dipende dalla quantità di pin disponibili.

Il convertitore può usare come tensione di riferimento la ***Vdd*** (default) oppure una ***Vref*** esterna. La Tensione di riferimento stabilisce la gamma in cui opera la conversione.

Dentro il modulo ADC.

Schematicamente, consideriamo il modulo ADC del **12F675** che contiene gli elementi essenziali comuni a tutti questi PIC:



La selezione tra i pin utilizzabili come ingressi del modulo ADC è controllata da un multiplex, programmabile attraverso un registro specifico **`ANSEL`**:

ANSEL — ANALOG SELECT REGISTER (ADDRESS: 9Fh)

U-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-1	R/W-1
—	ADCS2	ADCS1	ADCS0	ANS3	ANS2	ANS1	ANS0
bit 7							bit 0

I bit **ANS3:0** se posti a 1 abilitano la funzione analogica **AN3:0** sul relativo pin.

Da tenere presente che, **mentre in componenti obsoleti, come i noti 16F87x, era possibile selezionare solamente combinazioni fisse di ingressi analogici, qui, con i bit ANS3:0 di ANSEL si può abilitare anche un solo ingresso analogico qualsiasi**, il che permette una flessibilità decisamente maggiore.

E' necessario considerare che al default questi bit sono posti a 1. Questo vuol dire che i pin sono configurati come ingressi dell'ADC. La funzione sovrappassa quella digitale. Quindi, volendo utilizzare i pin come semplici I/O digitali occorre disabilitare i corrispondenti bit in ANSEL, portandoli a 0 da programma.

Ovviamente, un pin utilizzato per la funzione analogica non potrà essere usato per altri scopi e viceversa. Esiste comunque la possibilità di variare durante l'esecuzione del programma la funzione attribuita ad un pin.

Tornando allo schema di principio del modulo ADC, gli ingressi fanno capo ad un multiplxer comandato dai bit **CHS1:0**. Questi bit consentono di selezionare quale canale è applicato all'ingresso del convertitore. Sono contenuti nel registro **ADCON0**:

ADCON0 — A/D CONTROL REGISTER (ADDRESS: 1Fh)

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

La logica è la seguente:

CHS1:0	Canale	Pin
00	AN0	GP0
01	AN1	GP1
10	AN2	GP2
11	AN3	GP4

Attenzione perchè la corrispondenza *label del canale- pin* è quella indicata: **AN3** non corrisponde a **GP3**, dato che **GP3/MCLR**, non può assumere una funzione analogica, ma a **GP4**.

Questi bit vanno comandati dal programma per selezionare il canale desiderato. Ovviamente, è possibile la conversione di un solo canale per volta.

I pin, per operare come ingressi analogici, devono avere il relativo bit del registro di direzione **TRIS** uguale a 1. Questo è il default all'accensione; se si opera in altri momenti occorrerà verificare che la condizione sia soddisfatta, altrimenti l'uso previsto non sarà possibile.

A differenza dei Baseline in cui il riferimento della conversione è generalmente fissato alla ***Vdd***, qui abbiamo la possibilità di applicare sia la ***Vdd*** che una ***Vref*** proveniente dall'esterno. Questo consente di separare la tensione di alimentazione da quella di fondo scala del convertitore. Il bit di controllo è **VCFG**:

VCFG	Vref
0	Vdd
1	esterna

Una ***Vref*** esterna, fornita da uno dei tanti riferimenti di tensione disponibili sul mercato, permetterà un diverso fondo scala e una maggior stabilità e precisione del risultato.

In particolare, l'uso della ***Vdd*** come riferimento della conversione, è consigliabile fino ad una risoluzione di 6-8bit. Oltre, per avere risultati significativi, un riferimento di tensione esterno è indispensabile. Questo è dovuto non tanto al fatto che la ***Vdd*** fornita da un alimentazione tradizionale non è precisa come valore: la precisione si potrà ottenere con una operazione di taratura. Piuttosto, essa è soggetta a variabilità e rumore che si potrebbero riflettere negativamente sulla precisione dei bit minori, rendendoli inutili.
Un riferimento esterno, poi, è indispensabile con una alimentazione a batterie, dove la ***Vdd*** varia con lo stato di carica.

Dallo schema di principio, osserviamo che il convertitore può venire acceso/spento con un bit **ADON**, che nel diagramma precedente vediamo agire su un MOSFET usato come interruttore di alimentazione del modulo ADC; al default il modulo è spento per minimizzare il consumo di corrente. Per renderlo operativo occorrerà portare a il bit **ADON** a livello 1 da programma.

ADON	ADC
0	Spento (default)
1	acceso

E' necessario considerare che l'accensione e lo spegnimento del modulo ADC NON ha niente a che fare con l'assegnazione della funzione di ingresso analogico ai pin.
Le due cose sono indipendenti: anche se il modulo è spento, gli ingressi possono essere analogici e viceversa.

In particolare, al default, il modulo è spento, ma tutti i pin disponibili sono configurati come ingressi analogici. Di conseguenza, i pin che non utilizziamo come analogici, ma di cui desideriamo la funzione digitale, devono essere gestiti da programma, come pure l'accensione del modulo ADC.

Per contro, possiamo aver acceso l'ADC, ma se non abbiamo configurato con **ANSEL** e **TRIS** la funzione analogica dei pin, non li potremo utilizzare.

Una volta configurato il modulo ADC per l'operatività voluta, selezionato l'ingresso desiderato e acceso il modulo stesso, la conversione non è avviata se non dopo aver portato a 1 il bit **GO_DONE**.

```
        banksel  ADCON0            ; avvia conversione
        bsf      ADCON0, GO_DONE
```

Questo è un **bit-flag**: al momento della sua scrittura a 1, si avvia la conversione; da questo punto in avanti dovrà essere letto per indicare la fine della conversione stessa che lo porterà a 0.

```
adtst   btfsc    ADCON0, GO_DONE   ; fine conversione ?
         goto    adtst
        ...
```

Questo non è, però, il solo modo per verificare la fine della conversione, in quanto il modulo ADC è sorgente di interruzione e quindi dispone di un flag **ADIF** nel registro **PIR1** che può essere testato o la fine conversione può essere gestita non in polling, ma come interrupt.

PIR1 – PERIPHERAL INTERRUPT REQUEST REGISTER 1 (ADDRESS: 0Ch)

U-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
—	ADIF	RCIF(1)	TXIF(1)	SSPIF(1)	CCP1IF(2)	TMR2IF(2)	TMR1IF
bit 7							bit 0

Un bit **ADIE** nel registro **PIE1** consente di abilitare l'interrupt di fine conversione:

PIE1 – PERIPHERAL INTERRUPT ENABLE REGISTER 1 (ADDRESS: 8Ch)

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	ADIE	RCIE(2)	TXIE(2)	SSPIE(2)	CCP1IE(1)	TMR2IE(1)	TMR1IE
bit 7							bit 0

Da osservare che l'interrupt di fine conversione è periferico e quindi richiede l'abilitazione di **PEIE** oltre che **GIE**.

La durata della conversione.

E' normale che si desideri ridurre al minimo il tempo di conversione.
Questo tempo dipende da diversi fattori, ma la conversione in sé dipende esenzialmente dal clock applicato, che è programmabile attraverso i bit **`ADC2:0`**, secondo questa tabella:

ADCS2:0	Frequenza	Tad
`000`	***Fosc/2***	**Tcy/2**
`001`	***Fosc/8***	**2 Tcy**
`010`	***Fosc/32***	**8 Tcy**
`011`	***Frc*** interno (500kHz circa)	**2-6us**
`100`	***Fosc/4***	**Tcy**
`101`	***Fosc/16***	**4 Tcy**
`110`	***Fosc/64***	**16 Tcy**
`111`	***Frc*** interno (500kHz circa)	**2-6us**

Dove ***Tcy*** è il ciclo istruzione, pari a ***Fosc/4***.

 **Nella scelta del clock, però, abbiamo un limite: *Tad* minimo deve essere maggiore o uguale a 1.6us.**

Questo è un limite fisso, dovuto alla struttura dell'hardware.
Da notare che non dipende dal clock della conversione, né da quello principale: una qualsiasi combinazione di clock primario e di clock di conversione sarà accettabile solo se il ***Tad*** è quello specificato.

In relazione a questo limite, possiamo dire che la scelta di ***Frc*** come oscillatore è sempre adeguata, qualsiasi sia il clock del processore (***Fosc***) in quanto è indipendente da questo. Il ***Tad***, con ***Frc***, è al minimo 2us e al massimo 6us (i fogli dati indica un valore tipico di 4us; la differenza rientra nella possibile tolleranza della frequenza di questo oscillatore).

Se, invece, scegliamo uno dei modi che utilizzano una frazione di ***Fosc***, dobbiamo tenere conto che il tempo ***Tad*** dipenderà dal valore di questa frequenza.
Precisiamo che si parla di ***Fosc***, cioè della frequenza dell'oscillatore, non di quella del ciclo di istruzione che è ***Fosc/4***.

Per contro, se ***Tad*** minori di 1.6us sono sconsigliati, ***Tad*** maggiori di 7us lo sono altrettanto: se ***Tad*** è troppo breve, la conversione non riesce a completarsi correttamente, mentre se è troppo lungo la tensione sul condensatore di campionamento può cadere prima che la conversione sia completata.

Possiamo tracciare una tabella indicativa:

Tad		Fosc			
Ciclo	**ADCS2:0**	**20MHz**	**5MHz**	**4MHz**	**1.25MHz**
2 Tcy	**000**	100ns	400ns	500ns	1.6us
4 Tcy	**100**	200ns	800us	1us	3.2us
8 Tcy	**001**	400ns	1.6us	2us	6.4us
16 Tcy	**101**	800ns	3.2us	4us	12.8us
32 Tcy	**010**	1.6us	6.4us	8us	25.6us
64 Tcy	**110**	3.2us	12.8us	16us	51.2us
Frc	**x11**	2-6us	2-6us	2-6us	2-6us

Le celle in rosso indicano valori di ***Tad*** troppo bassi, che violano il minimo indicato, mentre quelle in grigio sono valori troppo alti e pertanto sconsigliati.

A seconda della ***Fosc*** occorrerà scegliere una diversa combinazione dei bit **ADCS**.

 Ovviamente, dato che **in sleep l'oscillatore primario è spento**, si potrà utilizzare solamente ***Frc***, tenendo presente che i fogli dati consigliano di completare la conversione in sleep, pena un possibile risultato affetto da errore.

Come esempio, prendiamo il caso del comune **clock a 4MHz**: per mantenerci entro i limiti indicati, si sceglierà ***8Tcy*** o ***16Tcy***, ovvero un ***Tad*** pari a 2us o 4us.
Oppure potremo usare ***Frc***, scelta adeguata per qualsiasi ***Fosc***.

Perchè, allora, non usare sempre ***Frc***?
Semplicemente perchè, nel caso esemplificato, ***Frc*** rende un ***Tad*** medio di **4us**, mentre scegliendo ***2Tcy*** rende **2us**, dimezzando il tempo di conversione.

Nel caso in cui avessimo un oscillatore esterno a **20MHz** la scelta è compresa tra ***32Tcy*** e ***64Tcy*** , con ***Tad*** di 1.6-3.2us o ***Frc*** che rende un ***Tad*** medio di 4us.

Si potrebbe, con questo, dichiarare concluso il problema del tempo di conversione, ma non è così.

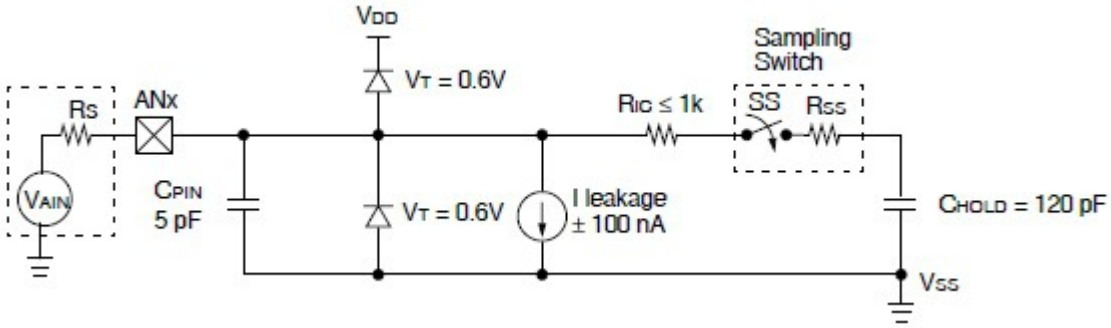
Se abbiamo un ***Tad*** di 2us, la conversione è completata in ***11Tad***, ovvero 22us, ma questo riguarda solamente il passaggio dal valore analogico presente all'ingresso del modulo a quello digitale alla sua uscita. L'operazione completa della conversione comprende altre azioni.

Selezione degli ingressi e tempo di acquisizione.

Un solo pin alla volta può essere collegato al convertitore e la selezione si effettua con i bit **CHS1:0** di **ADCON0**:

CHS1:0	Pin
00	AN0
01	AN1
10	AN2
11	AN3

E' utile dare uno sguardo a come è realizzato il circuito che sta tra il pin e il il convertitore:



Il pin, per ragioni costruttive, ha una propria capacità ***Cpin***, ma di valore molto basso e quindi solitamente trascurabile.

Il segnale in ingresso è limitato da una coppia di diodi in modo che non possa superare i limiti di ***Vdd+0.6V*** e ***Vss-0.6V***. Questo protegge il circuito successivo dall'applicazione di tensioni improprie. Va inteso che questa protezione non sostituisce un sistema di protezione esterno da applicare nei casi in cui il segnale di ingresso possa essere fonte di disturbi, sovra e sotto tensioni o cariche statiche.

Il sistema assorbe una corrente di perdita ***Ileakage*** molto bassa e quindi solitamente trascurabile.

La tensione applicata all'ingresso carica un condensatore di sample/hold ***Chold*** da 120pF attraverso l'impedenza della sorgente della tensione e le resistenze interne del circuito integrato (***Ric*** e ***Rss***). Quando viene selezionato un canale, il condensatore viene collegato alla tensione sul pin ed è richiesto un tempo per la carica di questo condensatore (sample - tempo di acquisizione). All'avvio della conversione il condensatore viene staccato dalla sorgente (hold).
Se il tempo di lavoro dell'ADC è entro i limiti previsti, la tensione ai capi del condensatore non si abbassa durante la conversione.

La massima impedenza consigliata per la sorgente esterna è di 10kohm; se il valore è maggiore se ne dovrà tenere conto nel tempo di acquisizione, dato che viene allungato. In generale, minore è l'impedenza della sorgente, minore sarà il tempo di acquisizione e viceversa.

In sostanza, quindi, la misura non è effettuata direttamente sulla tensione in ingresso, ma ai capi del condensatore Chold che è stato caricato a questa tensione. Dato che la corrente di carica è limitata

dalle varie impedenze in serie, prima quella della sorgente, il condensatore richiede un certo tempo per caricarsi.
Si rende necessario, quindi, un tempo di attesa detto tempo di acquisizione, ***Taq***.

Da un punto di vista numerico, è dato da:

$$Tacq = Tamp + Tc + Tcoff$$

dove

- ***Tamp*** è il tempo di stabilizzazione del circuito, pari a 2us
- ***Tc*** è il tempo di carica del condensatore, che dipende dalle impedenze interne del circuito e da quella della sorgente
- ***Tcoff*** dipende dalla temperatura che ha un andamento descrivibile come ***[(Temp - 25°C) (0.05 ms/°C***)], ma la sua aggiunta va considerata solo per temperature ambiente maggiori di 25°C (caso comunque facilmente superabile alle nostre latitudini, ovunque se il circuito è posto all'interno di un contenitore chiuso).
  Nel caso di una ***Temp = 50°C*** assume un valore di ***1.25us***

I fogli dati riportano esempi di calcoli del ***Tacq*** necessario per una situazione media in cui la temperatura al massimo raggiunge i 50°C, con una sorgente di impedenza massima di 10kohm.

Il ***Tc*** è calcolato come:

$$Tc = -Chold * (Rs + Ric + R_{SS}) \ln(1/2047)$$

Sostituendo i valori tipici:

$$Tc = -(120\ pF)(10kohm + 1kohm + 7kohm) \ln(1/2047)$$

è pari a ***16.47us***.

Ne risulta che ***Tacq*** è uguale a:

$$Tacq = Tamp + Tc + Tcoff = 2 + 1.25 + 16.47 = 19.72us.$$

arrotondato a ***20us***.
Le equazioni sottintendono un errore massimo di ***1/2LSb*** calcolato sui 1024 step della conversione.

Questo ***Taq=20us*** è il tempo tipico necessario di attesa tra la selezione di un canale di ingresso e l'avvio della conversione.

Non rispettando questa attesa minima, il risultato della conversione sarà falsato.

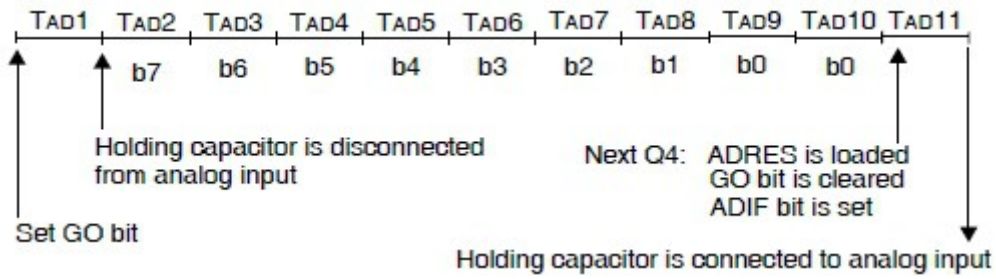
In istruzioni, ad esempio:

```
; tempo di acquisizione - 20us @ 4MHz

        movlw   0x06
        movwf   d1
Delay20us_0
        decfsz  d1, f
        goto    Delay20us_0
        nop
```

Tempo necessario alla conversione.

Ricapitolando in forma grafica, il ciclo della conversione è, schematicamente, il seguente:



- la conversione si avvia portando il bit **GO_DONE** a 1
- dopo 1 ***Tad*** il condensatore di hold è staccato dal pin di ingresso
- 7 ***Tad*** seguono per la conversione di un bit ciascuno
- durante l'undicesimo ***Tad*** il risultato viene salvato negli appositi registri **ADRES**, il flag **GO_DONE** viene portato a 1 e il flag di interrupt viene settato

alla fine dell' undicesimo ***Tad*** il condensatore di holding è ricollegato all'ingresso.

Se il ***Taq*** è necessario all'acquisizione del valore di tensione da convertire, occorre considerare che l'ADC richiede a sua volta un certo tempo per trasformare il valore analogico in uno digitale. Si tratta del tempo di conversione.

$$Tconv = Taq + Tcv$$

Tcv è il tempo di conversione pari a ***11Tad,*** che, per un minimo ***Tad*** di 1.6us rende:

$$Tcv = 11 * 1.6 = 17.6us$$

Quindi, la durata complessiva della conversione è:

$$Tconv = Taq + Tcv = 20 + 17.6 = 37.6us$$

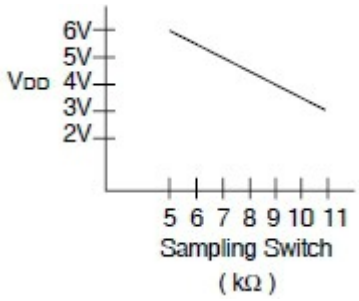
Questa è la durata minima di una operazione di acquisizione analogico/digitale.

 **Da notare che, se vengono ripetute successive conversioni occorre aggiungere un ritardo di *2Tad* tra la fine di una e l'avvio della successiva.**

Una interessante osservazione riguarda **l'impedenza della sorgente**: le resistenze interne al chip sono parametri non modificabili dall'utente, mentre se l'impedenza della sorgente si riduce, si riduce anche il ***Taq***; ad esempio, con una sorgente da 50ohm, il ***Taq*** diventa 10.61us.

Questo giustifica l'uso di buffer a bassa impedenza di uscita tra la sorgente del segnale e il convertitore.

Inoltre, si devono considerare anche altri fattori.



Mentre la tensione usata per il riferimento della conversione non influisce sui tempi di conversione, la tensione di alimentazione sì.

In particolare, riducendo la Vdd si ha un aumento della resistenza del sampling switch R_{SS}., come si vede dal diagramma a lato. In pratica, questo significa che sarà necessario un tempo di acquisizione maggiore al diminuire della tensione di alimentazione.

Si dovrà, quindi tenere conto del fatto che **la conversione complessivamente risulterà più lenta a basse tensioni di alimentazione**, anche ricordando quanto detto a proposito del rapporto tra il massimo clock applicabile e la stessa tensione di alimentazione.

Dove l'impedenza della sorgente del segnale analogico è alta (>1kohm), si comincia a notare l'influenza della corrente assorbita dalla sorgente per caricare il condensatore di campionamento.

Se la tensione in misura non cambia troppo rapidamente, può essere utile, come misura minimale, inserire un condensatore da 0,1 mF sull'ingresso analogico. Questo condensatore si carica alla tensione analogica e fornire l'istantanea corrente necessaria per caricare il condensatore interno. L'applicazione di un buffer realizzato con un amplificatore operazionale è comunque la soluzione migliore.

 **Da osservare che il tempo reale necessario per una conversione comprende anche le necessarie manipolazioni dei bit di controllo e il salvataggio dei dati.**

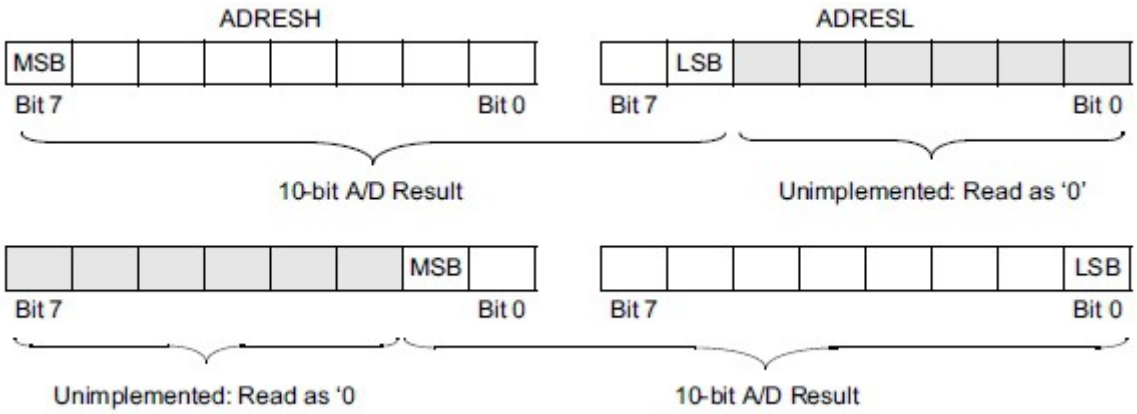
Il risultato della conversione.

Dato che il convertitore rende un risultato a 10bit, occorrono due bytes per contenerlo.

Si tratta dei registri **`ADRESH:ADRESL`**, che si trovano allo stesso indirizzo relativo, ma su due banchi diversi:

	1Dh		9Dh
ADRESH(2)	1Eh	ADRESL(2)	9Eh
ADCON0(2)	1Fh	ANSEL(2)	9Fh
	20h		A0h

I due bytes offrono uno spazio di 16bit in cui il risultato della conversione può essere salvato in due modi, selezionabili con bit **`ADFM`** di **`ADCON0`**:



Nel primo caso abbiamo quella che è definita come ***Left Justified***, ovvero giustificazione a sinistra, ottenibile portando a 0 il bit **ADFM**.
Questa soluzione vede i primi 8 bit più significativi accumulati in **ADRESH**; questo permette con immediatezza di considerare il risultato della conversione come se fosse a 8bit, escludendo **ADRESL** con i due bit meno significativi.

Nel secondo caso abbiamo ***Right Justified***, ovvero giustificazione a destra, dove i due bit più significativi sono in **ADRESH** e i rimanenti in **ADRESL**, per operazioni su tutti i dieci bit.

La scelta tra i due modi dipende da come si deve trattare il risultato della conversione.

La conversione in pratica.

La prima cosa necessaria è quella di configurare correttamente il modulo ADC:

1. Configurare i pin analogici di ingresso desiderati (registro **ANSEL**), la loro direzione (registro **TRIS**) e la sorgente di riferimento nel registro **ADCON0**
2. Selezionare il canale di ingresso da convertire (**ADCON0**)
3. Selezionare il clock come visto sopra (**ADCON0**)
4. Configurare il modo di presentazione del risultato della conversione (**ADCON0**)
5. Accendere il modulo ADC (**ADCON0**)
6. Configurare l'interrupt se richiesto, cancellando **ADIF** (in **PIR1**)e settando **ADIE** (in **PIE1**) e gli interruttori generali **GIE** e **PEIE** (in **INTCON**)
7. Attendere il tempo di acquisizione
8. Avviare la conversione portando a 1 il bit **GO_DONE** (**ADCON0**)
9. attendere la fine della conversione controllando in polling **GO_DONE** a zero oppure **ADIF** a 1 o lasciare il controllo alla gestione dell'interrupt, se attivato
10. leggere il risultato della conversione in **ADRES**
11. se usato l'interrupt, cancellare il flag **ADIF**
12. se si ripete a ciclo stretto una successiva conversione, attendere ***2Tad***.

Dal punto di vista delle istruzioni, una inizializzazione dopo il POR, sfruttando i default, dove i pin

sono settati come ingressi e come analogici :

```
   banksel  ADCON0
   movlw    0xC1          ; FRC, ADC on, AN0, Right just.
   movwf    ADCON0
; se richiesto
   banksel  PIE1
   bsf      PIE1, ADIE    ; Abilita interrupt ADC
   bsf      INTCON, PEIE  ; Abilita interrupt periferico
   bsf      INTCON, GIE   ; Abilita interrupt generale
;
; Attesa pari al tempo di acquisizione calcolato
;
   banksel  ADCON0
   bsf      ADCON0, GO_DONE   ; avvia conversione

; test in polling su GO_DONE
adtst:
   btfsc    ADCON0, GO_DONE
    goto    adtst
; oppure su ADIF
adtst:
   btfss    PIR1, ADIF
    goto    adtst
   bcf      PIR1, ADIF
```

Se è attivo l'interrupt si gestirà la cosa nella relativa routine, ricordando sempre di cancellare il flag **ADIF** una volta identificato.
Se l'inizializzazione avviene in altri momenti diversi occorre verificare anche il coretto settaggio dei bit di **ANSEL** e di **TRIS**.
L'**ANSLEx** relativo all'**ANx** deve essere a 1 per disporre della funzione e pure il **TRISx** relativo dovrà essere a 1.

Il risultato potrà essere salvato in RAM o conservato in **ADRES**, tenendo presente però che una successiva conversione cancellerà il risultato della precedente.

```
; salva risultato conversione - little endian
   banksel  ADRESL
   movf     ADRESL,w
   banksel  targetram
   movwf    targetram
   banksel  ADRESH
   movf     ADRESH,w
   banksel  targetram
   movwf    targetram+1
```

I due bytes **ADRES**, se il modulo ADC non viene usato, possono essere impiegati come RAM generica.

ADC e sleep.

Il modulo convertitore AD può operare durante sleep se la sorgente di clock scelta è ***Frc***.
Se la sorgente del clock è diversa, sleep provoca l'interruzione della conversione in corso; il modulo ADC è disattivato, anche se il bit **ADON** resta a 1.

Se abbiamo scelto Frc come clock, l'ADC attende un'istruzione prima di iniziare la conversione, così da modo all'istruzione **sleep** di essere eseguita, eliminando così gran parte del rumore di commutazione dalla conversione.
Quando la conversione è completa, il bit **GO_DONE** viene azzerato e il risultato è caricato in **ADRESH:L**.
Se l'interrupt dell'ADC è stato abilitato, si genera un waleup.

Se l'interrupt non è abilitato, il modulo ADC viene spento, anche se il bit **ADON** rimane settato.

Altri PIC.

Quanto abbiamo detto vale per il 12F675 e analoghi a 8 pin, ma è una guida di base anche per i PIC maggiori.

Il numero di pin superiore consente innanzitutto di avere un numero altrettanto maggiore di in ingressi analogici. Ad esempio:

PIC	Pin	AN
12F675	**8**	**4**
16F688	**14**	**8**
16F690	**20**	**14**

E' evidente che l'aumento richiederà un numero maggiore di registri di controllo.

PIC	FSR
12F675	**ANSEL** **ADCON0**
16F688	**ANSEL** **ADCON0** **ADCON1**
16F690	**ANSEL** **ANSELH** **ADCON0** **ADCON1**

Fino a 8 ingressi, un byte di controllo **`ANSEL`** è sufficiente per abilitare/disabilitare la funzione analogica.

ANSEL – ANALOG SELECT REGISTER (ADDRESS: 11Eh)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0
bit 7							bit 0

Per 14 ingressi occorreranno due registri, chiamati **`ANSEL`** e `ANSELH`, che si trovano nello stesso banco:

ANSELH – ANALOG SELECT HIGH REGISTER (ADDRESS: 11Fh)

U-0	U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	—	ANS11	ANS10	ANS9	ANS8
bit 7							bit 0

Come visto prima, si può abilitare a piacere qualsiasi combinazione di ingressi analogici o digitali. Per default tutti i bit sono a 1, quindi è abilitata la funzione analogica.

I controlli principali si trovano sempre in **ADCON0**. Per **16F688**:

ADCON0 – A/D CONTROL REGISTER (ADDRESS: 1Fh)

R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	VCFG	—	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

I tre bit **CHS2:0** consentono di selezionare gli 8 canali, con la logica intuitiva **CHS2:0=000 -> AN0** e **CHS2:0=111 - > AN7**.

Altrettanto per **16F690**, dove si hanno **CHS3:0** per selezionare i **14 AN**.

ADCON0 – A/D CONTROL REGISTER (ADDRESS: 1Fh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	VCFG	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

Osserviamo come Microchip mantenga, dove possibile, sempre la stessa disposizione dei bit, il che facilita il lavoro dell'utilizzatore.

Ad **ADCON0** si associa **ADCON1**, che contiene i bit di controllo del clock:

ADCON1 – A/D CONTROL REGISTER 1 (ADDRESS: 9Fh)

U-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	ADCS2	ADCS1	ADCS0	—	—	—	—
bit 7							bit 0

I bit **ADCS** hanno le stesse funzioni viste in precedenza, come pure tutti gli altri bit dei vari registri di controllo.

In particolare, tutto quanto riguarda i tempi di conversione ed acquisizione, le caratteristiche degli ingressi analogici e la procedura di conversione sono quelle descritti in precedenza.

L'unica differenza sensibile riguarda la possibilità offerta da alcuni moduli ADC di leggere anche la tensione di riferimento del comparatore e la tensione del bandgap reference integrato. Ciò avviene, ad esempio, in **16F690**, dove le selezioni effettuate dai bit **CHS3:0** sono queste:

CHS3:0	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011
In	AN0	AN1	AN2	AN3	AN4	AN5	AN6	AN7	AN8	AN9	AN10	AN11
CHS3:0	1100			1101			1110-1111					
In	CVref			VP6			Non utilizzabili					

In sostanza, gli ingressi analogici esterni dai pin sono 12, mentre CVref e VP6 sono tensioni interne la cui conversione non interessa i pin.

CVref è la tensione programmata sul riferimento di tensione del comparatore.

Va specificato che in questi chip la tensione CVref del comparatore non può essere usata come Vref della conversione AD, ma solo misurata.

VP6 è il bangap reference interno a 0.6V la cui valutazione consente di effettuare misure indirette sulla tensione di alimentazione.
(vedere anche http://www.microcontroller.it/Tutorials/PIC18/vddevaluate.htm)

Risoluzione della conversione.

Con "risoluzione" si intende il minimo valore significativo del risultato, ovvero la tensione di ingresso corrispondente ad 1bit in uscita (*Vstep*).

Per un numero n di bit pari a 10 la conversione rende 2^n livelli, ovvero 2^{10} (da 0 a 1023), con 1023 (2^n -1) punti di soglia.

Matematicamente è data da:

$$Vstep = Vref / 1024$$

Ad esempio, per una ***Vref=5V***, un a conversione a **10bit** avrà una risoluzione di:

$$Vstep = 5 / 1024 = 0.0048828\ V\ per\ bit$$

Dove occorre una risoluzione maggiore sarà necessario impiegare un convertitore con un numero maggiore di bit in uscita.

Ad un valore di tensione in ingresso *Vin* corrisponderà una uscita del convertitore (***ADC***) multipla di quanto sopra calcolato.

$$Vin = ADC * Vref/ 2^n = ADC * Vstep$$

Quindi, ad esempio, se, con i dati del caso esemplificato sopra, il valore reso è 78h (120 dec):

$$Vin = 120 * 0.0048828 = 0.585V$$

Al contrario, si potrà calcolare il valore in uscita dalla conversione in base alla tensione in ingresso:

$$ADC = Vin * 2^n / Vref = Vin /Vstep$$

Quindi, una tensione di ingresso di 2.5V corrisponderà ad un valore digitale di:

$$ADC = 2.5 / 0.0048828 = 512 \rightarrow 200h$$

Accuratezza della conversione.

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.

Nonostante ciò, la ricerca della massima precisione ottenibile può essere una richiesta dell'appliacazione.

La precisione assoluta specificata per il convertitore AD comprende la somma di tutti i contributi per errore di quantizzazione, errore integrale, errore differenziale, errore fondo scala, errore di offset, e monotonia.
Essa è definita come la deviazione massima da una transizione reale rispetto a una transizione ideale per qualsiasi codice in uscita.
L'errore integrale, differenziale e di guadagno del convertitore AD viene specificato in un massimo di ± ***1 LSB*** per ***Vref=5V*** (param. A03).

Gli errori fissi possono essere calibrati su uno specifico sistema.
L'errore di linearità si riferisce all'uniformità dei codice e dipende dal convertitore stesso; errori di linearità non possono essere calibrati al di fuori da uno specifico sistema.

Però, nella maggior parte dei casi in cui la conversione non è soddisfacente, la cosa dipende da altri fattori esterni, come :

- la stabilità della tensione di riferimento, la sua precisione, l'assenza di ripple e disturbi su questa tensione, tra cui il rumore generato dalla parte digitale del sistema e da circuiti accessori (PWM, relais, ecc),
- l'impedenza della sorgente di riferimento e di quella in conversione
- la presenza di ripple e rumore sul segnale in conversione che non sono stati considerati in fase di progetto.
- la scorretta considerazione dei tempi necessari alla acquisizione ed alla conversione è causa di risultati scadenti.

La giusta considerazione della necessità di una qualche forma di condizionamento e filtraggio del segnale in ingresso, soprattutto se variabile, aggiunto ad una cura nel separare l'area analogica da quella digitale ed in particolare le uscite di potenza rispetto agli ingressi di misura, è fattore indispensabile di un buon progetto che richieda conversioni AD.

Nei sistemi in cui la frequenza dispositivo è bassa, l'uso del clock Frc è preferibile.
Nei sistemi in cui il dispositivo entra in modalità sleep dopo l'inizio della conversione, la Frc è richiesta come sorgente di clock. Da considerare anche che, essendo in sleep sospeso il clock principale, il rumore digitale da altri moduli è minimizzato e, in questo senso, si ha una elevata correttezza nel risultato.

In particolare, se si utilizza la Vdd come tensione di riferimento, è necessario disporre di una alimentazione accuratamente priva di ripple e disturbi, pena la perdita di una certa parte dei bit

meno significativi, che, in caso di presentazione su display, si traduce in uno sgradevole sfarfallio delle cifre meno significative.

Tensione di riferimento.

Un convertitore AD di qualunque tipologia richiede una tensione di riferimento bassa (Vref-) e alta (Vref+): il livello di tensione di ingresso pari a o inferiore a Vref- produrrà un risultato della conversione uguale a 0, mentre un valore uguale o superiore a Vref+ produrrà il risultato digitale massimo. In questo caso si parla di ingresso raziometrico.

Nei **Midrange** che consideriamo si utilizza in generale la massa (Vss) come Vref- e la tensione di alimentazione Vdd o una Vref esterna come Vref+.
Alcuni chip hanno la possibilità di programmare un pin come ingresso per una Vref- al posto della Vss. In tal caso il range di conversione andrà tra Vref- e Vref+.

La *Vref-* non indica una tensione negativa rispetto alla Vss, ma un valore superiore a questa e inferiore a *Vref+*.
Ad esempio, avendo ***Vref-=1V*** e ***Vref+=4V*** la conversione renderà 1024 step tra 1 e 4V.
La presenza di una Vref- esterna è utile nei casi in cui occorra eliminare un offset.

Ugualmente la ***Vref+*** non può essere superiore alla Vdd.
Tipicamente la ***Vref*** deve essere compresa tra 2.2V e 5.5V, mentre la tensione applicata all'ingresso dovrebbe essere compresa tra ***Vss*** e ***Vref***.

Il consumo di corrente è minimo, meno di 150uA sono assorbiti dalla sorgente.
Gli elementi impiegabili per questo scopo sono molti e sono da preferire integrati del genere voltage reference piuttosto che semplici zener. Ad esempio gli integrati della serie MCP1525 e MCP1541 di Microchip, LM4040, ecc.

Un ingresso ***k*** sarà convertito in un codice digitale:

$$(2n - 1) * (k - Vref-) * (Vref+ - Vref-)$$

dato che ***Vref-=Vss=0***

$$k * (2^n-1) / Vref+$$

dove ***n*** è il numero di bit della risoluzione.

Il risultato della conversione sarà dato da:

$$Vk = Vref- + ((Vref+ - Vref-)* k) / (2^n-1)$$

quindi:

$$Vk = (Vref+ * k) / (2^n-1)$$

La preferenza tra 1023 (***2^n-1***) e 1024 (***2^n***) va al secondo valore:

$$Vstep = Vref / 1024 = Vfullscale / 1023$$

e non ***Vref / 1023*** , dato che:

$$Vfullscale = Vref * 1023 / 1024$$

e non ***Vref***. Quindi:

$$Vstep = Vref * 1023 / 1024 / 1023 = Vref / 1024$$

Comunque, dato che la conversione del valore analogico è costituita da step discreti, ovvero una rampa lineare è sostituita da una scalinata, esiste sempre una incertezza sui valori della rampa intermedi tra due gradini e questo fa parte dell'approssimazione della conversione. Se è richiesta una definizione maggiore, l'unica soluzione è quella di utilizzare una conversione che renda un maggior numero di bit.

Riepilogo dei registri usati per il modulo ADC.

Questi sono i registri interessati alle operazioni col modulo ADC (dal foglio dati di 16F690):

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
05h/105h	PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--xx xxxx	--uu uuuu
06h/106h	PORTB	RB7	RB6	RB5	RB4	—	—	—	—	xxxx ----	uuuu ----
07h/107h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu
0Bh/8Bh/ 10Bh/18Bh	INTCON	GIE	PEIE	T0IE	INTE	RABIE	T0IF	INTF	RABIF	0000 000x	0000 000x
0Ch	PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
11Eh	ANSEL	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0	1111 1111	1111 1111
11Fh	ANSELH	—	—	—	—	ANS11	ANS10	ANS9	ANS8	---- 1111	---- 1111
1Eh	ADRESH	A/D Result Register High Byte								xxxx xxxx	uuuu uuuu
1Fh	ADCON0	ADFM	VCFG	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON	0000 0000	0000 0000
85h/185h	TRISA	—	—	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	--11 1111	--11 1111
86h/186h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	—	—	—	—	1111 ----	1111 ----
87h/187h	TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	1111 1111	1111 1111
8Ch	PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
9Eh	ADRESL	A/D Result Register Low Byte								xxxx xxxx	uuuu uuuu
9Fh	ADCON1	—	ADCS2	ADCS1	ADCS0	—	—	—	—	-000 ----	-000 ----

La tabella indica anche la situazione dei bit nei registri a seguito di reset per **POR/BOR** (caduta della tensione sotto i minimi ammissibili) o per altri reset che non comportano direttamente la mancanza di tensione (**WDT, MCLR**).

I registri di gestione del modulo sono suddivisi su più banchi, sia che il chip abbia 2 o 4 banchi:

ADRESH	1Eh		ADRESL	9Eh		ANSEL	11Eh
ADCON0	1Fh		ADCON1	9Fh		ANSELH	11Fh

Attenzione perchè i registri **PIE1/PIR1/TRISC** sono distribuiti sui vari banchi.

Solo **INTCON** è accessibile da qualsiasi banco.

Risorse addizionali

Microchip offre un certo numero di risorse aggiuntive con informazioni utili sulla conversione AD e argomenti correlati al suo utilizzo.
Le rpincipali informazioni si potranno avere consultando il foglio dati del componente; questo è indispensabile prima di iniziare qualsiasi progetto.

Esistono poi alcune ***Application Notes*** (***AN***):

- ***AN546***, Using the Analog to Digital Converter, una nota generale sull'uso dei moduli ADC
- ***AN682***, Using Single Supply Operational Amplifiers in Embedded Systems
- ***AN722***, Operational Amplifier Topologies and DC Specifications
- *AN723*, Operational Amplifier AC Specifications and Applications
- ***AN679***, Temperature Sensing Technologies,
- ***AN684***, Single Supply Temperature Sensing with Thermocouples
- ***AN685***, Thermistors in Single Supply Temperature Sensing Circuits.

Inoltre:
- **http://www.microcontroller.it/Tutorials/PIC/AD/AD_0.htm**

Per quanto riguarda i riferimenti di tensione:
- **AN82** Understanding and Applying Voltage References di Linear Tech.
- **AN42** Voltage reference circuit collection di Linear Tech.