

***Istituto Professionale di Stato per l'Industria e l'Artigianato
MORETTO
Via Luigi Apollonio, 21 BRESCIA***

ACQUISIZIONE DATI INDUSTRIALE

*Sistema di acquisizione a microprocessore
a cura di Franzoni Roberto
Visualizzazione dati su Personal Computer
a cura di Cherubini Luca
Interfacciamento e conversione A/D e D/A
a cura di Giro Maurizio*

Classe 5EI TIEE

Brescia giugno 1993

INTRODUZIONE

L'acquisizione dati è un settore sempre più in evoluzione. Acquisire dati significa leggere informazioni in tempo reale e inviarle a un sistema informativo che li visualizzi e li elabori ottimizzando la produzione e riducendo i costi.

Il sistema realizzato rappresenta un completo acquirente dati che, in simulazione, opera in un generico reparto produttivo monitorizzando la temperatura di due vasche di lavorazione e di alcuni attuatori (motori, pompe, nastri trasportatori).

I dati acquisiti sono inviati a un personal computer che li visualizza su una pagina di lavoro.

La comunicazione fra sistema e personal computer avviene via RS232 con un protocollo basato su pacchetti di dati riscontrati da caratteri ACK NAK.

Il sistema di elaborazione si basa sul microcontroller industriale a 8 bit programmato in linguaggio ASSEMBLY. Il personal computer, un IBM compatibile, è invece programmato in linguaggio PASCAL.

IL PERSONAL COMPUTER

Un personal computer è realizzato in modo modulare; i suoi componenti fondamentali sono: l'unità centrale, la tastiera, il video, l'unità dischi. L'unità centrale è la parte di computer che esegue materialmente i programmi, la tastiera rappresenta l'unità da dove l'utente dialoga con la macchina, il video consente di poter visualizzare le informazioni che la macchina e chi la usa si scambiano, l'unità dischi permette di registrare in modo permanente i dati. Generalmente le unità tastiera e video sono separati dalla macchina, mentre l'unità dischi è posta nella macchina (hard disk), anche se esistono delle memorie di massa (dischetti) che possono essere trasportati da una macchina all'altra. Analizziamo ora i vari elementi costitutivi di un computer: unità centrale.

I dispositivi fondamentali dell'unità centrale sono: il bus di sistema, il sistema di elaborazione, la memoria RAM, i sistemi di controllo, le memorie ROM.

Il bus di sistema collega i dispositivi fra loro, è costituito da linee di controllo, da linee per dati, da linee di alimentazione, da linee di indirizzi per la memoria e i dispositivi di I/O. Il sistema di elaborazione è costituito dalla CPU. La CPU è il microprocessore che svolge le funzioni di elaborazione e delega alcune operazioni speciali al coprocessore. La memoria principale è costituita da schede contenenti integrati di memoria RAM. Le ROM-BIOS consentono l'utilizzo e la gestione delle varie parti del personal computer mediante routine di servizio. Un'unità disco viene utilizzata per poter archiviare su disco i dati e i programmi sotto forma di file. Per poter archiviare un file su disco occorre che il disco sia formattato; questa formattazione dipende dal sistema operativo e dalla macchina. I floppy disk, unità di memoria di massa trasportabile, si distinguono per registrazione su singola o doppia faccia e per densità di registrazione che può essere doppia o quadrupla. Il video è un'unità periferica composta da un adattatore e da un monitor per la visualizzazione dei dati sullo schermo. Vi sono diversi modi per visualizzare i dati, ad esempio: si può far variare il numero di colonne di caratteri sullo schermo, un carattere di testo può essere riprodotto con formati diversi, ecc. Queste sono le modalità video. Attualmente esistono un buon numero di adattatori video e video di alta qualità; i più diffusi sono: adattatore EGA (Enhanced Graphics Adapter), VGA (Video Graphics Adapter) e PGA (Professional Graphics Adapter). La tastiera è l'unità periferica che consente all'utente di comunicare con la macchina. Per la gestione delle periferiche e per consentire il caricamento di programmi si fa uso di un sistema detto MS-DOS. Oltre a consentire le operazioni appena dette il sistema operativo consente di gestire a livello software il personal computer, di modificare a proprio piacimento alcuni caratteri di uso comune della macchina, nonché poter adeguare le caratteristiche del computer alle esigenze dell'utente.

La programmazione dei computer può essere fatta facendo uso del linguaggio macchina oppure di linguaggi ad alto livello. Il **linguaggio macchina** è un linguaggio specifico di una determinata CPU

e quindi presuppone una buona conoscenza del set di istruzioni dei modi di indirizzamento, delle modalità di esecuzione delle istruzioni da parte della CPU e dell'architettura del sistema utilizzato. I **linguaggi ad alto livello** sono completamente indipendenti dall'hardware, sono molto più vicini al modo di esprimersi che è proprio dell'uomo e non richiedono la conoscenza specifica dell'architettura del sistema su cui viene redatto il programma. Tra i linguaggi ad alto livello ricordiamo:

- FORTRAN** adatto alla risoluzione di programmi di tipo scientifico.
- COBOL** adatto alla risoluzione di problematiche di tipo commerciale.
- PL/1** primo linguaggio strutturato è adatto per problematiche scientifiche.
- BASIC** linguaggio ad alto livello più vicino al modo di esprimersi di un uomo, risolve sia problemi di tipo commerciale che scientifico.
- PASCAL** linguaggio strutturato per antonomasia, risolve sia problemi commerciale che di altro genere.
- C** linguaggio derivato dal PASCAL ed utilizzato per la scrittura di sistemi operativi o altri strumenti software.

Il software scritto dal programmatore ed introdotto nella memorie dell'elaboratore viene detto programma **SORGENTE**; quello eseguito materialmente dalla CPU è codificato in linguaggio macchina viene denominato programma **OGGETTO**. Sarà perciò necessario disporre di un traduttore che partendo da istruzioni scritte in linguaggio ad alto livello le trasformi in equivalenti istruzioni stese in linguaggio macchina. Ogni linguaggio dispone quindi di un traduttore in linguaggio macchina (compilatore). Nel modo compilatore il programma sorgente si trova di solito su disco in un file: la preparazione del programma sorgente su disco avviene attraverso l'uso di un apposito "strumento software" detto **EDITORE**. Il compilatore fa un'analisi sintattica di tutto il programma sorgente fornendo eventuali segnalazioni di errore che servono al programmatore per correggere il programma. Nel caso che il controllo sintattico abbia dato esito positivo, viene effettuata la traduzione in linguaggio macchina del programma sorgente; traduzione che viene registrata sottoforma di file. Il programma oggetto così prodotto dal compilatore in genere non è ancora eseguibile in quanto privo delle necessarie informazioni di base per eseguire operazioni matematiche più o meno complesse o operazioni di Input-Output. Si rende necessaria un'altra operazione detta operazione di **LINK** delle librerie che viene attuata per mezzo di un'altro pacchetto software chiamato **LINKING-LOADER**. Il programma oggetto prodotto dal compilatore "linkato" con la routine di libreria è quindi in grado di funzionare automaticamente; anch'esso viene salvato su disco e costituisce il prodotto finale e conclusivo della compilazione. L'impiego di un linguaggio interprete presenta questi pregi:

-Il programma viene modificato in modo molto semplice e in modo altrettanto semplice è possibile verificare la validità delle correzioni.

-Il programma sorgente è sempre presente in RAM quindi è di solito "listabile" su video o stampante.

-La velocità di esecuzione del programma è più elevata rispetto a quella del programma interpretato.

-L'area di memoria a disposizione è la massima possibile in quanto per il suo funzionamento il programma oggetto non ha bisogno ne di compilatore ne tantomeno del programma sorgente che in casi di programmi di una certa lunghezza addirittura non potrebbe nemmeno stare nell'intera RAM del calcolatore.

Si conclude perciò che, ove un determinato linguaggio abbia la possibilità di configurarsi, sarà opportuno operare in ambiente interprete nella fase di stesura, di correzione e di verifica, poi sarà quindi opportuno trasferirsi in ambiente compilatore per procedere alla traduzione in linguaggio macchina del programma predisposto. La videata è gestita da un programma scritto in PASCAL. La versione da noi utilizzata è la 5. 5.

TURBO PASCAL 5. 5

Il turbo PASCAL 5. 5 prevede quattro tipi fondamentali di variabili su cui operare:

Tipo Semplice
Tipo Strutturato
Tipo Puntatore
Tipo Identificatore

per i nostri scopi verranno presi in considerazione solamente i *Tipi Semplici* e i *Tipi Strutturati* e in un caso solo una variabile di *Tipo Puntatore*.

Ai *Tipi Semplici* appartengono i *Tipi Ordinali* e il *Tipo Reale*.

Tutti i possibili valori di un dato *Tipo ordinale* costituiscono un insieme ordinato nel senso che ad ogni elemento è possibile associare un valore intero. Il turbo Pascal prevede 7 tipi ordinali predefiniti (di cui 5 numerici):

ShortInt
Integer
LongInt
Byte
Word

Boolean (0=falsa, 1=vera)
Char

e due classi di tipi ordinali definibili da parte dell'utente:

SHORTINT	-128..+127	8 BIT
BYTE	0...255	8 BIT
INTEGER	-32768...+32767	16 BIT
WORD	0...65535	16 BIT
LONGINT	-2147483648...+2147483647	32 BIT

Una quantità di tipo reale è costituita da un numero che generalmente contiene anche una parte decimale (ma non necessariamente):

REAL	2.9E-39...1.7E38	6 BIT
SINGLE	1.5E-45...3.4E38	4 BIT

DOUBLE	5E-324...1.7E308	8 BIT
EXTENDED	3.4E-4932...1.1E4932	10 BIT

Per visualizzare una quantità reale in formato diverso da quello previsto dalla notazione scientifica bisogna fare ricorso alle istruzioni WRITE e WRITELN, che permettono di scrivere ad esempio un commento a dei dati.

Ai tipi strutturati appartengono gli *ARRAY*, il tipo *STRING*, i *RECORD* e i *FILES*. Un *ARRAY* è un'insieme ordinato di variabili tutte dello stesso tipo; ciascuna di esse viene indicata attraverso il nome dell'array e la posizione che occupa in questo vettore. Una variabile di tipo *STRING* è una sequenza di caratteri variabili fra 1 e 255 caratteri. I *RECORD* sono un tipo di variabile strutturato costituito da un insieme di variabili di tipo diverso fra loro.

Esistono delle strutture dette decisionali, che permettono di eseguire un determinato percorso al verificarsi o meno di determinate condizioni. L'istruzione IF elabora un'espressione di tipo Boolean per determinare quale fra le due possibili alternative deve essere seguita. La clausola ELSE fa parte opzionalmente del costrutto. Il costrutto FOR è adatto per situazioni in cui una istruzione o una serie di istruzioni devono essere ripetute n volte, ove n deve essere noto prima che il ciclo abbia inizio. La struttura REPEAT. .. UNTIL è un costrutto per impieghi generali il cui test di uscita dal ciclo è situato in fondo alla struttura stessa. Si rimane nel ciclo fino a che la condizione dichiarata alla fine è vera. Vediamo ora come funziona il programma scritto in PASCAL.

L'uso della scheda richiede librerie specifiche che consentono di accedere al modo grafico (Graph) e al modo di testo (CRT). Dopo aver dichiarato il modo di utilizzo del programma, bisogna dichiarare le costanti e le variabili che verranno utilizzate. Nel programma non sono utilizzate le costanti, ma solo variabili. Le variabili utilizzate nel programma sono del tipo: Integer, Word, Longint, Real, String. Per eseguire il programma è stato utilizzato anche un Array da 1 a 10 di variabili intere. Questo significa che nel programma abbiamo a disposizione un array composto da dieci variabili di tipo integer. In seguito è stato fatto uso di Function e di Procedure. Possono essere definite come un modulo di codice sorgente. Lo scopo primario è di suddividere programmi vasti e complessi in unità di modeste dimensioni. Possiedono una struttura simile a quella di un programma e in esse possono essere presenti dichiarazioni di costanti o di variabili. Una function si comporta come una procedura ad eccezione del fatto che essa restituisce sempre un valore; tipicamente il valore di ritorno di una function rappresenta il risultato di calcoli basati su valori passati alla function nella fase di esecuzione di un programma. Una procedura viene chiamata esplicitamente attraverso una istruzione costituita solo dal nome della procedura; una funzione viene chiamata implicitamente utilizzando il suo nome in una espressione all'interno di una istruzione.

Nel programma la prima function dichiarata è stata utilizzata per sincronizzare il programma con il " system clock " del computer. Per poter costruire questa function si è fatto uso di un puntatore, questo per avere la certezza che quando viene effettuata la lettura dei valori in ingresso i dati da leggere non stiano modificando il loro valore. Per questo viene preso il dato in ingresso e poi viene confrontato con quello contenuto nel puntatore che aveva eseguito una lettura dello stesso dato subito dopo la function. In seguito viene eseguita una coppia di funzioni che permette di effettuare una temporizzazione espressa in diciottesimi di secondo usando il "system tick" del PC. Dopo aver settato questo "time-out" ne viene controllato lo stato. Se è stato raggiunto la variabile end_time_ver assumerà valore logico 1, altrimenti gli verrà assegnato il valore logico 0. Sfruttando il principio della prima procedura ne viene creata un'altra che immette un ritardo di n diciottesimi di secondo. Finita questa parte relativa ai ritardi e al sincronismo viene poi inizializzata la porta seriale assegnandole l'indirizzo appropriato. La procedura che segue è utilizzata per inviare un carattere sulla seriale per poter ricevere il segnale di ACK da parte del microprocessore (segnale

che informa del corretto arrivo dei dati). Con una funzione , invece, viene controllata la presenza di un carattere sulla porta seriale.

Ora comincia il programma principale. Per prima cosa viene ricercata dal programma la scheda grafica del PC; ciò significa che questo programma si adatta a qualsiasi PC si ha a disposizione. In seguito viene chiamato il protocollo per i colori e per i differenti modi di disegnare richiesti dal programma.

Il comando *setfillstyle* è una modalità definita in ambiente graph che definisce la figura e il colore di tutte le operazioni di riempimento di bar3d. Nel nostro caso è stata utilizzata la sintassi:

```
setfillstyle (interleavefill,lightmagenta)
```

interleavefill = Riempimento con tratto intercalato

lightmagenta = Viola chiaro.

Il comando *bar3d* è una procedura definita in ambiente graph che disegna una barra rettangolare a 3 dimensioni (3D) piena utilizzando la figura ed il colore di riempimento definiti da *setfillstyle*. Per poter scrivere dei commenti in ambiente graph è necessario (se queste scritte sono sparse nello schermo) posizionare il cursore nel posto desiderato: è definita in ambiente graph ed è una procedura che porta in posizione X,Y il puntatore grafico(move to). Outtext è una procedura definita in ambiente graph che invia una stringa di caratteri sullo schermo in corrispondenza delle coordinate assunte dal puntatore grafico corrente. Dopo aver fatto un ritardo di 18 diciottesimi di secondo il programma si "immette" in un ciclo repeat-until che ripeterà fino a quando non riceverà il carattere rappresentante ACK da parte del microprocessore (sullo schermo appare la scritta "RICHIESTA DATI").

A questo punto il programma invia sul canale seriale il pacchetto di dati S, R, E. Questo pacchetto comprende il check-sum diviso in parte alta e parte bassa e i dati appena ricevuti dal microprocessore. L'invio viene ripetuto 6 volte (tanti sono i dati ricevuti) immettendo un dato alla volta nel canale seriale o interrotto nel caso che ci sia qualche problema nella trasmissione dei dati (In questo caso viene visualizzata la scritta "ERRORE IN COMUNICAZIONE"). Dopo aver importato un ritardo di 54 diciottesimi di secondo ("delay (54)") il programma torna a leggere i dati sul canale seriale ed esegue i controlli sul check-sum (sia sulla parte alta che sulla parte bassa), in caso che essi siano errati sullo schermo appare la scritta "ERRORE IN RICEZIONE DATI". Bisogna dire che questi controlli sul check-sum sono dei normali controlli che vengono eseguiti quando sono utilizzati i canali seriali e questi controlli prendono il nome di somma di verifica. Nel caso che il check-sum sia corretto sullo schermo appare la scritta "RICEZIONE DATI CORRETTA" e poco dopo appariranno i dati. Dato che ogni volta che appaiono le scritte vengono a sovrapporsi, bisogna riempire il rettangolo con il comando bar. Questo comando disegna un rettangolo pieno senza contorno esterno) utilizzando la figura e il colore definiti in *setfillstyle*. Ora il programma sistema i dati ricevuti in ingresso alla seriale in modo da avere (per le vasche) una scala di temperature che varia da 0 gradi centigradi fino a 300°C. Se la temperatura supera il valore 250°C si innesca una sirena che smetterà di suonare solo se la temperatura scenderà sotto i 250°C. Per comunicare il PC e il microprocessore viene usato il canale seriale. Analizziamo come funziona questo tipo di comunicazione.

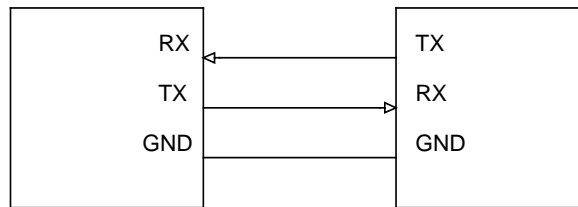
COMUNICAZIONI SERIALI

Il canale seriale RS232 è il mezzo standard attraverso cui un dispositivo a microprocessore (microcontroller, PC, PLC, modem telefonico ecc.), può connettersi con altri dispositivi dotati di questa capacità di comunicazione.

Un collegamento seriale RS-232 richiede tipicamente tre fili:

- 1) La linea di Tx (trasmissione);
- 2) La linea di Rx (ricezione);
- 3) La massa comune (Gnd);

che consentono a due sistemi di scambiare informazioni. Schematicamente:



Come si può vedere, la linea Tx di ogni sistema è connessa alla linea Rx dell'altro mentre le masse sono unite.

Il principio di funzionamento di un collegamento seriale è basato sulla scomposizione dei dati da trasmettere (byte, parole binarie) in bit inviati SERIALMENTE (in sequenza) NEL TEMPO, lungo la linea di comunicazione. Il principale vantaggio della comunicazione seriale è il ridottissimo numero di linee necessarie. Si tratta di un vantaggio molto importante: si pensi al risparmio di cavi (e di denaro) in un collegamento tra sistemi molto distanti. Lo svantaggio è il tempo richiesto: la trasmissione seriale comporta continue temporizzazioni (l'intervallo tra l'invio di un bit ed il successivo).

L'alternativa alla comunicazione seriale è la comunicazione PARALLELA. In questo tipo di collegamento sono utilizzati almeno DIECI fili oltre alla massa comune (in realtà le linee sono ancora di più, perchè sono previsti anche segnali addizionali di controllo, sincronismo e segnalazione errore). L'esempio più comune di comunicazione parallela si trova nel collegamento PC- stampante: in tale tipo di collegamento, otto delle linee richieste trasmettono il byte di informazione (linee "DATA"), una ("STROBE") convalida l'informazione presente sulle linee "DATA" e l'ultima è usata dalla stampante per segnalare al computer la sua condizione di "READY" (condizione che abilita l'invio dei dati da parte del computer: quando la stampante non è "READY" il PC deve attendere). La comunicazione parallela è spesso

UNIDIREZIONALE (PC ==> stampante e non viceversa) e questo rappresenta un altro svantaggio (esistono tuttavia collegamenti paralleli bidirezionali). La comunicazione parallela è naturalmente molto più veloce della comunicazione seriale ma anche più costosa. È del tutto inutilizzabile, inoltre, attraverso linea telefonica (sarebbero necessari almeno dieci telefoni dedicati!).

Si comprende quindi come il collegamento parallelo sia utilizzato per brevi distanze, oppure nei casi in cui l'elemento determinante sia la velocità.

torniamo alla trasmissione seriale precisando però che trattarla in modo approfondito esula dalle finalità di queste pagine, anche perchè l'ampiezza dell'argomento è tale da richiedere, da solo, una monografia molto voluminosa. Ci limiteremo quindi ad esaminare gli aspetti fondamentali di una comunicazione RS-232 che sono:

1) Il LIVELLO FISICO dei segnali, vale a dire la tensione o la corrente che tali segnali caratterizza. Un collegamento RS-232 può avvenire con segnali TTL (ma solo su brevi distanze), con segnali $\pm 12V$ (si parla in questo caso di RS-232C), con segnali in corrente ("current loop" a 10, 20, 30 mA). Nella definizione del livello fisico è inclusa la specifica relativa al LIVELLO LOGICO: tipicamente un segnale ZERO rappresenta un bit ad UNO, un segnale ad UNO un bit a livello ZERO.

2) Il secondo parametro fondamentale è la VELOCITÀ di trasmissione, vale a dire la velocità con cui sono emessi i bit di informazione. La velocità di trasmissione è espressa in BAUD RATE o NUMERO DI BAUD AL SECONDO. Il baud viene definito come variazione del segnale all'interno di un canale di comunicazione (in pratica il passaggio da livello logico ZERO ad UNO oppure da UNO a ZERO). La baud rate esprime quindi il massimo numero di variazioni che un segnale può avere in un secondo relativamente ad un certo canale di comunicazione. Si noti che la baud rate NON esprime un numero di bit al secondo anche se in pratica si tende ad assimilare il baud al bit; affermare che una certa comunicazione avviene a 1200 baud equivale quindi (praticamente) ad affermare che vengono trasferiti 1200 bit al secondo. Il protocollo RS-232 fissa alcune velocità standard per le trasmissioni; le più comuni sono:

300, 600, 1200, 2400, 4800, 9600, 19200 baud.

Non esistono dispositivi standard che dialoghino, per esempio, a 1000 o 5000 baud (anche se sono previste alcune velocità intermedie allo standard quali 1800, 3600, 7200 baud).

3) Il terzo parametro fondamentale dell'RS-232 è il FORMATO DI TRASMISSIONE vale a dire il NUMERO DI BIT della parola da trasmettere (per ogni parola), il numero di BIT DI START, il numero di BIT DI STOP, l'eventuale BIT DI PARITÀ o DI DISPARITÀ. Il bit di START è il segnale che origina SEMPRE la trasmissione di una parola, il bit di STOP è il segnale che la termina SEMPRE. Per definizione il bit di STOP è il contrario del bit di START. Si possono avere 1, 1.5 oppure 2 bit di STOP. Senza bit di START e di STOP sarebbe impossibile per il ricevente capire quale sia l'inizio di una trasmissione, quale sia il bit zero di un byte, quale ne sia il bit sette. Se si stabilisce che il bit di START sia ZERO ed il bit di STOP UNO, l'informazione sarà certamente delimitata da uno ZERO iniziale e da un UNO finale.

Il NUMERO DI BIT DI DATO specifica la dimensione della parola da trasmettere: tipicamente si trasmettono parole di 7 od 8 bit (dimensioni diverse sono poco usate).

Il bit di PARITÀ o DISPARITÀ è opzionale: si tratta di un elemento introdotto per aumentare l'affidabilità della comunicazione. Può accadere, infatti, che la ricezione di un bit avvenga in modo errato cioè un livello logico ZERO diventi UNO o viceversa (a causa, per esempio, di interferenze elettriche nel canale di comunicazione): in un caso del genere il ricevente non ha modo di accorgersi dell'errore, condizione che può invece essere rivelata dal bit di parità. Il bit di parità viene aggiunto al dato per far in modo che IL NUMERO COMPLESSIVO DI BIT A LIVELLO LOGICO UNO nel dato stesso sia PARI. Verificando la condizione di parità, il ricevente può stabilire la correttezza o meno dell'informazione trasmessa: se il numero di bit ad UNO nel dato ricevuto è PARI la trasmissione è avvenuta correttamente, se tale numero è DISPARI allora si è in presenza di errore. Vediamo praticamente come la parola binaria di otto bit:

1010 1011

possa venire integrata con la parità. Il numero di bit a livello logico UNO in essa è CINQUE, cioè un valore DISPARI. Aggiungiamo quindi un bit ad UNO ottenendo:

1010 1011 1

cioè una nuova parola di nove bit in cui il numero complessivo di bit a livello logico UNO è PARI (vi sono infatti ora SEI bit ad UNO). La parola binaria:

1000 1011

presenta un numero PARI (QUATTRO) di bit ad UNO; la parità sarà quindi un bit a valore ZERO:

1000 1011 0

Usando invece il bit di DISPARITÀ si ottiene un dato in cui il numero complessivo di bit ad UNO è DISPARI.

È importante osservare che l'aggiunta della parità (o della disparità) NON garantisce l'assenza di errori: infatti se i bit di dato alterati sono due, la parità non lo rivela (e nemmeno

la disparità). Da notare anche che l'aggiunta del bit di parità ALLUNGA la parola da trasmettere e quindi AUMENTA il tempo richiesto per trasmettere i dati.

Per aumentare ulteriormente l'affidabilità di un collegamento seriale, si possono impiegare (anche contemporaneamente), più tecniche di controllo: parità particolari, check-sum (somma di verifica), integrazioni matematiche all'informazione ecc. Il modo forse più ovvio, rispedire al trasmittente i dati ricevuti affinché ne controlli la correttezza, presenta lo svantaggio di RADDOPPIARE il tempo totale di comunicazione (con conseguenze diverse: impossibilità se i dati riguardano un processo che si sta svolgendo in tempo reale, costi se la comunicazione avviene per via telefonica): è quindi scarsamente utilizzato.

4) Il quarto parametro fondamentale di una comunicazione RS-232 è il PROTOCOLLO DI COMUNICAZIONE. Il protocollo di comunicazione specifica la struttura dei dati che devono essere scambiati e l'organizzazione dell'intera trasmissione. Ad esempio si può stabilire che l'unità MASTER (principale) invii allo SLAVE (unità secondaria), un carattere convenzionale cui lo slave risponda con delle informazioni. Caratteri convenzionali possono essere "STX" ("Start of transmission"), "ETX" ("End of transmission"), "XON" ("Trasmissione abilitata"), "XOFF" ("Trasmissione disabilitata") e parecchi altri. Un protocollo di comunicazione deve anche gestire l'eventualità che lo slave sia impossibilitato a rispondere, o che una richiesta sia stata ricevuta erroneamente: per le diverse eventualità devono essere indicati dei procedimenti (esempio: ripresentazione richiesta "n" volte, attesa risposta per un tempo massimo definito ecc.).

In alcuni casi la comunicazione seriale richiede delle linee fisiche aggiuntive che permettono a due sistemi di meglio sincronizzarsi: tali linee aggiuntive vengono genericamente definite di "HAND-SHAKE"; alcuni esempi sono: il segnale "RTS" ("Request To Send", "richiesta di trasmissione"), il segnale "DTS" ("Data Terminal Ready", "ricezione dati abilitata") ed altri, usati tipicamente in collegamenti modem.

GENERALITÀ SUI SISTEMI ELETTRONICI A MICROPROCESSORE

Il microprocessore è nato per soddisfare le differenti necessità che si presentano nei vari campi del controllo industriale, dell'automazione e dell'acquisizione dati (oltre ad essere impiegato su larghissima scala nei prodotti di consumo). Questo componente ha una larga scala di integrazione e contiene al suo interno elementi in grado di svolgere le più diverse funzioni; è programmabile e, proprio per questo, estremamente flessibile.

La disponibilità del microprocessore riduce notevolmente il numero di integrati dei circuiti applicativi, integrati le cui funzionalità sono sostituite dal software.

Il microprocessore è un dispositivo che non può essere utilizzato da solo, ma fa parte di un "sistema" comprendente memorie elettroniche capaci di immagazzinare istruzioni e dati di programma e circuiti in grado di trattare ("interfacciarsi" con) segnali di vario genere.

L'insieme dei circuiti che costituiscono un "sistema" a microprocessore rappresenta la struttura rigida ed è definito HARDWARE, mentre l'insieme dei dati e delle istruzioni è definito SOFTWARE.

Possiamo definire due diversi metodi di progettazione e costruzione di circuiti elettronici:

1) LOGICA CABLATA:

Il circuito è destinato a eseguire un solo compito senza possibilità di cambiamento

2) LOGICA PROGRAMMATA:

Questa soluzione offre la possibilità al sistema di svolgere, modificando le istruzioni e la circuiteria, altre funzioni.

Naturalmente sia l'una che l'altra soluzione hanno pregi e difetti, la tendenza è tuttavia sempre più orientata verso la logica programmabile. La struttura dei sistemi a logica programmata è certo più complessa e richiede i seguenti elementi:

1) UNITÀ CENTRALE:

Svolge l'elaborazione dei dati eseguendo le istruzioni di programma; questa unità è meglio conosciuta come CPU o come microprocessore.

2) MEMORIE:

Possono essere permanenti o volatili e sono destinate a memorizzare le istruzioni e i dati relativi alla CPU.

3) UNITÀ DI INGRESSO :

Sistemi elettronici che rendono possibile la comunicazione tra il mondo esterno e il sistema.

4) UNITÀ DI USCITA :

Sistemi elettronici che rendono possibile la comunicazione tra il sistema ed il mondo esterno.

5) BUS DATI :

Elemento di collegamento attraverso il quale transitano i dati e le istruzioni in riferimento al quale si definisce la capacità di parola del sistema che in generale è di 8, 16 o 32 BIT.

6) BUS CONTROLLI:

Insieme di conduttori su cui si muovono informazioni binarie che integrano i segnali di bus.

7) BUS INDIRIZZI :

Collegamento che va dal microprocessore alle altre parti del sistema. Rappresenta "l'indirizzo" delle celle di memoria e di I/O del sistema.

MB-6501 MICROBOARD

La scheda utilizzata è la MB-6501; tale scheda possiede grandi capacità di controllo sul mondo esterno essendo fornita di una CPU ad otto bit, 30 linee di I/O programmabili via software, due canali seriali indipendenti, due timer a 16 BIT, due zoccoli configurabili per chip RAM/EPROM/EEPROM da 8/16/32 KBYTE, un real-time-clock con batteria autonoma in back-up e altre funzionalità che rendono il funzionamento più lineare possibile.

La MB-6501 può sfruttare al massimo gli elementi cui viene collegata grazie ai jumper di predisposizione che servono anche per la selezione del tipo di chip di memoria utilizzati.

Le connessioni di I/O sono riportate su una striscia di 30 contatti strip Molex così organizzate:

Pin 1 ==> 8 Linee PDO,PD1.....PD7 del 6501Q

Pin 9 ==> 14 Linee PC5,PC4..... PC0 del 6501Q

Pin 15 ==> 22 Linee PB7,PB6..... PB0 del 6501Q

Pin 23 ==> 30 Linee PA0,PA1..... PA7 del 6501Q

I PORT PA7 e PA6 sono impegnati per la comunicazione seriale.

Si noti che tutte le linee di I/O fanno parte del MICROCONTROLLER R6501 (o microcomputer "single-chip"). Il microprocessore è il nucleo fondamentale di un sistema elettronico in logica programmata; la sua struttura è in grado di gestire dati in forma binaria operando secondo le modalità fornite dal programma.

Un microprocessore presenta le seguenti caratteristiche fondamentali:

1) Lunghezza di parola: Già citata in precedenza, esprime la dimensione in BIT delle parole binarie che la CPU è in grado di trattare si parla di CPU a 8,16,32 BIT.

2) Qualità e quantità delle istruzioni previste: A questo proposito esistono due tendenze: "processori veloci con un numero di istruzioni ridotte (RISC =Reduced Instruction Set Computer)

oppure processori meno veloci con un numero di istruzioni maggiore (CISC = Complex Instruction Set Computer)".

3) Velocità di clock: Il clock è un segnale (generato tipicamente da un oscillatore a quarzo), che sincronizza l'esecuzione delle istruzioni di programma da parte della CPU.

La struttura di un microprocessore, è costituita essenzialmente da REGISTRI. I registri del 6501Q sono:

1) ACCUMULATORE :

È il registro principale della CPU. Esteso su 8 BIT, svolge la maggior parte delle operazioni aritmetico-logiche. Viene indicato con "A".

2) REGISTRO INDICE :

È un registro ad 8 BIT in grado di svolgere alcune operazioni aritmetico-logiche anche se non con la flessibilità dell'accumulatore; è usato per accedere in modo indicizzato alla memoria. Viene indicato con "X".

3) REGISTRO DI STATO :

È un registro ad 8 BIT, alcuni dei quali riflettono l'esito dell'istruzione precedentemente eseguita. Ogni BIT è detto "FLAG", e alcuni di essi regolano le modalità operative della CPU. Viene indicato con "P".

4) STACK POINTER :

È un registro ad 8 BIT che lavora in modalità LIFO (Last Input First Output) su una particolare area di memoria della RAM detta Stack. Stack pointer significa letteralmente "puntatore di catasta" e si indica con "SP".

5) DECODIFICATORE DELLE ISTRUZIONI:

È il registro che interpreta le istruzioni, cioè i comandi impartiti dalla CPU.

6) PROGRAM COUNTER:

È un registro a 16 BIT che contiene l'indirizzo della prossima istruzione da eseguire. Normalmente è diviso in due parti: "contatore di programma parte alta" e "contatore di programma parte bassa", inoltre lavora in connessione con il BUS degli indirizzi. Le due parti del contatore si indicano con "PCH" e "PCL" (High e Low).

7) UNITÀ ARITMETICO LOGICA :

È l'unità deputata a svolgere quasi tutte le operazioni aritmetico-logiche che avvengono internamente o esternamente al microprocessore.

8/9) BUFFER DATI E INDIRIZZI :

È il registro che gestisce l'indirizzamento esterno per l'accesso al programma.

Il registro di stato ha una grande importanza nell'uso della CPU: i suoi bit vengono chiamati FLAG, termine che possiamo tradurre con "INDICATORE" (di uno stato macchina). I flag del registro di stato servono per regolare il funzionamento della CPU, controllano l'esecuzione dei salti condizionati e determinano le modalità operative di alcune istruzioni, qui di seguito sono riportati l'uso e il significato dei flag:

FLAG Z:

È il flag di ZERO, e assume livello logico UNO quando l'esecuzione di un'istruzione provoca, in un registro interno alla CPU, o in una locazione di memoria interna un valore zero (Un byte contiene il valore hex 00). Valori risultanti diversi da zero dispongono il flag a livello logico ZERO.

FLAG C:

È il flag di carry, e assume livello logico UNO quando l'esecuzione di un'operazione di confronto, di somma o sottrazione produce un riporto, oppure quando un'operazione di shift determina la fuoriuscita di un BIT a livello logico uno.

FLAG V:

È il flag di overflow, e assume significato solo nello svolgimento di somme e sottrazioni di valori con segno rappresentati in complemento a due, in BCD non ha significato. Il flag V si attiva solo quando l'esecuzione di un'istruzione di somma o sottrazione binaria supera la capacità di calcolo nell'ambito degli otto BIT, si possono rappresentare valori che vanno da 0 a 127 e valori che vanno da -1 a -128.

FLAG N:

È il flag di negativo e si attiva quando l'ultima istruzione eseguita produce, in un registro della CPU o in locazione di memoria esterna, un byte avente il BIT 7 a livello logico uno, al contrario si dispone a livello logico zero se l'istruzione eseguita produce un byte avente bit 7 a livello logico zero.

FLAG D:

È il flag di decimale controlla lo svolgimento delle operazioni aritmetiche. Se fissato a livello logico uno

dispone la CPU ad eseguire operazioni in BCD, al contrario se fissato a livello logico zero dispone la CPU a lavorare in binario.

FLAG I:

È il flag che controlla la sensibilità della CPU alle interruzioni di programma richieste da dispositivi interni o esterni al chip. Se è disposto a livello logico zero rende la CPU sensibile alle interruzioni, disposto a uno la rende insensibile.

Per l'utilizzo del microprocessore è necessario conoscere anche i vettori di reset e di interrupt, che stabiliscono gli indirizzi di inizio (RESET) ed interruzione (BREAK) programma. Un'interruzione di programma è un segnale hardware (elettrico) che il processore riceve e che lo porta a interrompere il programma in prova per eseguirne un'altro: il nuovo programma è appunto la procedura di interrupt (concettualmente l'interrupt è una subroutine la cui esecuzione dipende NON da un'istruzione di tipo "JSR" MA da un segnale elettrico).

Gli interrupt sono di due tipi: "mascherabili" e "non mascherabili". L'interrupt mascherabile è una richiesta di interruzione che può essere ignorata tramite abilitazione software; l'interrupt non mascherabile comporta invece l'esecuzione obbligatoria della relativa sequenza. Scopo degli interrupt è per esempio gestire sequenze di emergenza, rispondere alla pressione di un pulsante, oppure il "multitasking" ("multi- processo") in cui la CPU viene portata ad eseguire processi "paralleli".

In particolare, il 6501Q consiste in una CPU ad otto bit potenziata, un oscillatore di clock interno, 192 BYTES di RAM (Memoria ad accesso casuale) ed una versatile circuiteria di interfaccia che include due TIMER/COUNTER programmabili a 16 BIT, 30 linee bidirezionali INPUT/OUTPUT, 10 sorgenti di interruzione e un BUS espandibile ed è compatibile con tutti i membri della famiglia R6500.

Un microprocessore ad otto bit viene tipicamente programmato in linguaggio ASSEMBLY.

Il sistema realizzato utilizza un microcontroller 6501Q interfacciato con un convertitore A/D ad otto bit e ad alcune linee di input filtrate e triggerate. Due degli otto canali del convertitore A/D sono utilizzati per leggere lo stato dei potenziometri che rappresentano la temperatura delle due vasche monitorizzate. Le sei linee digitali rappresentano lo stato ON/OFF degli attuatori e sono, nella simulazione, riportate su interruttori.

ASSEMBLY

L'assembler è il linguaggio "naturale" della macchina, è un linguaggio in cui le istruzioni sono rappresentate da un CODICE OPERATIVO (comando vero e proprio) eventualmente seguito da uno o due byte di operando che, quando esistono, specificano DOVE e SU CHE COSA l'istruzione deve agire (destinazione).

In assembler le istruzioni sono composte da uno, due o tre elementi.

L'espressione "INX" che significa "INcrement X register", aumenta di una unità il valore contenuto nel registro X ed è denominato SIMBOLO MNEMONICO del byte che in linguaggio macchina è dato da \$E8.

L'espressione "LDA ALFA", che significa "LoAD Accumulator with", carica l'accumulatore con il contenuto di "ALFA" (nome simbolico per una qualsiasi locazione di memoria o di I/O); in

linguaggio macchina è dato da \$AD \$XX \$YY dove \$AD è il codice operativo e \$YYXX è l'operando (indirizzo esadecimale di "ALFA").

Si noti che la CPU R-6501 richiede che gli indirizzi siano sempre dati nell'ordine parte bassa/parte alta.

Esistono vari modi di indirizzamento:

La CPU R-6501 offre tredici modi di indirizzamento, compreso quello "DI ACCUMULATORE" e prevede istruzioni di caricamento e scaricamento dei registri interni (Funzioni logiche di OR, AND e di EXOR, di set/reset del bit ecc).

INDIRIZZAMENTO IMPLICITO: In questo modo di indirizzamento l'istruzione richiede un byte soltanto poiché è relativa ai soli registri interni della CPU.

DEY ==> Decrementa di uno il contenuto del registro Y
CLC ==> Disattiva il flag "C" (Carry) nel registro di stato

INDIRIZZAMENTO IMMEDIATO: In questo modo di indirizzamento l'istruzione, convertita in linguaggio macchina, è costituita da due byte di cui solo il secondo rappresenta il dato a cui si riferisce il codice operativo.

LDA #\$55 ==> Carica l'accumulatore con il byte \$55
CPX #\$00 ==> Confronta il contenuto del registro X con 0

Il carattere "#" è usato per indicare il modo di indirizzamento immediato in assembler.

INDIRIZZAMENTO ASSOLUTO: Una volta convertita in linguaggio macchina l'istruzione è costituita da tre byte: Un codice operativo e un indirizzo che esteso su due byte rappresenta la locazione di memoria su cui agisce il codice operativo.

LDA \$1234 ==> Carica l'accumulatore con il contenuto della locazione di memoria di indirizzo \$1234

STX ALFA ==> Deposita il contenuto del registro X nella locazione di memoria ALFA

INDIRIZZAMENTO IN PAGINA ZERO: Simile al modo di indirizzamento assoluto, solo che in questo caso, si opera soltanto sulla memoria di PAGINA ZERO, cioè l'area di memoria di indirizzo compreso tra \$0000 e \$00FF (Ricordiamo che una pagina è formata da 256 BYTE), poichè la parte alta dell'indirizzo cui si riferisce l'istruzione è zero, è consentito specificare solo la parte bassa:

LDA *\$50 ==> Carica l'accumulatore con il contenuto della locazione di memoria \$0050

STY *BETA==> Deposita il contenuto del registro Y nella locazione di memoria BETA (necessariamente in pagina zero)

IL carattere "*" indica il modo di indirizzamento in PAGINA ZERO.

INDIRIZZAMENTO ASSOLUTO INDICIZZATO IN X: In questo modo di indirizzamento esteso su tre byte, il dato cui si riferisce l'istruzione è ottenuto sommando all'indirizzo specificato il valore contenuto nel registro X:

LDA \$1234,X ==> Carica l'accumulatore con l'indirizzo contenuto nella locazione di memoria \$1234 più il valore contenuto in X.

STA ALFA,X ==> Deposita il contenuto dell'accumulatore nella locazione di memoria di indirizzo ALFA aumentato di X.

INDIRIZZAMENTO ASSOLUTO INDICIZZATO IN Y: In questo modo di indirizzamento, anch'esso esteso su tre byte, il dato è ottenuto sommando all'indirizzo specificato il valore contenuto nel registro Y. L'unica differenza dal precedente modo è l'utilizzo del registro Y invece dell'indirizzo X:

LDA \$1234,Y ==> Carica l'accumulatore con l'indirizzo contenuto nella locazione di memoria \$1234 più il valore contenuto in Y

STA ALFA,Y ==> Deposita il contenuto dell'accumulatore nella locazione di memoria di indirizzo ALFA aumentato di Y.

INDIRIZZAMENTO DI PAGINA ZERO INDICIZZATO IN X: Questo modo di indirizzamento è simile al modo assoluto indicizzato in X, con la sola differenza che in questo caso, l'istruzione richiede solo due byte, essendo zero la parte alta cui si riferisce; analogamente, abbiamo un modo di indirizzamento in pagina zero indicizzato in Y l'unica differenza è dell'uso di uno dei due registri:

LDA *\$12,X ==> Carica l'accumulatore con il contenuto della locazione di memoria di indirizzo \$0012 più il valore contenuto in X.

STA *ALFA,X ==> Deposita l'accumulatore nella locazione di memoria data dall'indirizzo "ALFA" aumentato di X locazioni.

Per il modo di indirizzamento di pagina zero indicizzato in Y, sostituire Y a X.

INDIRIZZAMENTO RELATIVO: In questo modo di indirizzamento, che riguarda solo i salti condizionati o di "BRANCH", si specifica il numero di byte da saltare rispetto all'indirizzo corrente di programma:

BCC ALFA ==> Salta ad "ALFA" se il carry si trova a livello logico ZERO.

BBS5 PORTB BETA ==> Salta a "BETA" se il bit cinque della locazione di memoria "PORTB" si trova a livello logico UNO.

INDIRIZZAMENTO INDIRETTO: In questo modo di indirizzamento, si specifica quale operando su cui si vuole operare:

JMP(\$5000)

Nell'esempio viene eseguito un salto indiretto alla locazione di memoria il cui indirizzo si trova alla posizione \$5000 (parte bassa) e \$5001 (parte alta).

INDIRIZZAMENTO INDIRETTO PRE-INDICIZZATO IN X: Si tratta di un modo di indirizzamento che differisce dall'indiretto puro per due aspetti:

- 1) La coppia di byte contenenti l'indirizzo può essere soltanto in pagina ZERO.
- 2) All'indirizzo specificato nell'istruzione viene aggiunto prima dell'uso, il contenuto del registro X, e viene così letto il contenuto dell'indirizzo ottenuto.

LDA (\$50,X)====> Carica l'accumulatore con il byte presente nella locazione di memoria

\$0050+X (parte bassa) e \$0051+X (parte alta) di indirizzo
"PTR"+X (parte bassa) e "PTR"+1+X (parte alta).

INDIRIZZAMENTO POST-INDICIZZATO: Si tratta di un modo di indirizzamento che differisce dall'indiretto puro in due aspetti:

- 1) La coppia di byte contenenti l'indirizzo può essere soltanto in pagina ZERO.
- 2) All'indirizzo finale viene aggiunto il contenuto del registro Y (post-indicizzazione).

LDA (\$50),Y====> Carica l'accumulatore con il byte presente nella locazione di memoria

il cui indirizzo è dato dal contenuto della locazione \$0050 (parte bassa) e \$0051 (parte alta), contenuto aumentato del valore presente nel registro Y.

STA (PTR,X)====> Deposita l'accumulatore nella locazione di memoria dal contenuto dell'indirizzo "PTR" (parte bassa) e "PTR"+1. (parte alta), contenuto aumentato del valore presente nel registro il cui indirizzo

INDIRIZZAMENTI ORIENTATI AL BIT:

Sono due modi di indirizzamento che riferiscono l'azione svolta dall'istruzione ad un particolare bit nel byte di operando. L'istruzione "RMB5 *PORTA" azzerava il bit 5 della locazione di memoria "PORTA"; appartengono a questo modo di indirizzamento tutte le istruzioni di branch su bit SET/RESET, esempio:

BBS2 *PORTA LOOP

Che significa: "salta a LOOP se il bit 2 del PORTA è set (UNO)".

Un sistema di sviluppo per il 6501 è ovviamente necessario per generare il firmware da installare, in Eprom (o Rom) sul Single-Board computer; per questo sarà opportuno eseguire una verifica della funzionalità del programma steso (fase di DEBUGGING).

IL PROGRAMMA ASSEMBLER DELL'ACQUISITORE DATI

Il software realizzato è costituito da vari blocchi che hanno funzioni diverse. Il programma procede eseguendo in sequenza le differenti SUB-ROUTINE. Parte dalla ricezione di una RICHIESTA DATI e procede con UNA TRASMISSIONE DATI. Le operazioni avvengono a condizione che non si verifichino errori durante la comunicazione con il PC (impossibilità di comunicare, incompatibilità nel formato o nel protocollo, interferenze ecc.).

Progettando un programma, si inizia sempre dichiarando i SIMBOLI che verranno utilizzati, esempio lo "start address del programma" con sintassi del tipo:

```
PROG EQU $E000 ; Start address programma.  
PORTA EQU $0 ; Registro 6501Q.  
CK_SUM EQU $50 ; Variabile calcolo check-sum dati seriali.
```

La prima istruzione che viene eseguita dal programma attiva il Flag I del registro di stato CPU rendendo la stessa insensibile alle interruzioni di programma (non usate nel nostro caso). Viene successivamente inizializzato lo stack pointer attraverso l'istruzione TXS (Trasferisci il contenuto di X nello Stack; X viene pre-caricato al valore \$FF). A questo punto la subroutine "RS_INIT" inizializza il canale seriale programmandolo nel modo 4800 Baud (velocità di trasmissione), No parity, 8 Bit, 1 Stop. Si noti che la "RS_INIT" non altera i registri della CPU perchè li salva nello stack prima di eseguire la sua funzione e li recupera al termine. Attraverso l'istruzione RTS termina l'esecuzione della subroutine "RS_INIT" ed il flusso ritorna al programma principale (istruzione immediatamente successiva alla chiamata della sub-routine e appena eseguita).

La parte di programma che segue è ETICHETTATA "MAIN_0" e attiva un led di segnalazione attraverso il PORTB (PB0) che resta acceso fino a quando non viene ricevuta una richiesta dati; per la sua accensione, si portano i BIT del PORTB al valore esadecimale \$FE.

Viene quindi caricato il registro X con il valore zero e si passa al punto chiamato "MAIN_1" dove un'altra subroutine ("JSR RS_GETCH") ha il compito di attendere i dati seriali entro il tempo massimo prestabilito di circa 0.5 sec ("time-out").

L'esecuzione di questa sub-routine immette il carattere ricevuto nel registro A e pone il flag C a UNO, se nessun carattere viene ricevuto entro il tempo utile stabilito il flag C è posto a ZERO.

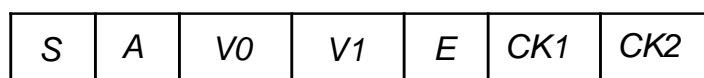
Il tempo di 0.5 secondi è dato dal decremento della variabile di programma "RS_TIMER" pre-caricata al valore \$9C40 (40000 decimale); la variabile "RS_TIMER" è usata come "contatore di loop" e determina l'esecuzione di un massimo di 40000 test sulla condizione di carattere ricevuto (sequenza che comporta la temporizzazione di "time-out").

Al ritorno dalla "RS_GETCH" il carry viene controllato dall'istruzione BCC: se trovato a livello logico ZERO, il programma ritorna a MAIN_0 (dove, tra l'altro, riaccende il Led) e ricomincia una nuova sequenza di attesa richiesta dati, al contrario (Carry ad UNO), il programma deposita il dato ricevuto nella posizione di "RX_BUFFER" indicata dal registro X. Subito dopo X viene incrementato di uno e confrontato con il valore 5 fino a quando X è inferiore a tale valore il programma ritorna a MAIN_1 e continua ricevendo caratteri e depositandoli in "RX_BUFFER" (sempre con indice X). Dopo una corretta ricezione di richiesta dati, il buffer "RX_BUFFER" contiene:



Viene quindi effettuato un controllo sui caratteri ricevuti: in caso di correttezza il programma il programma risponde con "ACK" ("acknowledge", cioè "richiesta dati correttamente ricevuta"), "NAK" ("not acknowledge") in caso di errore. Il controllo viene effettuato confrontando i caratteri ricevuti con i corrispondenti valori ASCII: "S"=53, "R"=82, ed "E"=69. Viene quindi calcolato il "check-sum" dei primi tre caratteri ricevuti e confrontato con gli ultimi due.

Se non vi sono errori, viene organizzato il pacchetto dati. Viene caricato in "TX_BUFFER"+1 il carattere 'S' iniziale; viene letto il PORTA (input digitali) isolati i BIT 0-5 e depositato il risultato in "TX_BUFFER"+2, vengono effettuate le letture dei canali analogici 0 e 1 (conversione A/D) e depositate nelle posizioni 3 e 4 di "TX_BUFFER". Viene poi immesso il carattere di chiusura dati 'E' in "TX_BUFFER"+5, calcolato il check-sum globale depositato infine nelle posizioni 6 (parte bassa) e 7 (parte alta) di "TX_BUFFER", che contiene così:



- S = Start iniziale
- A = Stato interruttori
- V0 = Stato canale 0
- V1 = Stato canale 1
- E = End finale
- CK1 = Check Sum parte bassa
- CK2 = Check Sum parte alta

Ora non resta che inviare il pacchetto dati al PC; una volta inviati, viene attesa la risposta "ACK"; se questo non avviene entro il tempo utile la comunicazione viene annullata e il programma riparte da MAIN_0, altrimenti ripete la trasmissione.

La trasmissione viene ripetuta solo se arriva un carattere NAK.

La fase di conversione del segnale analogico viene ripetuta due volte, questo per leggere e convertire i due canali.

Nelle ultime righe di programma sono riportati gli indirizzi dei vettori di reset e di interrupt, vettori che hanno origine all'indirizzo \$FFFA.

Il programma realizzato (programma simbolico) tradotto in linguaggio macchina è stato poi trasferito in EPROM.

Per la traduzione si è usato un assembler che ha trasformato il testo sorgente (data_acq.asm) in obj (data_acq.obj) e un linker che ha tradotto il file obj in linguaggio macchina puro (data_acq.mac).

LA CONVERSIONE

Convertire significa trasformare una grandezza, in un'altra. La conversione viene usata in molti campi, come in quello matematico, fisico, o elettronico.

Nel campo matematico l'operazione di conversione viene usata per convertire un numero in base generica in un'altro numero in base 10, o viceversa. Le basi più usate sono quelle decimali, esadecimali, ottali e binarie. Per convertire un numero da una base generica in base 10 bisogna prendere le singole cifre che compongono il numero, partendo dalla cifra più a destra si moltiplica questa cifra per la base del numero da convertire elevata zero poi si prende la seconda cifra e la si moltiplica ancora per la base elevata alla uno, e così via incrementando sempre l'esponente di uno mano a mano si prendono cifre di peso maggiore, poi si sommano i vari prodotti. Esempio convertiamo il numero 143 ottale, in decimale:

$$143_8 = 3 \cdot 8^0 + 4 \cdot 8^1 + 1 \cdot 8^2 = 99_{10}$$

Nel campo fisico la conversione viene utilizzata per convertire grandezze fisiche in altre grandezze. È molto usata la conversione in energia elettrica tramite i trasduttori, che convertono una grandezza fisica non misurabile direttamente in una energia elettrica. Questa conversione è molto usata perché l'energia elettrica è più facile da misurare, gestire, elaborare, tramite apparecchi analogici o digitali.

I trasduttori si dividono in attivi e passivi. Gli attivi sono quelli che effettuano essi stessi la conversione perciò funzionano da veri e propri generatori elettrici. I passivi sono quelli la cui funzione è limitata al controllo della conversione di energia, perciò l'energia d'uscita è fornita da un generatore ausiliario.

Nel campo dell'elettronica la conversione viene utilizzata per rendere un segnale comprensibile, o misurabile, per un qualsiasi strumento di misura o di elaborazione.

I circuiti per convertire i segnali analogici in segnali codificati sotto forma digitale, e viceversa, sono diventati di grande importanza nei sistemi elettronico-informatici, perché le connessioni di varie apparecchiature che comunicano fra loro richiedono mezzi adatti a tradurre da una forma all'altra.

In generale si verifica nei sistemi di misura e di controllo, le uscite dei sensori forniscono segnali di tipo analogico, i quali debbono essere tradotti in forma numerica, per subire successive elaborazioni. Di qui la necessità di disporre di sistemi di conversione.

una alternativa a questi sistemi ibridi, che comprendono contemporaneamente unità analogiche e digitali è quella di sviluppare sistemi tutti digitali. Molti sforzi sono concentrati per sviluppare nuovi sensori, che abbiano uscite digitali, e nuovi elementi di controllo e azionamento, che abbiano ingressi digitali.

Tuttavia, nonostante qualche risultato ottenuto nel campo dei programmi di ricerca spaziale e militare, questa alternativa rimane, per ora, soprattutto una tendenza di ricerca e non è ancora di pratica attuazione.

Attualmente i sistemi elettronici sviluppati per esercitare una predeterminata funzione comprendono, nella catena di blocchi funzionali elementari, sia componenti analogici che digitali.

Sarà quindi necessario disporre di componenti, o dispositivi, in grado di effettuare la conversione del segnale da digitale in analogico, oppure, da analogico in digitale.

-I convertitori digitali analogici;

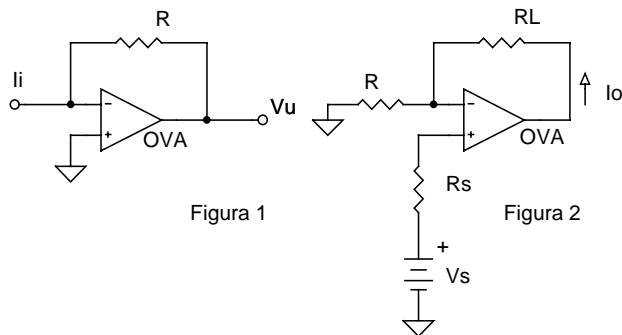
D A C (Digital Analog Converter), oppure D/A;

-I convertitori analogici digitali;
A D C (Analog Digital Converter), oppure **A/D**;

Ma non esistono solo due tipi di conversione nel campo dell'elettronica ci sono anche conversioni che permettono di ottenere da un segnale di ingresso in corrente, un segnale di uscita proporzionale a quello in ingresso in tensione, e viceversa. Si può anche convertire una tensione in frequenza, o frequenza in tensione.

-Convertitori corrente tensione e tensione corrente

I convertitore corrente-tensione (fig 1) lo si ottiene tramite l'utilizzo di un amplificatore operazionale OVA (Operation Voltage Amplifier). Questo sistema si può utilizzare per misurare la corrente massima erogabile da un sistema elettronico, in condizioni di cortocircuito: questa tecnica è spesso usata per rilevare segnali provenienti da sorgenti con resistenza interna elevata, ad esempio i trasduttori. Collegando ai terminali del generatore un amplificatore operazionale, si viene effettivamente a creare un circuito fra di essi grazie alla presenza del cortocircuito virtuale fra gli ingressi dell'operazionale. Pertanto tutta la corrente di cortocircuito I_i del generatore scorre in R determinando una tensione di uscita V_u , si può dire che: $I_i + I_u = V_u/R$ di conseguenza essendo i due piedini dell'amplificatore allo stesso potenziale, in questo caso a massa, si può dire. guardando la direzione delle due correnti che, $I_i + I_u = 0$ sostituendo I_u si ottiene che $I_i + (V_u/R) = 0$ da questa formula si ricava V_u e si ottiene che: $V_u = -R \cdot I_i$, ed essendo I_i la corrente di cortocircuito del generatore si può dire che la tensione di uscita è direttamente proporzionale della corrente di ingresso ma cambiata di segno.



Il convertitore tensione corrente (fig 2) lo si ottiene tramite l'utilizzo di un amplificatore operazionale OVA (Operational Voltage Amplifier). In questo circuito non c'è inversione di fase tra v_s e I_o la quale determina conseguentemente nella maglia di ingresso dell'amplificatore una caduta di tensione ai capi di R tale da opporsi a V_s . Si ha quindi una reazione negativa.

Il circuito, fornisce in uscita una corrente I_o che attraversa la resistenza R_L , questa

corrente è circa uguale V_s/R .

Si può dire che il circuito eroga una corrente I_o praticamente indipendente dal valore di R_s e R_L , si comporta da convertitore tensione corrente quasi ideale con carico floating ovvero riferito a massa.

-Convertitori tensione-frequenza frequenza-tensione

Nel campo dell'acquisizione e trasmissione di segnali analogici è piuttosto diffuso l'impiego di convertitori tensione-frequenza. Essi forniscono in uscita una serie di impulsi frequenza proporzionale al valore della tensione applicata in ingresso; l'intervallo di frequenza si estende di solito da pochi hertz a 10KHz.

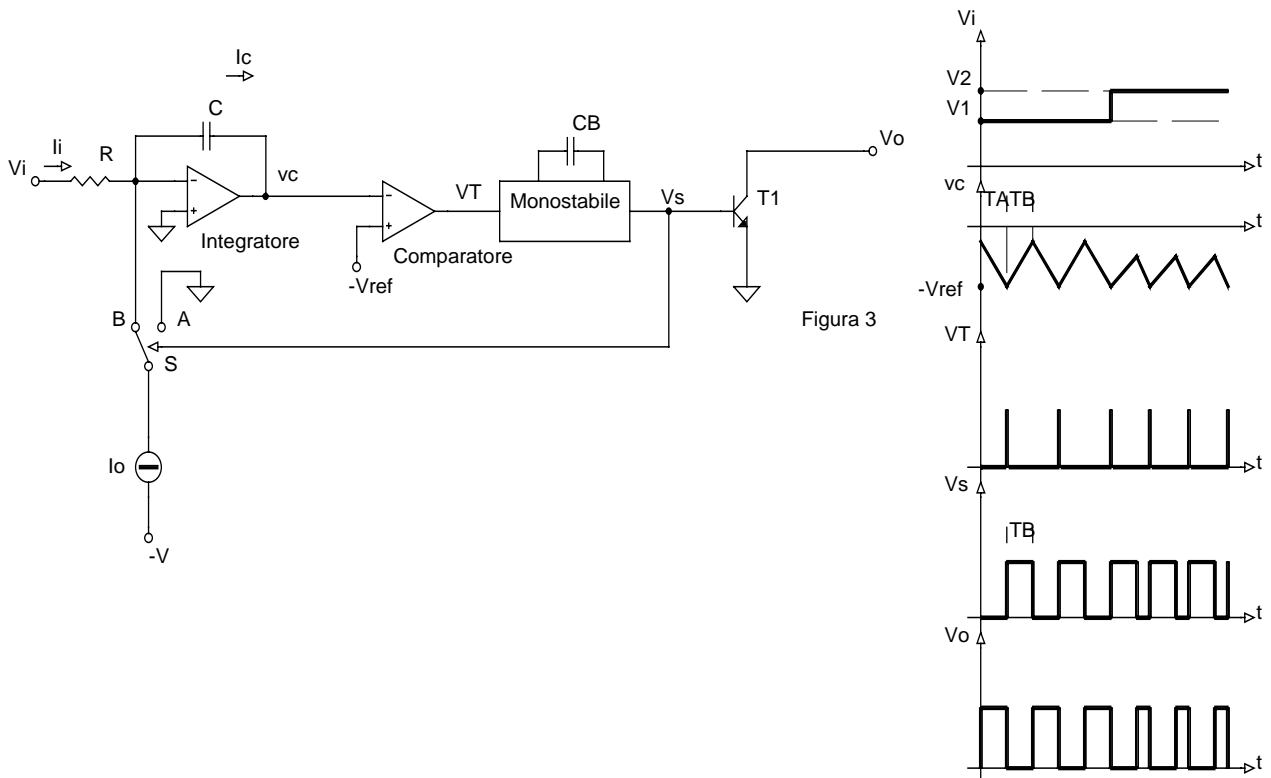
La maggior parte dei convertitori tensione-frequenza si basa sulla tecnica del bilanciamento di carica. In figura 3, si supponga che il commutatore S sia nella posizione A e che il segnale di ingresso V_i sia costante e positivo. Il condensatore C viene così percorso da una corrente di carica

$I_c = I_i = V_i/R$ e l'uscita dell'integratore risulta essere una rampa negativa esprimibile con la relazione:

$$v_c = -\frac{V_i}{R_c} \cdot t$$

Quando la tensione v_c scende al di sotto della tensione di soglia $-V_{ref}$, il comparatore commuta, portandosi a livello alto, e fa scattare il monostabile. L'impulso positivo generato dal monostabile, la cui durata T_B dipende da C_B , satura T1 mandando l'uscita V_o allo stato basso. Lo stesso impulso fa commutare S nella posizione B.

Durante l'intervallo T_B il condensatore è interessato da una corrente $I_c = I_i - I_o$. Dal momento che il generatore di corrente costante è realizzato in modo che I_o sia sempre maggiore della massima corrente di ingresso applicabile, la corrente nel condensatore si inverte; C si scarica con corrente costante $I_c = (V_i/R) - I_o$ e v_c inizia a salire facendo commutare il comparatore. Il monostabile rimane nel suo stato instabile per un intervallo di tempo fisso T_B , durante il quale il condensatore continua a scaricarsi e V_c a salire.



Al termine dell'intervallo T_B , il monostabile ritorna nello stato stabile, con l'uscita a livello basso. L'uscita V_o del convertitore si porta allora a livello alto mentre il commutatore S viene nuovamente posizionato in A. Ha così inizio una nuova integrazione del segnale V_i . In figura 3 sono illustrate le forme d'onda di V_i , V_1 e V_2 .

Consideriamo $V_i = V_1$. La quantità di cariche ΔQ_A immagazzinata dal condensatore durante la carica e quella ΔQ_B sottratta durante la scarica valgono, rispettivamente:

$$\Delta Q_A = I_i \cdot T_A \quad \Delta Q_B = (I_i - I_o) \cdot T_B$$

A regime la rampa positiva e quella negativa di v_c presentano escursioni identiche, ossia la carica accumulata da C durante T_A viene restituita durante T_B (bilanciamento di carica).

Ponendo $\Delta Q_A = -\Delta Q_B$, si ottiene:

$$T_A + T_B = \frac{I_0 \cdot T_B}{I_i} = \frac{R \cdot I_0 \cdot T_B}{V_i}$$

da questa formula si può ottenere la frequenza di uscita:

$$f = \frac{1}{T_A + T_B} = \frac{V_i}{R \cdot I_0 \cdot T_B}$$

Come si vede, la frequenza del segnale di uscita è proporzionale a V_i . Gli altri parametri che determinano il valore della frequenza sono: la corrente I_0 , che è una caratteristica propria di ciascun tipo di convertitore, R e CB (poiché quest'ultimo condiziona T_B), che sono generalmente componenti esterni. Il valore del condensatore C non influisce sulla frequenza ma solo sull'ampiezza delle escursioni di v_c . Si noti che quando V_i aumenta passando al valore V_2 , essendo T_B fisso, l'escursione positiva di v_c diminuisce in quanto diminuisce la corrente di scarica del condensatore; di conseguenza, quando S è in posizione A, la tensione $-V_{ref}$ viene raggiunta in un tempo T_A minore.

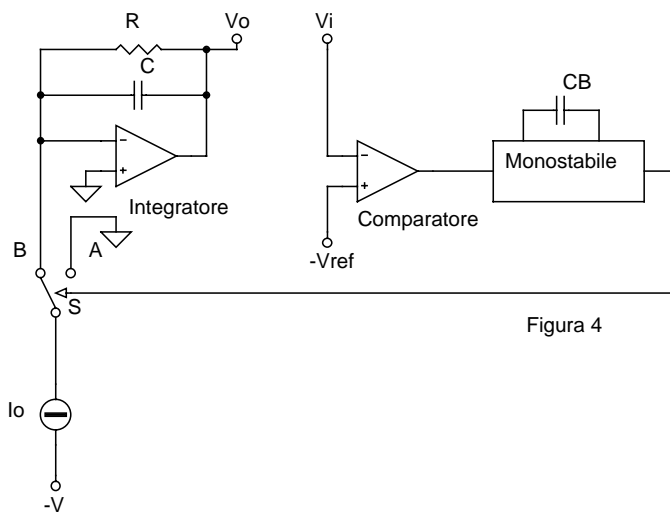


Figura 4

Quasi tutti i convertitori tensione-frequenza possono, con opportuna circuiteria esterna, lavorare come convertitori frequenza-tensione. In figura 4 è riportato lo schema di figura 3 con alcune modifiche: l'ingresso del comparatore, invece di essere collegato all'uscita dell'integratore, riceve il segnale V_i di frequenza variabile F_i . L'uscita dell'integratore costituisce ora il terminale di uscita del dispositivo.

Ogni volta che il segnale V_i scende al di sotto di $-V_{ref}$, il comparatore commuta a livello alto, pilotando il monostabile.

L'impulso positivo del monostabile porta

il commutatore S nella posizione B e lo mantiene per un tempo T_B pari alla durata dell'impulso stesso. Pertanto durante l'intervallo T_B il condensatore C dell'integratore tende a caricarsi per effetto della corrente I_0 . Alla fine dell'impulso del monostabile, il commutatore viene portato in posizione A, cosicché il condensatore inizia a scaricarsi su R.

A regime il valore medio della corrente iniettata nel condensatore è uguale al valore medio della corrente di scarica, che ovviamente tramite R è proporzionale al valore medio della tensione V_0 sul condensatore. Pertanto, al variare della frequenza del segnale di ingresso, varia proporzionalmente la corrente media iniettata in C e quindi il valore medio della tensione di uscita V_0 .

Si noti che V_i deve presentare escursioni di ampiezza compatibile con la tensione di riferimento $-V_{ref}$. Inoltre l'ingresso invertente del comparatore deve rimanere al di sotto di $-V_{ref}$ per un tempo inferiore alla durata TB dell'impulso del monostabile; infatti, se dopo il tempo TB l'uscita del comparatore fosse ancora alta, si potrebbe verificare ed errato scatto del monostabile. Per evitare questi inconvenienti, il segnale di ingresso, dopo essere stato eventualmente squadrato, ad esempio con un trigger di Schmit, viene di solito applicato al convertitore frequenza-tensione attraverso un circuito derivatore.

LA CONVERSIONE DIGITALE-ANALOGICA E ANALOGICA DIGITALE

Esistono apparecchiature elettroniche che forniscono unicamente segnali di tipo analogico: tra queste, ad esempio, molti trasduttori, che trasformano grandezze fisiche in elettriche e, nella maggior parte dei casi, il segnale prodotto è analogico, ovvero subisce variazioni proporzionali alla grandezza fisica applicata al suo ingresso.

Poiché la manipolazione dei segnali può avvenire attraverso circuiti digitali, è evidente la necessità di convertire la grandezza analogica in digitale mediante i circuiti definiti convertitori analogici digitali.

Viceversa il risultato dell'elaborazione di una rete logica deve comandare attuatori di tipo analogico. Dato che per attuatore si intende un sistema in grado di trasformare un segnale elettrico in una grandezza fisica proporzionale al segnale, è necessario che il risultato dell'elaborazione digitale sia convertito in analogico, per pilotare un attuatore.

Un sistema in grado di trasformare un risultato digitale in un segnale analogico viene definito convertitore digitale analogico.

Nella conversione digitale analogica e analogica digitale si incontra un problema fondamentale: una grandezza analogica è costituita da un insieme continuo di valori, una grandezza digitale, per la sua natura binaria, è invece costituita da un insieme finito di valori possibili.

Questo condiziona evidentemente la precisione della conversione.

Per aumentare la sensibilità e la precisione si ricorre alla quantizzazione e al campionamento.

QUANTIZZAZIONE

Il processo di digitalizzazione dei segnali analogici introduce il concetto di quantizzazione. Gli infiniti valori del segnale analogico devono essere quantizzati, ovvero raggruppati in un certo numero di fasce delimitate da livelli fissi detti livelli di quantizzazione; a ciascuna fascia di valori analogici corrisponderà un valore digitale. La distanza fra due livelli di quantizzazione costituisce il passo di quantizzazione Q a cui corrisponde il valore del bit meno significativo LSB.

Un dato digitale di n bit può esprimere 2^n valori, questo valore viene preso come fondoscala della grandezza analogica.

conseguentemente il valore minimo di scala sarà il rapporto fra il massimo valore di fondoscala e il numero di valori esprimibili dal dato digitale di n bit.

In figura 5a è illustrato un segnale a rampa V_a' variabile da 0 a 7.5 V, con i corrispondenti valori digitali. In figura 5b è riportata la forma d'onda a gradinata V_a' che si otterrebbe riconvertendo i valori digitali; come si vede, per tutti i valori di V_a' compresi ad esempio fra 2.5 e 3.5 V, il valore binario corrispondente è 011 che, riconvertito, fornirebbe V_a' uguale a 3V.

Così, per tutti i valori compresi fra 0 e 0.5V, il valore digitale corrispondente è 000. Pertanto l'errore che si commette nella quantizzazione è sempre minore o uguale a 0.5V, pari cioè al valore di $\frac{1}{2}$ LSB.

Si noti che in figura 5a i livelli di quantizzazione sono disposti in modo da avere al massimo un errore pari a $\frac{1}{2}$ LSB.

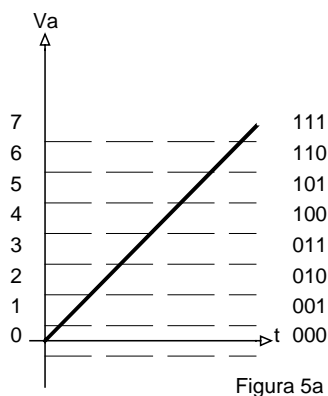


Figura 5a

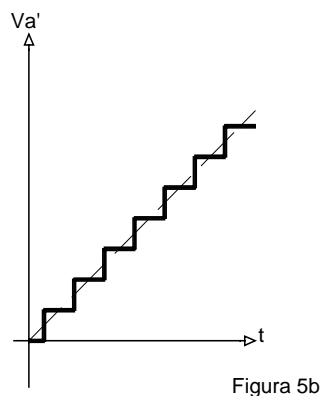


Figura 5b

In figura 6 in cui il valore digitale 000 è stata associata la fascia di tensione analogica 0-1V, evidenzia un errore di quantizzazione pari a LSB.

In un ADC i valori digitali di uscita non riproducono dunque fedelmente il segnale di ingresso ma ne danno una rappresentazione approssimata, tanto più precisa tanto più è piccolo il passo di quantizzazione Q.

Esistono convertitori a 8, 10, 12 bit, che consentono rispettivamente 256, 1024, 4096 livelli di quantizzazione.

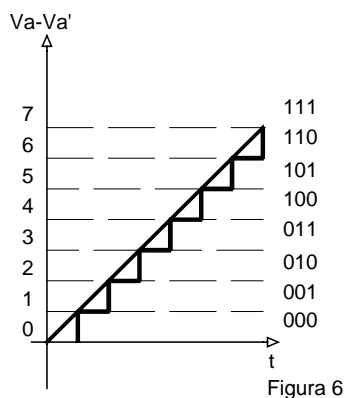


Figura 6

Il numero di bit in uscita di un convertitore A/D, così come il numero dei bit in ingresso di un convertitore D/A, viene generalmente chiamato risoluzione, perchè indica il valore minimo del segnale di ingresso che può essere riportato in uscita (il rapporto fra il valore massimo convertibile e i valori esprimibili da un dato digitale di n bit).

CAMPIONAMENTO

Utilizzato nei convertitori A/D. La conversione consiste nel prelevamento di un campione di un segnale ad un dato istante e nella determinazione del corrispondente valore digitale, che resterà fisso finchè non verrà prelevato un altro campione per una nuova conversione. La frequenza con cui il segnale viene prelevato è detta frequenza di campionamento; questa frequenza ha un'importanza fondamentale in riferimento al contenuto del segnale campionato e alle possibilità di ricostruire fedelmente il segnale analogico originario.

Il teorema del campionamento, noto come teorema di Shannon, stabilisce che la frequenza di campionamento deve essere maggiore o uguale al doppio della componente di frequenza più elevata del segnale in esame. Benchè la frequenza minima di campionamento è il doppio di quella del segnale analogico, si preferisce campionare a frequenza maggiore in modo da ottenere in uscita una rappresentazione più fedele del segnale di ingresso.

Le irregolarità nella frequenza di campionamento e la mancanza di sincronizzazione fra la frequenza di campionamento e il segnale da convertire provocano talvolta, nel segnale ricostruito, fenomeni di battimento e la comparsa di distorsioni di fase.

Questi errori sono particolarmente evidenti e dannosi quando la frequenza di campionamento è di poco superiore a quella del segnale (da 2 a 8 volte). Risulta invece facile evitarli se la frequenza di campionamento è elevata o sincronizzata con quella del segnale.

Dal momento che i convertitori A/D impiegano un tempo finito per digitalizzare un segnale analogico in ingresso, eventuali variazioni del segnale durante il processo di conversione possono determinare errori significativi. Se la variazione del segnale di ingresso durante il tempo di

conversione è superiore del bit LSB, il dato digitale di uscita può presentare un errore superiore ad 1 LSB; in pratica la risoluzione specificata per il convertitore non viene mantenuta. Questo problema può essere risolto utilizzando circuiti di campionamento e mantenimento.

CIRCUITI DI CAMPIONAMENTO E MANTENIMENTO (SAMPLE AND HOLD)

Questi circuiti sono in grado di compiere un campionamento veloce del segnale analogico e di mantenere stabile il valore acquisito durante tutto il processo di conversione. In figura 7 è illustrato un semplice circuito S/H. Durante il campionamento il segnale di controllo V_c è ad 1 logico e chiude l'interruttore analogico consentendo al condensatore C di caricarsi al Valore di V_a ; la costante di tempo di carica risulta assai ridotta perchè le resistenze in gioco sono essenzialmente l'impedenza di uscita dell'operazionale che come noi sappiamo è molto bassa. Una limitazione può essere lo slew rate dell'operazionale di ingresso, se il segnale da campionare compie escursioni ampie e veloci. Quando V_c scende a 0, l'interruttore si apre isolando il condensatore dal circuito di ingresso; C resta carico al valore campionato per un tempo idealmente infinito, data l'elevata impedenza di ingresso del secondo operazionale, e dell'interruttore aperto. Una lieve scarica può essere in realtà determinata dalla corrente di polarizzazione di ingresso dell'operazionale e dalle correnti di perdita dell'interruttore e del condensatore; per questo motivo occorre utilizzare componenti con prestazioni adeguate, ad esempio operazionali con ingressi a FET e condensatori al teflon. La scelta del valore capacitivo sarà determinata da un tempo di carica durante il campionamento molto piccolo e di una fase di scarica durante il mantenimento molto lenta. La scelta dell'interruttore può cadere su un interruttore analogico a CMOS o a JFET.

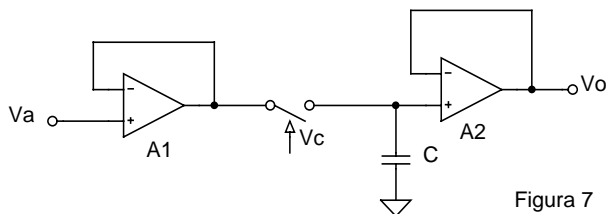


Figura 7

Nel circuito di figura 7, le inevitabili tensioni di offset dei due operazionali si sommano dando origine ad un errore di offset complessivo che, può essere, in certi casi, inaccettabile.

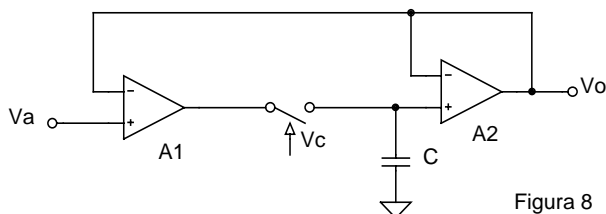


Figura 8

Un miglioramento si ottiene inserendo i due operazionali in un unico anello di reazione, come in figura 8. In questo modo l'errore di offset complessivo è costituito dal solo offset del primo operazionale perchè, a causa della reazione, la tensione di offset del secondo operazionale viene ridotta del guadagno ad anello aperto del primo operazionale. In

questo circuito durante la fase di mantenimento, quando l'interruttore è aperto, il primo amplificatore operazionale tende ad andare in saturazione, essendo interrotta la reazione; all'inizio della successiva fase di campionamento, quando l'interruttore si chiude, deve intercorrere un certo tempo prima che l'uscita del primo operazionale esca dalla saturazione. Ciò ovviamente rende questo circuito più lento di quello precedente e ne limita l'impiego al solo campionamento di segnali di bassa frequenza.

I circuiti S/H trovano impiego oltre che nell'applicazione citata, anche nel processo di distribuzione dei dati. Un convertitore D/A è collegato a più circuiti S/H, che costituiscono i canali di uscita del sistema; essi svolgono la duplice funzione di trasferire il segnale di uscita dal convertitore a diversi elementi di carico e di eliminare gli impulsi di disturbo che si generano nel segnale di uscita del DAC in coincidenza con le commutazioni dei dati in ingresso.

I circuiti S/H sono definiti da dei parametri:

-Tempo di acquisizione: il tempo richiesto affinché l'uscita raggiunga il valore finale, entro una data banda di errore, dopo che è stato dato il comando di campionamento.

-Tempo di apertura: rappresenta il tempo che intercorre da quando viene dato il comando di hold a quando l'interruttore si apre completamente; la variazione di questo parametro costituisce l'incertezza di apertura.

-Decadimento: rappresenta la variazione della tensione di uscita durante la fase di hold; questo parametro può essere specificato come variazione di tensione o con l'indicazione della corrente di perdita.

-Dinamica di ingresso: l'escursione massima della tensione di ingresso.

-Segnale di controllo: riguarda i livelli logici e la tensione necessari per portare in ON e in OFF l'interruttore analogico.

CONVERTITORI DIGITALI ANALOGICI DAC

Il convertitore DAC è un dispositivo che riceve all'ingresso un codice binario ad n bit e lo trasforma in un uscita analogica di corrente o tensione, d'ampiezza proporzionale al codice binario.

La struttura di base di un DAC è riportato in figura 9 in questo schema si riconoscono 3 blocchi fondamentali:

-La tensione di riferimento, fornita da un sistema di alimentazione di elevata precisione, costituisce la costante moltiplicativa nella operazione di conversione.

-Il convertitore digitale analogico, la cui struttura è diversa nei veri tipi di apparato.

-Il convertitore corrente tensione, ad amplificatore operazionale, trasforma i livelli di corrente prodotti dal DAC, in livelli di tensione.

L'equazione fondamentale del convertitore DAC, che esprime il legame uscita-ingresso in qualsiasi condizione di ingresso binario è:

$$V_u = -V_{ref} \cdot \left(\frac{b_n}{2^1} + \frac{b_{(n-1)}}{2^2} + \dots + \frac{b_1}{2^n} \right)$$

V_{ref} : è la tensione di riferimento ed, per una corrispondenza esatta tra il valore di V_u in V e il numero binario, è sufficiente porre $V_{ref} = 2^n (Volt)$.

$b_1 - b_n$: Sono i bit di ingresso, in particolare b_n è il bit più significativo e b_1 è il bit meno significativo del codice d'ingresso.

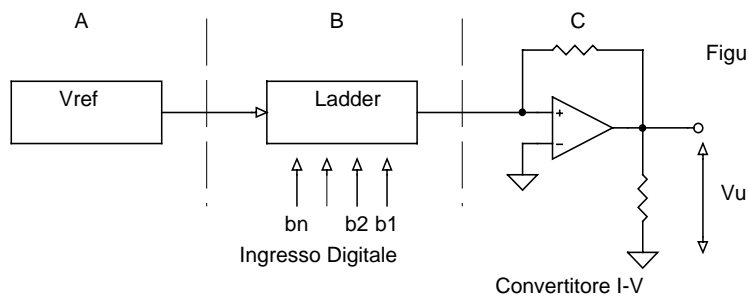


Figura 9

Il convertitore DAC è, come accennato, un dispositivo costituito essenzialmente da deviatori comandati dai bit del codice d'ingresso e resistenze di precisione. In funzione della struttura interna del convertitore si definiscono diversi tipi di

convertitori DAC.

CONVERTITORE DAC A RESISTORI PESATI

In figura 10 è illustrata la struttura circuitale di principio del più semplice convertitore DAC. L'ingresso è costituito da un segnale binario di n bit; ciascun bit controlla uno dei commutatori $S_0, S_1, \dots, S_{(n-1)}$ in modo tale che ciascun resistore viene a trovarsi collegato alla tensione di riferimento V_{ref} o a massa a seconda che il corrispondente bit si trovi al livello logico 1 o 0. L'altro estremo dei resistori si trova a massa virtuale per la presenza dell'operazionale. Si noti che i resistori presentano valori inversamente proporzionali ai pesi delle cifre binarie. Pertanto, a seconda del valore 0 o 1 dei bit di ingresso, nei corrispondenti resistori scorrerà una corrente nulla oppure una corrente inversamente proporzionale al valore del resistore e quindi direttamente proporzionale al peso dei bit.

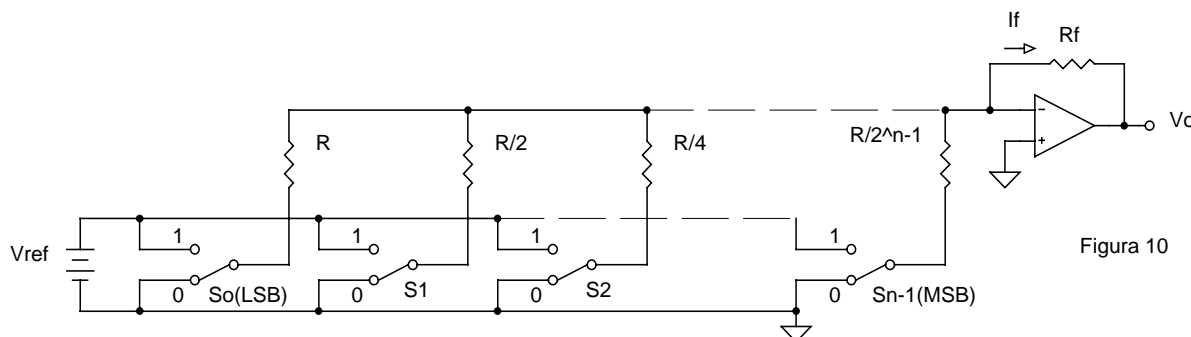


Figura 10

Ad esempio con tutti i bit a 0 ad eccezione di quello meno significativo (LSB), all'ingresso dell'operazionale si avrà una corrente $I_f = I_o = V_{ref} / R$. Da questo esempio si può ricavare la formula generale per determinare la corrente I_f con qualsiasi combinazione binaria.

$$I_f = \left(\frac{V_{ref}}{R}\right) \cdot S_0 + 2 \cdot \left(\frac{V_{ref}}{R}\right) \cdot S_1 + 2^2 \cdot \left(\frac{V_{ref}}{R}\right) \cdot S_2 + \dots + 2^{(n-1)} \cdot \left(\frac{V_{ref}}{R}\right) \cdot S_{(n-1)}$$

L'operazionale che lavora come convertitore corrente-tensione somma le correnti dei rami in cui i deviatori S sono a 1 e fornisce in uscita una tensione proporzionale alla corrente totale ovvero al valore binario del segnale di ingresso

$$V_0 = -V_{ref} \cdot \frac{R_f}{R} \cdot (2^{(n-1)} \cdot S_{(n-1)} + \dots + 2^2 \cdot S_2 + 2 \cdot S_1 + S_0)$$

Il principale inconveniente di questo convertitore è costituito dal fatto che esso richiede resistori di valore estremamente diversi. Ad esempio, in un convertitore a 12 bit, se il resistore corrispondente al bit più significativo MSB vale 2KΩ, il resistore corrispondente al LSB dovrà valere 4.1MΩ. Resistori di valore così diverso che offrono le stesse caratteristiche di precisione e stabilità termica non sono facili da realizzare.

CONVERTITORE DAC A SCALA R-2R

Il convertitore DAC illustrato in figura 11 impiega resistori di due soli valori, R e 2R; nella figura sono indicati soltanto 4 bit, ma la rete può essere espansa inserendo un numero qualsiasi di celle R-2R. Quando uno dei deviatori S è a zero vuol dire che il corrispondente resistore è connesso a massa, mentre quando S è a 1 il corrispondente resistore è collegato a V_{ref} . Si osservi che la resistenza vista da ciascuno degli ingressi S vale sempre 3R indipendente dalla configurazione dei bit in ingresso; così vale anche 3R la resistenza equivalente della rete a monte dell'operazionale.

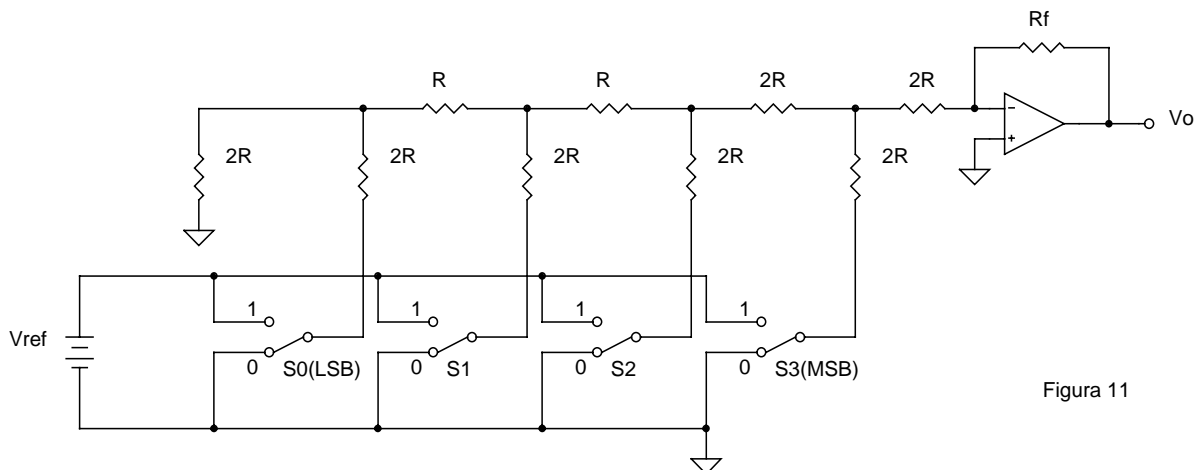


Figura 11

Considerando S_3 a 1 e tutti gli altri deviatori a 0, la corrente nel resistore 2R collegato ad S_3 vale $I = V_{ref}/3R$. La particolare disposizione della rete resistiva ripartisce questa corrente in modo tale che all'operazionale previene una corrente $I/2$. Considerando invece ad 1 solo il deviatore S_2 si ricava che la corrente che previene dall'operazionale vale $I/4$. Ripetendo questo procedimento per gli altri deviatori e applicando poi il principio di sovrapposizione degli effetti, si ottiene l'espressione della corrente I_f in funzione della posizione dei commutatori S.

$$I_f = \frac{I}{2 \cdot S_3} + \frac{I}{4 \cdot S_2} + \frac{I}{8 \cdot S_1} + \frac{I}{16 \cdot S_0} = \frac{V_{ref}}{2^4 \cdot 3 \cdot R} \cdot (2^3 \cdot S_3 + 2^2 \cdot S_2 + 2^1 \cdot S_1 + S_0)$$

L'espressione generale della tensione di uscita in funzione del valore di bit in ingresso vale:

Questo tipo di convertitore presenta due inconvenienti che $V_0 = \frac{V_{ref}}{2^n} \cdot \frac{R_f}{3 \cdot R} \cdot (2^{(n-1)} \cdot S_{(n-1)} + \dots + 2^2 \cdot S_2 + 2^1 \cdot S_1 + S_0)$

limitano le prestazioni ad alte velocità. In corrispondenza della variazione dello stato dei bit in ingresso e quindi dei commutatori, le correnti nei resistori e le tensioni nei nodi della rete subiscono variazioni; le inevitabili capacità parassite presenti devono allora caricarsi e scaricarsi rallentando così il funzionamento del convertitore. Inoltre, a causa della diversa posizione dei commutatori nella rete e del tempo di propagazione non nullo, il tempo di risposta del circuito alle variazioni del segnale digitale di ingresso non è uniforme per tutti i bit; ciò può determinare impulsi di disturbo nel segnale di uscita.

CONVERTITORE DAC A SCALA R-2R INVERTITA

Il convertitore illustrato in figura 12 elimina i problemi precedentemente citati relativi al tempo di commutazione e di propagazione. I deviatori non interrompono e non modificano la corrente nei resistori ma, a seconda dello stato dei bit in ingresso, la deviano a massa o verso la massa virtuale dell'operazionale. Il generatore V_{ref} eroga sempre una corrente costante $I = V_{ref} / R$; così anche le correnti che scorrono nei singoli resistori $2R$ sono costanti e legate fra di loro da potenze di 2, come indicato in figura.

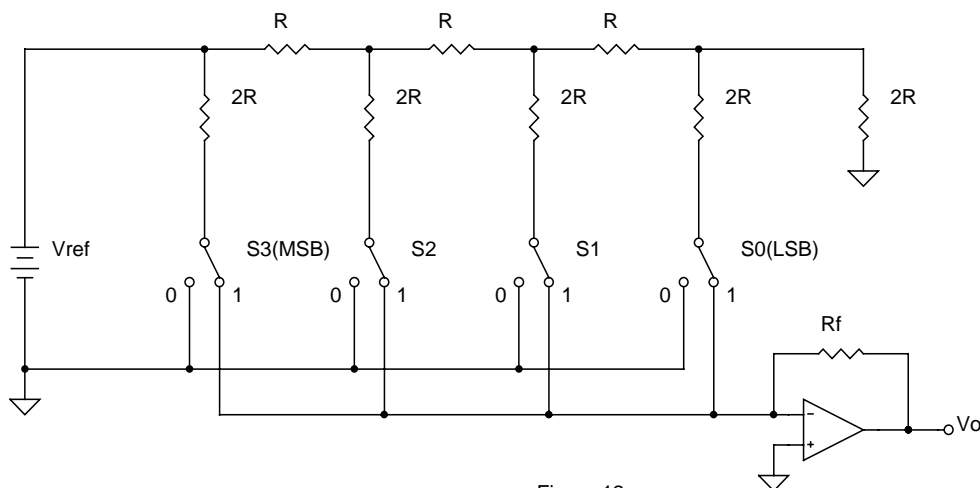


Figura 12

In questo caso la formula di I_f è uguale a quella del convertitore R-2R normale. Questo vale anche per la formula di V_0 .

CARATTERISTICHE DEI CONVERTITORI DAC

I convertitori DAC in commercio accettano in ingresso dati digitali espressi in codici diversi, binario, binario con offset, in complemento a due, BCD, con un numero di bit compreso generalmente fra 8 e 16 bit. I livelli elettrici dei dati di ingresso variano con la tecnologia con cui sono realizzati i DAC (TTL, CMOS, ECL).

I principali parametri che definiscono le prestazioni dei convertitori DAC sono:

-Risoluzione: Specifica il numero dei bit del dato digitale di ingresso e conseguentemente il numero dei valori distinti del segnale analogico di uscita.

-Precisione: Fornisce la misura della differenza fra il valore del segnale analogico di uscita reale e quello ideale, per un dato codice di ingresso; tiene conto di varie cause di errore, in particolare della non linearità del dispositivo e degli errori di guadagno e di offset della circuiteria interna.

-Linearità: In un convertitore DAC ideale, incrementi uguali del dato digitale di ingresso devono produrre incrementi uguali del segnale di uscita; pertanto la curva di trasferimento ingresso uscita ideale può essere rappresentata con una retta. L'errore di linearità esprime la massima deviazione della curva di trasferimento reale da quella ideale.

-Tempo di assestamento: È il tempo necessario affinché il segnale analogico di uscita, dopo una data commutazione degli ingressi, si assesti e si mantenga in un determinato intorno del valore finale (generalmente $\frac{1}{2}$ LSB). Il transitorio associato alla commutazione è causato dalle inevitabili capacità e induttanze parassite presenti e dalle caratteristiche dei commutatori.

-Voltage compliance: Per un dac con uscita in corrente, esprime il campo di valori consentiti per la tensione di uscita affinché siano ancora garantiti i valori di corrente specificati.

-Sensibilità alla temperatura: È legata alla deriva termica di molti elementi, quali le tensioni di riferimento interne, i resistori, i deviatori, l'amplificatore di uscita.

CONVERTITORI ANALOGICI DIGITALI ADC

I convertitori analogico-digitali trasformano un livello di tensione in un codice binario ad esso corrispondente.

Questa operazione può avvenire in molti modi, tuttavia un criterio di classificazione dei convertitori ADC si può effettuare in base alla velocità di conversione.

-Convertitori veloci (flash): In questo tipo, la codifica insegue continuamente le variazioni dell'ingresso analogico, con precisioni molto spinte.

L'impiego principale dei convertitori ADC flash si ha in trasduttori che forniscono informazioni a microprocessori; questi componenti possono lavorare infatti a notevoli velocità.

-Convertitore per misure: In questo tipo, la codifica è più lenta, ma la precisione è molto spinta. Questi tipi di convertitori trovano largo impiego negli strumenti di misura, dove la contemporaneità del dato analogico con il dato digitale a poca importanza, mentre la precisione è fondamentale.

CONVERTITORE DAC A COMPARATORI IN PARALLELO (FLASH)

In figura 13 è illustrato un convertitore con uscita a 3 bit costituito da sette comparatori, da un registro a latch per la sincronizzazione della conversione e da un codificatore.

Il segnale V_a da convertire viene applicato agli ingressi non invertenti; l'ingresso invertente di ciascun comparatore è connesso ad una rete resistiva che ripartisce la tensione di riferimento V_{ref} in otto fasce, così da fissare i livelli di riferimento o di quantizzazione.

Ciascun comparatore commuta la sua uscita ad uno quando la tensione V_a supera il rispettivo livello di riferimento.

Le uscite vengono memorizzate in sincronismo con il segnale di clock e codificate per fornire un dato digitale stabile. Il codice del dato di uscita è in questo caso binario.

Questo convertitore è in grado di convertire segnali analogici con escursione da zero a V_{ref} con un errore di quantizzazione costante pari ad $\frac{1}{2}$ LSB.

Questo tipo di convertitore, chiamato anche simultaneo o flash consente elevate velocità di conversione (nell'ordine di 10 nS) e non richiede generalmente l'impiego di circuiti S/H.

Poichè un convertitore con n bit di uscita necessita di $2^{(n-1)}$ comparatori, la realizzazione di dispositivi ad alta risoluzione comporta una notevole complessità circuitale; pertanto generalmente i convertitori simultanei hanno risoluzione limitata (tipicamente 6, 7 bit).

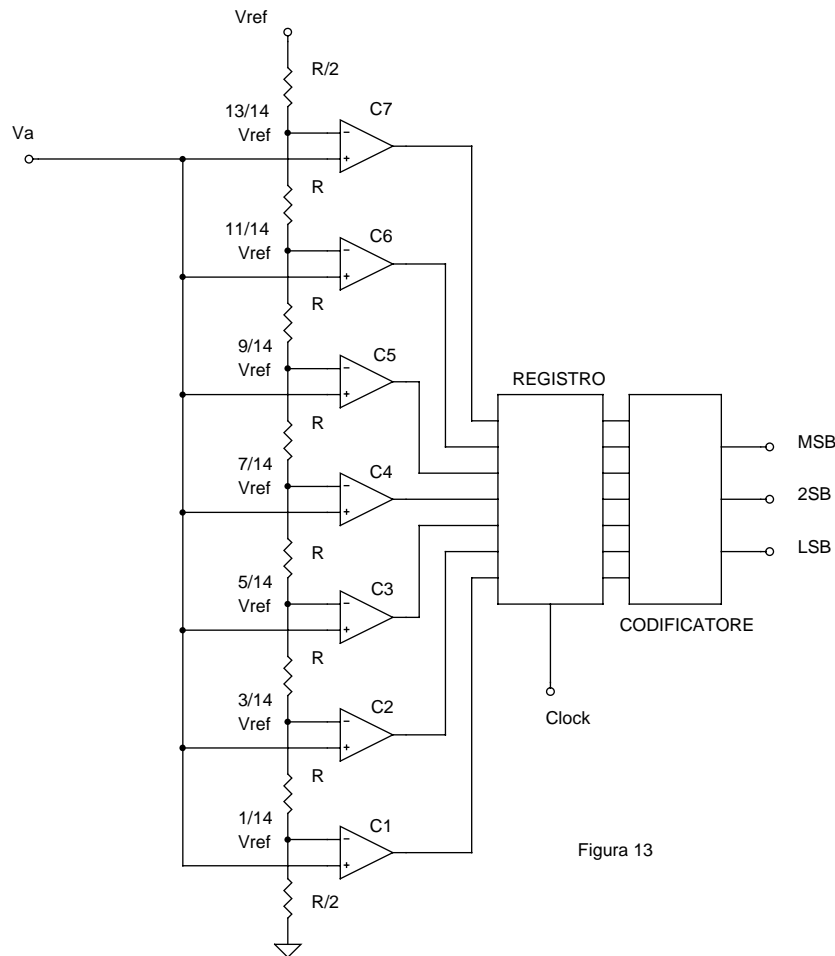


Figura 13

CONVERTITORI ADC AD APPROSSIMAZIONI SUCCESSIVE

Il metodo di conversione ad approssimazioni successive è sicuramente il più diffuso in quante consente un buon compromesso fra velocità di conversione e risoluzione.

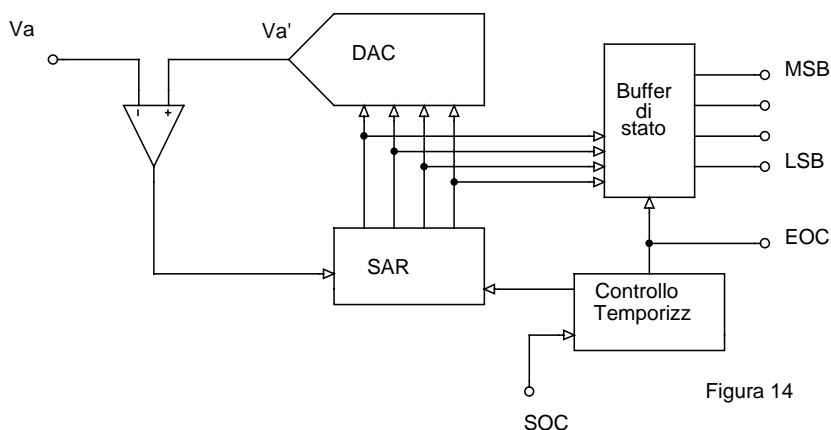


Figura 14

Nel convertitore a 4 bit in figura 14 il segnale di ingresso V_a viene comparato con precisi livelli di tensione generati dal convertitore DAC. Dopo l'applicazione del comando di conversione SOC (Start Of Conversion), che azzerà le uscite e inizializza il sistema, il registro ad approssimazioni successive SAR (Successive approximation Register) si

trova nello stato 1000. Questo dato viene presentato in ingresso al DAC che fornisce il primo livello analogico, pari a metà della tensione di fondo scala del convertitore, da confrontare con il segnale V_a .

Se $V_a > V_a'$, l'uscita del comparatore è alta e il bit più significativo del SAR, e quindi anche quello del dato di uscita, si porta a zero. A questo punto, in sincronismo con il clock, viene portato ad 1 il secondo bit più significativo del SAR, cosicché il dato presente sugli ingressi del DAC sarà 1100 oppure 0100 a seconda del risultato del confronto precedente. Il secondo confronto porta a 0 o mantiene ad uno il secondo bit del SAR e del buffer di uscita, a seconda che V_a risulti minore o maggiore di V_a' . Con procedimento analogo vengono effettuati il terzo e il quarto confronto. Alla fine della conversione, ovvero dopo quattro confronti successivi, il dato digitale contenuto nel buffer di uscita è pronto e valido; il blocco di temporizzazione segnala la fine della conversione EOC (End Of Conversion) e l'uscita può essere letta.

Utilizzando la tecnica delle approssimazioni successive si richiedono n interazioni, e quindi n cicli di clock, per convertire una tensione di ingresso in un dato ad n bit, indipendentemente dal valore della tensione stessa. Questo fatto unitamente alle buone prestazioni in risoluzione, fa preferire la tecnica ad approssimazioni successive nella realizzazione di convertitori ad media velocità, adatti ad esempio per applicazioni con microprocessori.

CONVERTITORE ADC 0808

Questi due tipi di convertitori ADC sono i più usati nel mondo dell'acquisizione dati industriale. Infatti nel nostro caso è stato utilizzato un convertitore analogico-digitale ad approssimazioni successive. Questo convertitore è ADC0808 usa una tecnologia monolitica a CMOS. A 8 bit di uscita ed 8 ingressi gestiti da un multiplexer, ed è compatibile con i microprocessori più usati.

Questo tipo di convertitore analogico-digitale utilizza il metodo delle approssimazioni successive. Al suo interno c'è un comparatore stabilizzato ad alta impedenza, il 256R divisore di tensione, un interruttore analogico (switch tree) e un registro ad approssimazioni successive.

Questo tipo di convertitore offre alte velocità, alta precisione, alta stabilità in temperatura, basso consumo, ed è in grado di mantenere il dato in memoria per un buon periodo di tempo. Questo dispositivo è stato progettato per essere inserito in processi o in macchine di controllo e consumo, e applicazioni nell'automazione.

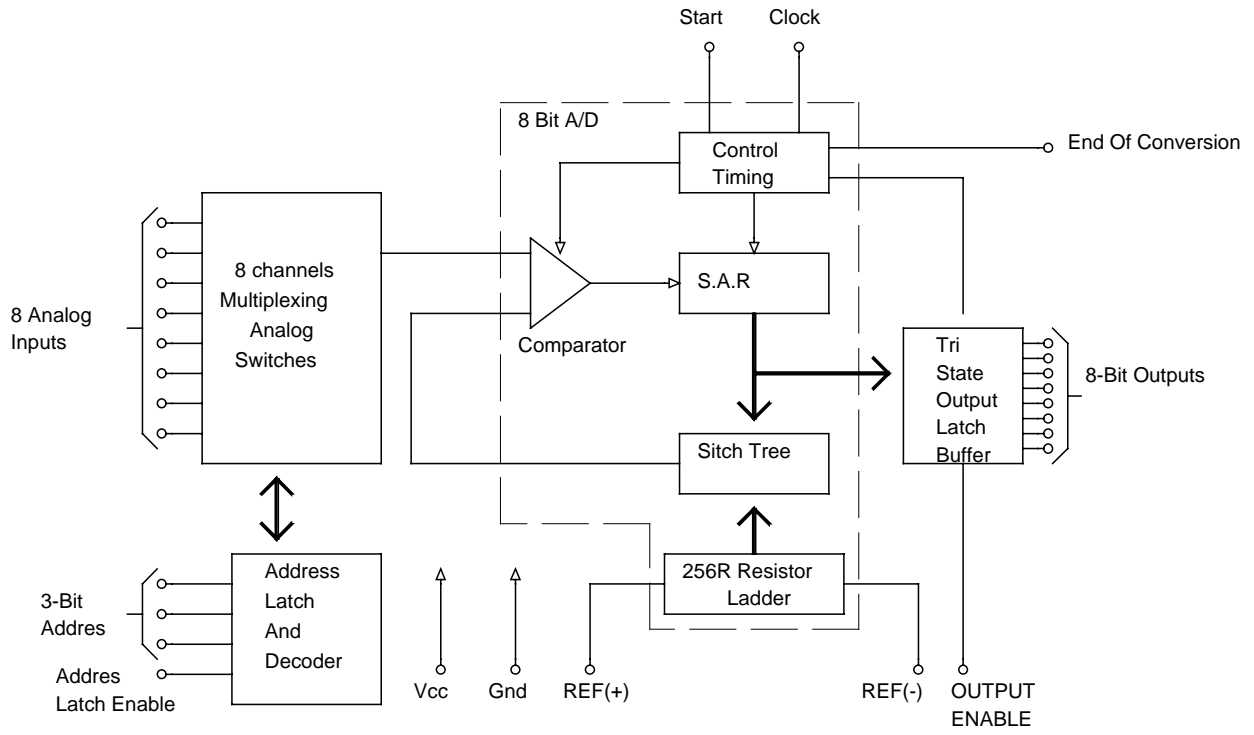


Figura 15