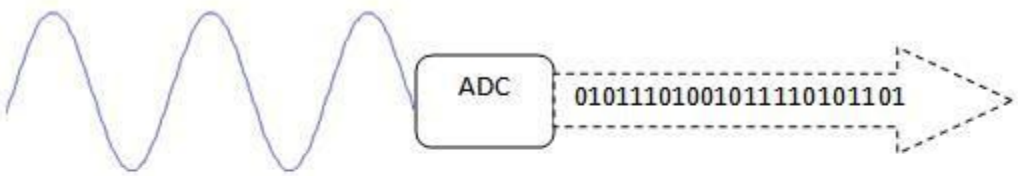
# Embedded System

L8

ADC

# **A**nalog to **D**igital **C**onverter

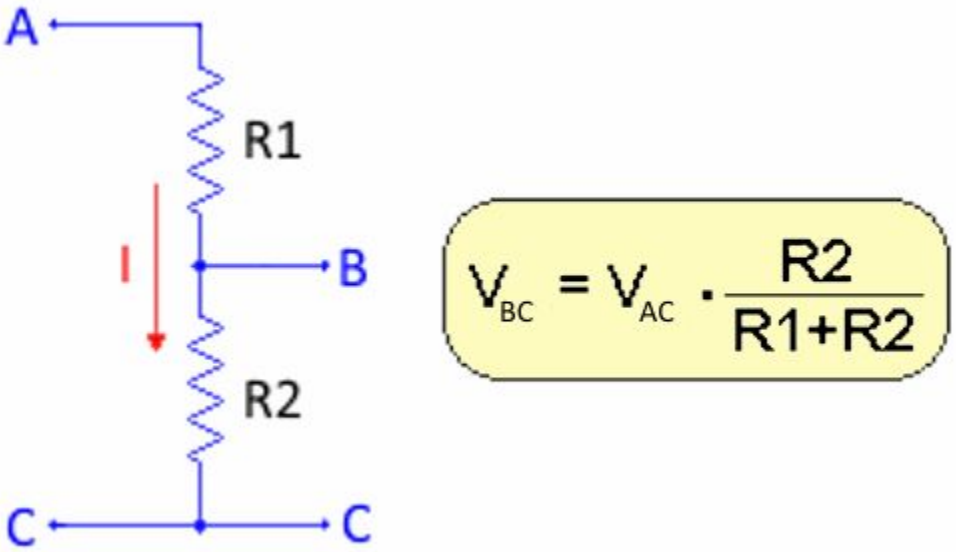
Il modulo ha la funzione di acquisire un segnale analogico, ovvero un segnale di tensione variabile nel tempo, dall'esterno e convertirlo in una parola digitale per poterlo quindi processare, effettuarvi dei calcoli o semplicemente visualizzarlo magari su un display o su una barra a led



L’esempio classico è l’interfacciamento con sonde di temperatura per poter realizzare termometri e termostati.

Esistono una varietà di sonde che forniscono segnali analogici: **sensori di prossimità** ad infrarossi, le **LDR** (Light Dependent Resistors) utili per realizzare controlli della luminosità ambientale e interruttori crepuscolari, **sensori di gas** utilizzati in ambito domestico per realizzare allarmi ma anche etilometri ecc.

Essendo i PIC dispositivi **TTL**, possono ricevere sugli ingressi del convertitore A/D segnali con tensioni variabili da **0** a **5** volt, non possono quindi accettare tensioni negative e, per tensioni superiori a 5 volt è necessario adottare dei **partitori di tensione** (realizzati semplicemente con resistenze) in maniera tale da non ricevere mai in ingresso tensioni superiori a 5 volts.




$$V_{BC} = V_{AC} \cdot \frac{R2}{R1+R2}$$

# Il PIC16F877A ha un convertitore A/D a 10bit

E’ capace di convertire un valore di tensione, variabile da 0 a 5 volt, in una parola formata da 10bit, ovvero in valori digitali che spaziano da zero fino al valore 1023 (1024 valori possibili).

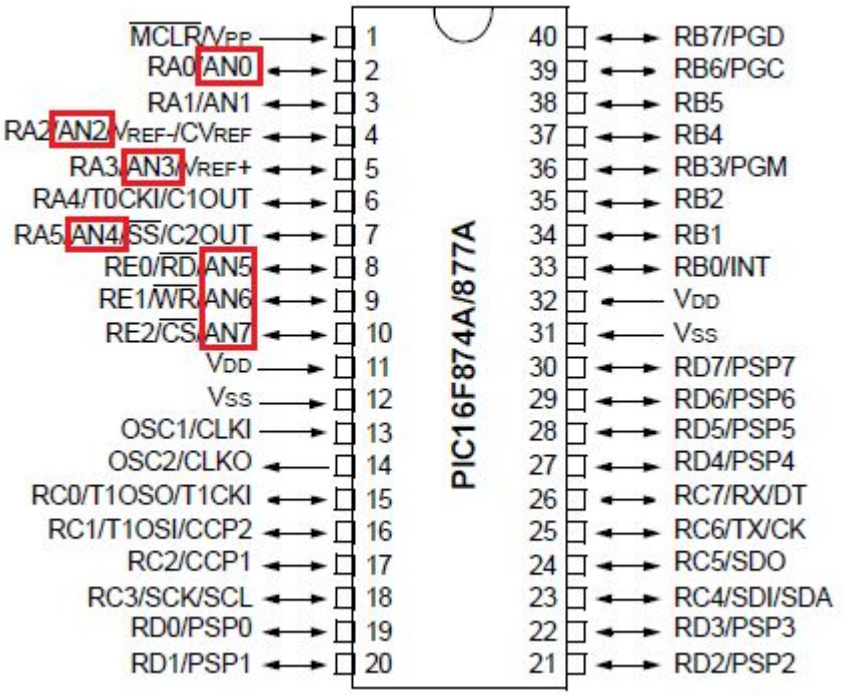
il minimo valore di tensione acquisibile (la sensibilità) vale:

$$\frac{5V}{1024} = 0.00488V = 5mV$$

A volte, quando i segnali in ingresso non forniscono almeno 5mV è necessario ricorrere all’utilizzo di amplificatori.

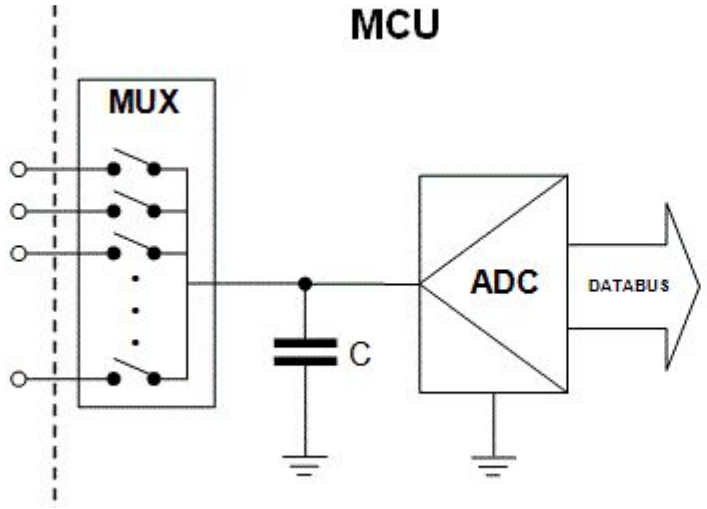
Il convertitore A/D all'interno dei PICMicro è sempre uno solo, ovvero c'è un unico modulo capace di eseguire questa operazione di conversione. Gli ingressi, invece, sono più di uno: i dispositivi a 28pin hanno in genere 5 ingressi analogici, i dispositivi a 40 e 44 pin ne hanno 8.

I pin capaci di acquisire un segnale analogico sono quelli contrassegnati sui datasheet con la sigla ANx dove x sta per il numero di porta analogica, il quale non segue la stessa numerazione della porta digitale a cui fa capo.



# Un ADC, ma molti ingressi

Facile capire che essendoci più di un ingresso analogico ma un solo convertitore, si dovrà fare in modo, ogni volta, di “avvisare” il modulo A/D da quale ingresso analogico prelevare il segnale. Essendo inoltre gli ingressi analogici distribuiti su alcune porte, ci saranno quindi anche dei registri appositi che ci permetteranno di dire se una determinata porta, contrassegnata anche come analogica, dovrà comportarsi come un I/O digitale oppure come un ingresso analogico.



# I Registri dell'ADC

I registri che si occupano della conversione analogico/digitale sono soltanto 4, di cui 2 destinati al setup della periferica e 2 che contengono il risultato della conversione.

- **`ADCON0`**
- **`ADCON1`**
- **`ADRESH`**
- **`ADRESL`**

# ADCON0

E’ il registro che si occupa di controllare le operazioni del modulo convertitore, ovvero: l’accensione/spegnimento, la selezione dell’ingresso dal quale prelevare il segnale, l’avvio delle operazioni di conversione ecc.

E’ un registro a 8 bit

**ADCON0 REGISTER (ADDRESS 1Fh)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| ADCS1 | ADCS0 | CHS2 | CHS1 | CHS0 | GO/$\overline{\text{DONE}}$ | — | ADON |
| bit 7 | | | | | | | bit 0 |

# ADCON0

**bit 7,6** (**ADCS1:ADCS0**) : Servono a selezionare (insieme ad un terzo bit, ADCS2, presente sull'altro registro, ADCON1) la frequenza di clock con la quale far funzionare il convertitore. Vedremo in seguito l'importanza di questa scelta e in base a cosa va fatta. I valori possibili sono i seguenti:

| ADCON1 <ADCS2> | ADCON0 <ADCS1:ADCS0> | Clock Conversion |
|---|---|---|
| 0 | 00 | Fosc/2 |
| 0 | 01 | Fosc/8 |
| 0 | 10 | Fosc/32 |
| 0 | 11 | FRC (clock derived from the internal A/D RC oscillator) |
| 1 | 00 | Fosc/4 |
| 1 | 01 | Fosc/16 |
| 1 | 10 | Fosc/64 |
| 1 | 11 | FRC (clock derived from the internal A/D RC oscillator) |

# ADCON0

**bit 5,4,3** (**CHS2:CHS0**) : Eseguono la selezione dell'ingresso analogico dal quale prelevare il segnale da convertire secondo la tabella:

**CHS2:CHS0:** Analog Channel Select bits
000 = Channel 0 (AN0)
001 = Channel 1 (AN1)
010 = Channel 2 (AN2)
011 = Channel 3 (AN3)
100 = Channel 4 (AN4)
101 = Channel 5 (AN5)
110 = Channel 6 (AN6)
111 = Channel 7 (AN7)

Sto parlando del pin dal quale prelevare il segnale, non del pin da impostare come ingresso analogico, operazione, quest'ultima, che vedremo tra poco.

# ADCON0

**bit 2** (**ADGO**, indicato come **GO/DONE**) : Portando a 1 tale bit, si avvia il processo di conversione. Tale bit viene riportato a zero dall'hardware nel momento in cui la conversione è terminata.

**bit 1** : inutilizzato

**bit 0** (**ADON**) : Accende (1) o Spegne (0) il convertitore A/D. Lo si spegne generalmente per consumare meno corrente e/o e quando il modulo non è utilizzato.

# ADCON1

E’ il registro che si occupa del settaggio delle porte analogiche. Tramite questo registro, difatti, possiamo dire al pic quali pin devono comportarsi come ingressi analogici.

**ADCON1 REGISTER (ADDRESS 9Fh)**

| R/W-0 | R/W-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| ADFM | ADCS2 | — | — | PCFG3 | PCFG2 | PCFG1 | PCFG0 |
| bit 7 | | | | | | | bit 0 |

Già visto nella configurazione di ADCON0

# ADCON1

**bit 7** (**ADFM**) : Serve a selezionare il formato nel quale il valore digitale, risultato della conversione, deve essere memorizzato. Abbiamo infatti detto che il risultato della conversione è un valore a 10bit. Essendo il picmicro in questione un sistema ad 8 bit, il valore digitale proveniente dal modulo A/D sarà per forza di cose memorizzato in due registri ad 8 bit, lasciando quindi inutilizzati 6 bit. Tale bit di configurazione serve in pratica a giustificare verso destra o verso sinistra il risultato della conversione, vedremo tra poco come.

# ADCON1

**bit 6** (**ADCS2**) : insieme ai bit 7 e 6 del registro ADCON0, serve a selezionare la frequenza di clock del convertitore A/D come abbiamo già visto prima.

**bit 5,4** : inutilizzati

# ADCON1

**bit 3,0** (**PCFG3:PCFG0**) : hanno la funzione di impostare quali pin devono comportarsi come ingressi analogici secondo la seguente tabella:

**PCFG3:PCFG0:** A/D Port Configuration Control bits

| PCFG <3:0> | AN7 | AN6 | AN5 | AN4 | AN3 | AN2 | AN1 | AN0 | VREF+ | VREF- | C/R |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000 | A | A | A | A | A | A | A | A | VDD | VSS | 8/0 |
| 0001 | A | A | A | A | VREF+ | A | A | A | AN3 | VSS | 7/1 |
| 0010 | D | D | D | A | A | A | A | A | VDD | VSS | 5/0 |
| 0011 | D | D | D | A | VREF+ | A | A | A | AN3 | VSS | 4/1 |
| 0100 | D | D | D | D | A | D | A | A | VDD | VSS | 3/0 |
| 0101 | D | D | D | D | VREF+ | D | A | A | AN3 | VSS | 2/1 |
| 011x | D | D | D | D | D | D | D | D | — | — | 0/0 |
| 1000 | A | A | A | A | VREF+ | VREF- | A | A | AN3 | AN2 | 6/2 |
| 1001 | D | D | A | A | A | A | A | A | VDD | VSS | 6/0 |
| 1010 | D | D | A | A | VREF+ | A | A | A | AN3 | VSS | 5/1 |
| 1011 | D | D | A | A | VREF+ | VREF- | A | A | AN3 | AN2 | 4/2 |
| 1100 | D | D | D | A | VREF+ | VREF- | A | A | AN3 | AN2 | 3/2 |
| 1101 | D | D | D | D | VREF+ | VREF- | A | A | AN3 | AN2 | 2/2 |
| 1110 | D | D | D | D | D | D | D | A | VDD | VSS | 1/0 |
| 1111 | D | D | D | D | VREF+ | VREF- | D | A | AN3 | AN2 | 1/2 |

A = Analog input  D = Digital I/O
C/R = # of analog input channels/# of A/D voltage references

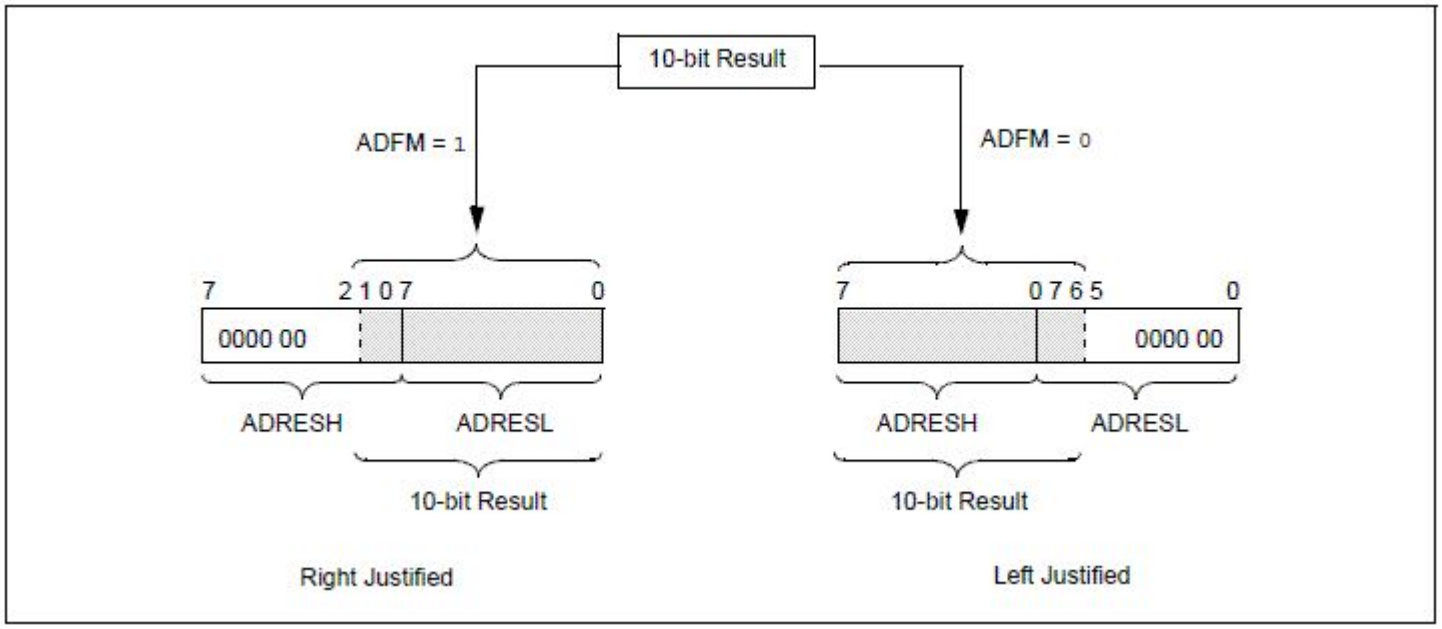
# ADCON1

La scelta degli ingressi analogici va ovviamente fatta, oltre che con questi 3 bit, anche con i registri TRIS interessati, in pratica bisognerà settare come ingressi le porte che vogliamo come analogiche.

Come vedete dalla tabella, non tutte le combinazioni di pin sono possibili e seguendo la colonna di destra possiamo scegliere la configurazione più adatta alle nostre esigenze in base al numero di canali analogici che necessitiamo per la nostra applicazione

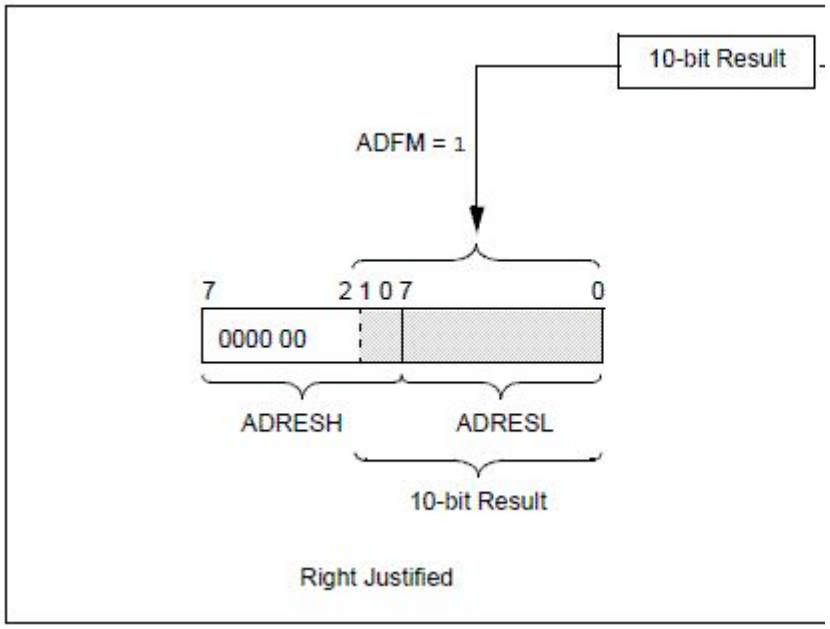
# ADRESH e ADRESL

Sono i due registri che contengono il risultato della conversione (nei pic con convertitore ad 8 bit c'è invece un unico registro chiamato soltanto ADRES). A seconda di come abbiamo impostato il bit ADFM, il risultato sarà memorizzato in maniera diversa:



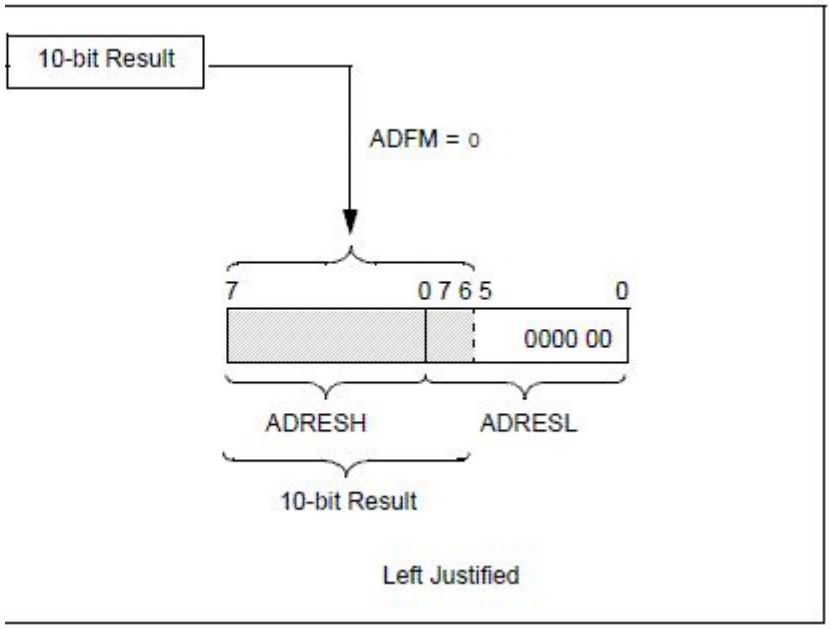
# ADRESH e ADRESL

Se abbiamo messo ADFM ad 1, il risultato verrà giustificato verso destra, ovvero gli 8 bit meno significativi del risultato saranno memorizzati in ADRESL e i due bit più significativi (il bit 9 e il 10 del risultato) saranno memorizzati in ADRESH come bit 0 e 1, i rimanenti 6 bit di ADRESH saranno posti a zero.



# ADRESH e ADRESL

Se abbiamo messo ADFM ad 0, il risultato verrà giustificato verso sinistra, ovvero gli 8 bit più significativi del risultato saranno memorizzati in ADRESH e i due bit meno significativi (il bit 0 e il 1 del risultato) saranno memorizzati in ADRESL come bit 6 e 7, i rimanenti 6 bit di ADRESL saranno posti a zero.



# ADRESH e ADRESL

Se utilizziamo quindi l'impostazione ADFM=1, per ottenere il valore a 16 bit sarà necessario sommare il valore del registro ADRESL a quello del registro ADRESH shiftato di 8 posizioni a sinistra, in quanto i bit più significativi del risultato si trovano in posizione 0 e 1 e devono invece trovarsi in posizione 8 e 9:

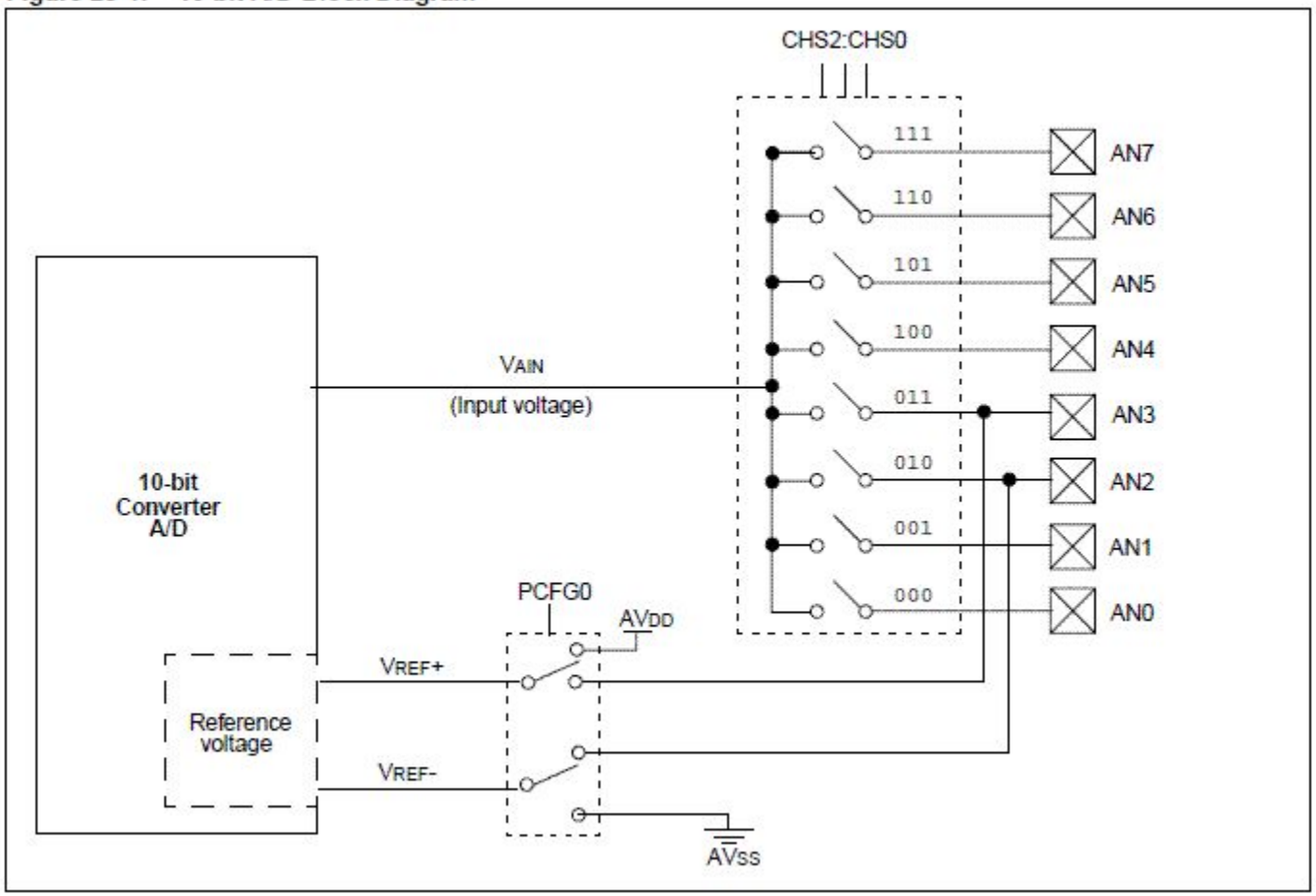
```
int valore;

valore = ADRESL + (ADRESH << 8);
```

# Schema a Blocchi del convertitore

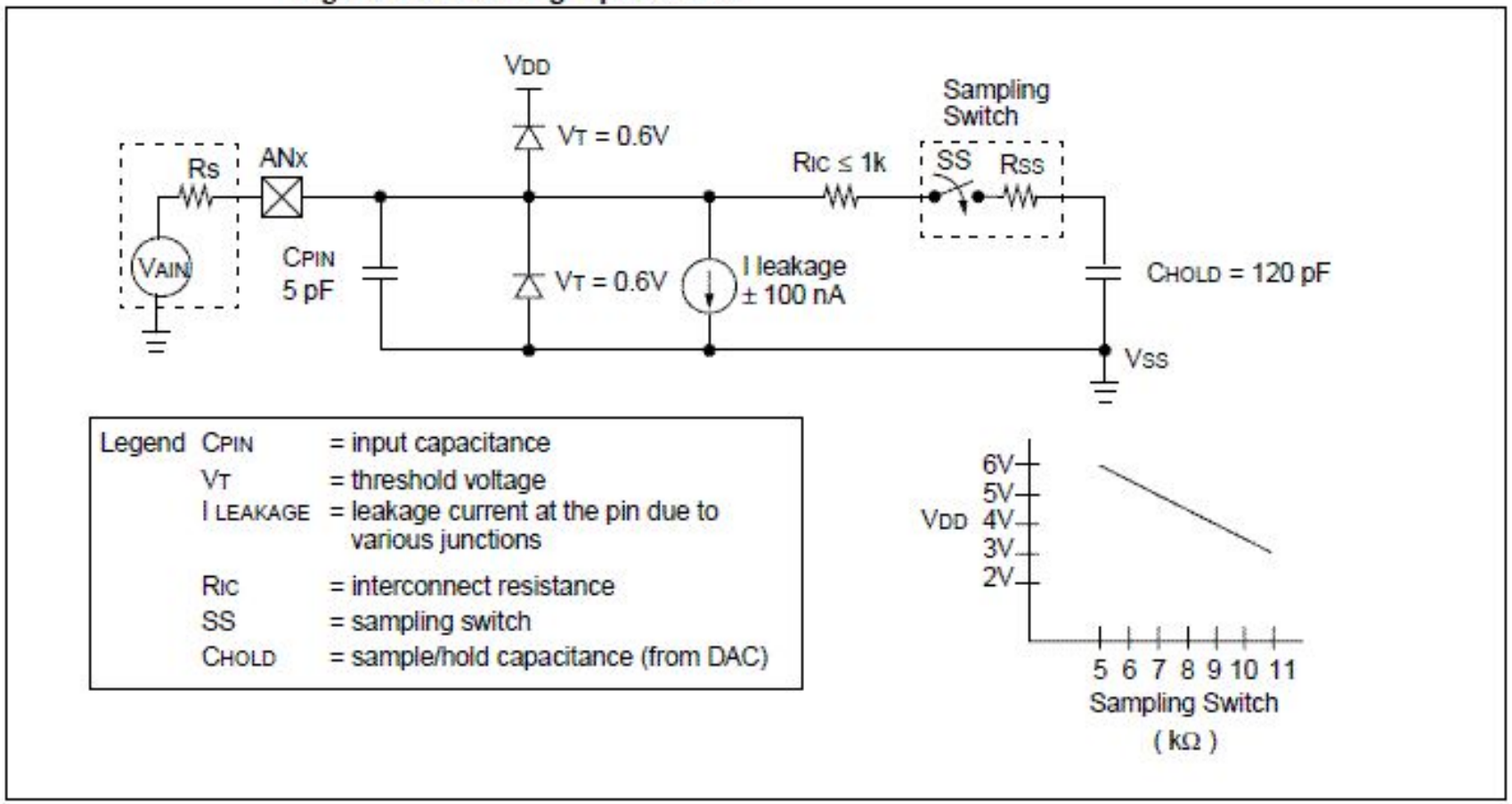
Possiamo quindi schematizzare con un diagramma come è costituito il modulo A/D:

**Figure 23-1: 10-bit A/D Block Diagram**

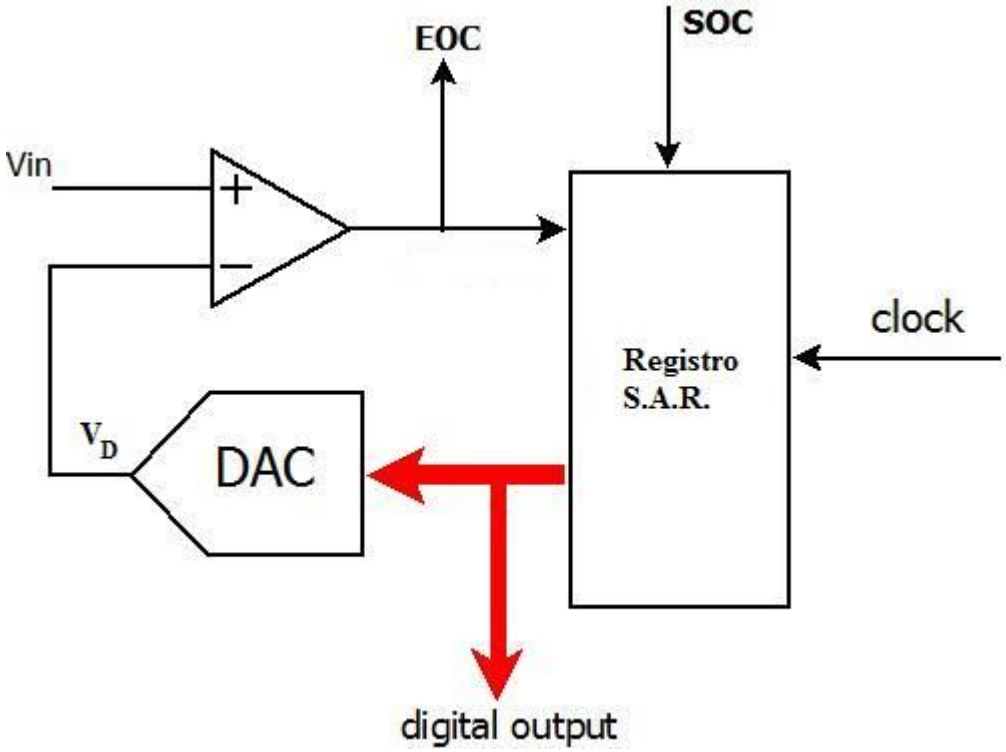


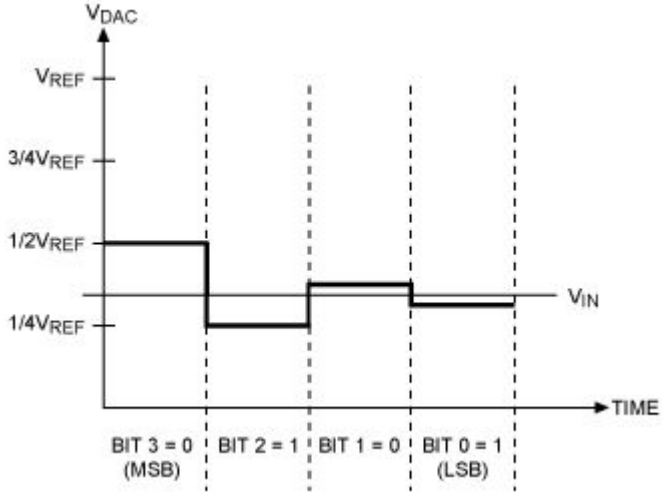
# Principio di funzionamento

**Figure 23-3: Analog Input Model**



# Convertitore ad approssimazioni successive



Al termine della conversione il valore digitale è presente nei registri ADRESH e ADRESL,

viene settato il flag di interrupt di fine conversione analogico/digitale (ADIF),

ADGO viene rimesso a zero

SS viene richiuso per permettere a $C_{HOLD}$ di ricaricarsi e tenersi quindi pronto per un'altra eventuale conversione.

Per effettuare quindi una corretta acquisizione e conversione del segnale bisogna rispettare alcune tempistiche ben precise, abbiamo:

un **tempo di acquisizione**, che è necessario per poter caricare a piena capacità il condensatore di campionamento,

un **tempo di conversione**, che inizia quando settiamo ADGO e termina quando viene azzerato, e rappresenta il tempo necessario al modulo A/D per poter effettuare l'operazione di conversione.

La somma dei due tempi prende il nome di **tempo di campionamento**.

$$Tcamp = Tacq + Tconv$$

# Tutto ritorna… anche **T**

Le impedenze Rs e Rss influiscono sul tempo di acquisizione in maniera non trascurabile.

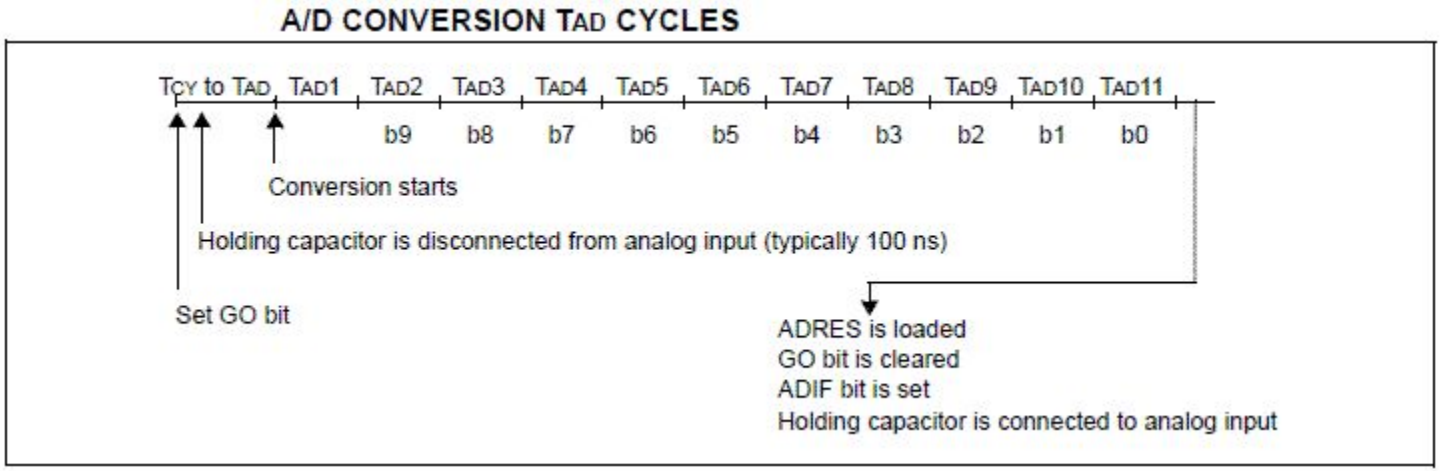
Chi ha studiato elettronica sa, difatti, che il tempo di carica di un condensatore è uguale a 5*$\mathbf{T}$ = 5*R*C

Quanto minore è l'impedenza di ingresso del segnale, meno tempo ci vorrà per caricare il condensatore e quindi tanto inferiore sarà il tempo di acquisizione. Con un'impedenza di ingresso di 50Ω si ha un tempo di acquisizione (ripeto: un tempo di carica di CHold) tipico di circa 10.6μS, con un'impedenza di 10KΩ il tempo tipico sale a 19.7μS.

Tale tempo, purtroppo, **non viene gestito in maniera automatica dal pic** e si capisce, quindi, che, appena finita una conversione non possiamo avviarne immediatamente un'altra perchè non diamo al condensatore di campionamento il tempo necessario per ricaricarsi.

Tra una conversione e l'altra, pertanto, dovremo rispettare un'attesa che è ragionevole scegliere tra 10 e 20μS.

Il tempo di conversione, invece, è funzione di un parametro chiamato TAD, che rappresenta il tempo necessario per convertire un unico bit o, in altre parole, il tempo di conversione per bit. La conversione AD richiede un tempo di 12TAD per una conversione completa a 10bit:



TAD rappresenta in pratica la frequenza di clock scelta per il convertitore A/D (impostata tramite i bit ADCS2:ADCS0 nei registri ADCON1 e ADCON0). La scelta del TAD va fatta in maniera accurata per alcuni semplici motivi:

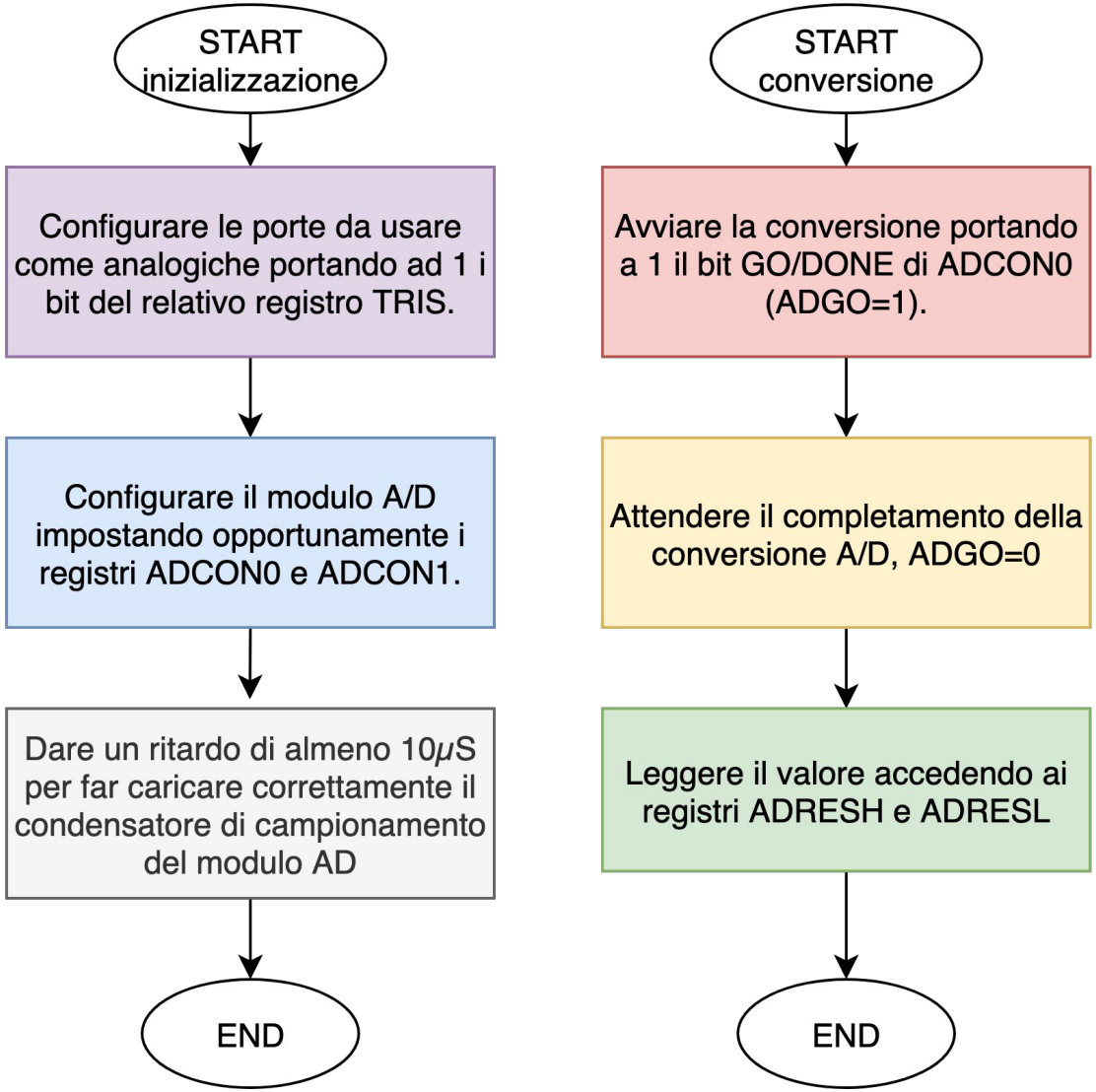
- Bisogna che sia rispettato, per TAD, un valore minimo di 1.6μS. Quindi un clock troppo rapido non potrebbe rispettare tale requisito.
- TAD non deve nemmeno essere troppo elevato: insieme al segnale da campionare viene infatti inevitabilmente acquisito anche del rumore e con tempi maggiori tale fattore aumenta sempre di più in maniera non trascurabile; inoltre potrebbe accadere che il condensatore di campionamento si scarichi prima che la conversione sia terminata. In entrambi i casi il valore restituito dal convertitore è sicuramente corrotto.

Nella seguente tabella possiamo vedere alcuni valori di TAD in funzione del quarzo utilizzato per il clock del pic e del settaggio di ADCS1:ADCS0 (Nota: tale tabella non riporta tutti i valori di clock per il convertitore AD ed assume che ADCS2=0)

| AD Clock Source (TAD) | | Device Frequency | | | |
|---|---|---|---|---|---|
| Operation | ADCS1:ADCS0 | 20 MHz | 5 MHz | 1.25 MHz | 333.33 kHz |
| 2Tosc | 00 | 100 ns(2) | 400 ns(2) | 1.6 µs | 6 µs |
| 8Tosc | 01 | 400 ns(2) | 1.6 µs | 6.4 µs | 24 µs(3) |
| 32Tosc | 10 | 1.6 µs | 6.4 µs | 25.6 µs(3) | 96 µs(3) |
| RC | 11 | 2 - 6 µs(1,4) | 2 - 6 µs(1,4) | 2 - 6 µs(1,4) | 2 - 6 µs(1) |

Notiamo alcuni valori ombreggiati: tali impostazioni non possono essere utilizzate perché TAD risulta troppo breve o troppo lungo e causa quindi problemi per quanto detto prima.

# Settaggio e avvio del modulo A/D



# Esercizio

Impostare tutte le funzioni necessarie all'utilizzo del convertitore ADC:

- void init_ADC()
- int read_ADC(char channel)

Predisporre un codice che possa acquisire la posizione del cursore P1 (collegato a AN0) e che ne visualizzi il valore in codifica binaria sui led collegati a PORTB e PORTD, senza trascurare nessuno dei 10 bit del dato.

# Esercizio

Scrivere un codice che:

- Acquisisca il dato a 10bit dall'ADC relativo allo slider P1
- Converta il numero (0-1023) in un numero con range 0-100
- Visualizzi tale numero sul display