COMUNICAZIONI

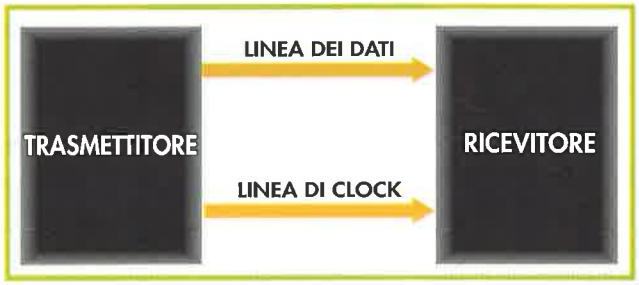
00

Internet

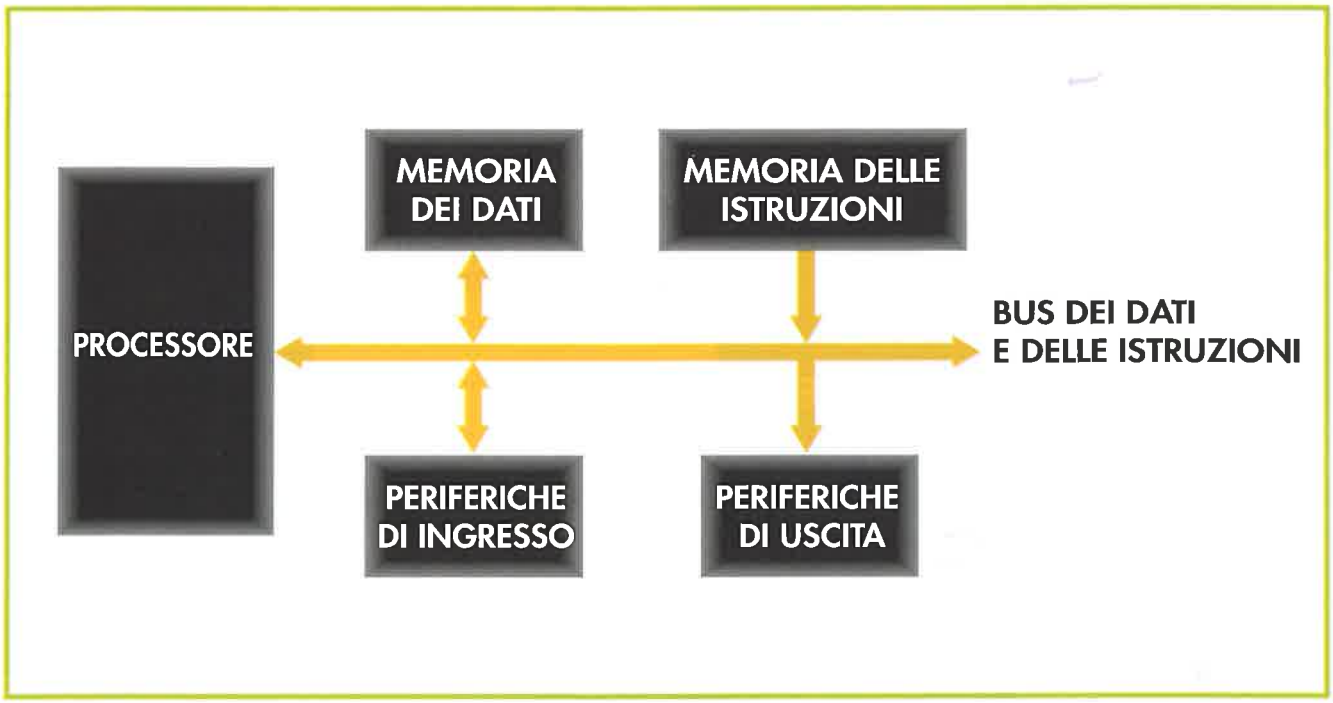
Comunicazione seriale e parallela

I processori ricevono le istruzioni dal programma in memoria, le eseguono gestendo i dati che hanno funzione di operandi e che arrivano dalla memoria dei dati o dalle periferiche di ingresso, infine generano i risultati che saranno scritti nella memoria o inviati alle periferiche di uscita. In questo continuo eseguirsi di istruzioni, il trasferimento dei dati è una parte vitale del processo.

Fondamentalmente il trasferimento di informazioni fra due dispositivi si può realizzare in due modi: in serie o in parallelo. Nella comunicazione seriale, l'informazione digitale costituita dai bit si muove su di una sola linea, di bit in bit. Il trasmettitore colloca il valore del bit da trasmettere sulla linea di comunicazione, e il ricevitore lo raccoglie. Normalmente, per sincronizzare il momento della trasmissione con quello della ricezione, si utilizza una linea ausiliaria di clock, in cui si trasmette un impulso di clock ogni volta che è presente un'informazione valida sulla linea dei dati, come rappresentato in figura.



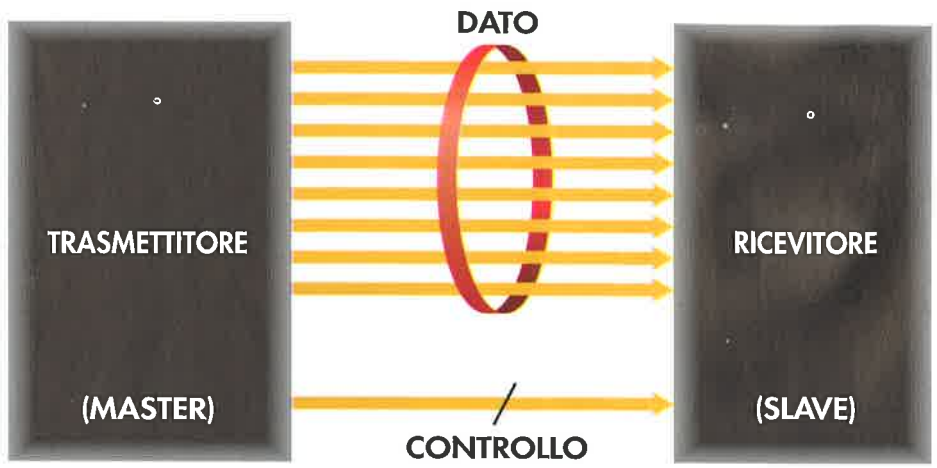
Nella comunicazione seriale sincrona, ogni volta che il trasmettitore deposita un bit sulla linea dei dati, è generato un impulso di clock per determinare la raccolta del medesimo dal ricevitore.



Per eseguire le istruzioni, il processore scambia continuamente informazioni con il resto del sistema.

COMUNICAZIONI

Presentazione

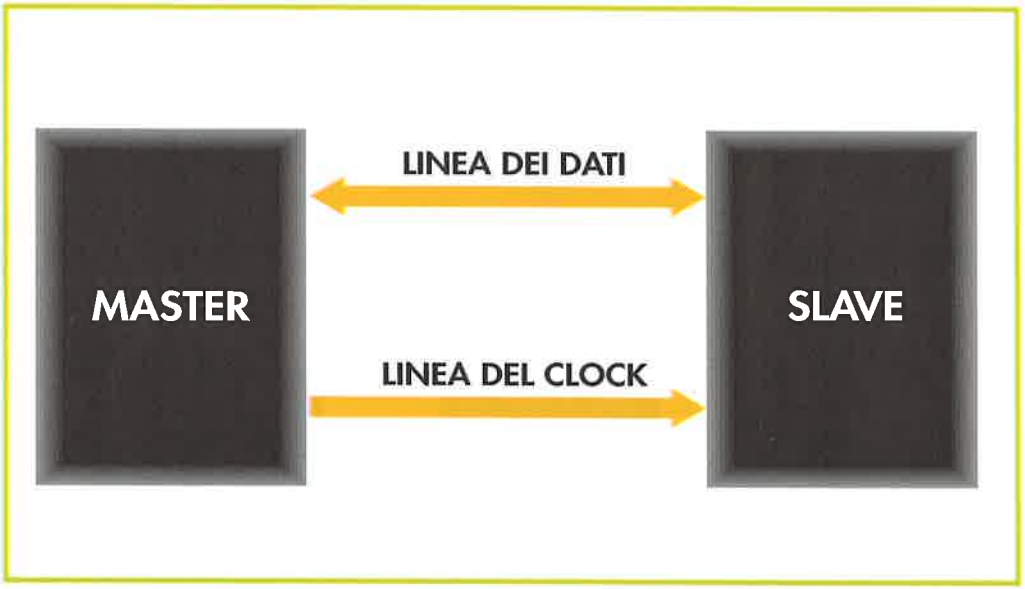


Nella comunicazione parallela il trasmettitore trasferisce diversi bit alla volta; di solito, come nel caso della figura, sono otto.

Con ogni impulso di clock si trasmette un bit, per cui la velocità di trasferimento dell'informazione in modo seriale è fortemente condizionata dalla tecnologia di fabbricazione dei componenti del sistema. Il grande vantaggio della comunicazione seriale sta nella sua semplicità e nel basso costo dell'implementazione, dato che è costituita con solo un paio di linee. Un altro vantaggio di questa architettura così semplice è dato dalla buona immunità al rumore, che assicura la pulizia dei dati e una maggiore lunghezza di trasmissione.

Nella comunicazione parallela il trasmettitore e il ricevitore sono collegati da un bus dei dati, composto da tante linee quanti bit ha il byte con cui si sta lavorando. La comunicazione parallela a 8 bit è molto diffusa; come per il trasferimento seriale, sono necessarie linee di controllo che determinano il verso della comunicazione e la sincronizzazione fra la trasmissione e la ricezione.

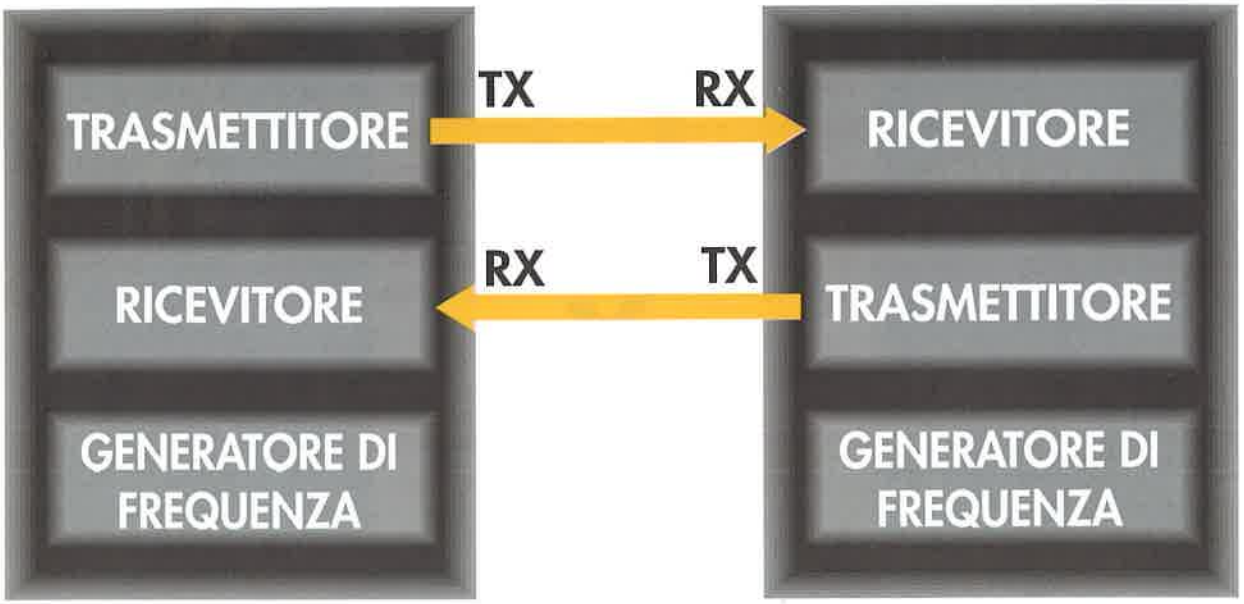
La comunicazione parallela è più rapida di quella seriale, dato che con ogni impulso di clock si trasferiscono più bit. Per contro è più complessa da gestire e la linea di collegamento è più cara, dato che ci sono più fili da collegare. Inoltre la vicinanza fra le linee dei dati la rende più vulnerabile al rumore e alle interferenze, riducendo la distanza utile di comunicazione e la sicurezza nell'interscambio dei dati.



Nella comunicazione seriale sincrona, il dispositivo che funziona come master genera gli impulsi di clock che determinano l'invio e la ricezione di ogni bit.

COMUNICAZIONI

Presentazione



Nella comunicazione seriale asincrona, ogni dispositivo contiene il proprio generatore di frequenza.

METODI DI COMUNICAZIONE SERIALE

Anche se il principio della comunicazione seriale si basa sulla trasmissione dei bit di informazione ad uno ad uno, i modi di farlo e le procedure per il colloquio fra il trasmettitore e il ricevitore differiscono a seconda dei casi, dando luogo a diverse norme e metodi, conosciuti con il nome di protocolli.

I microcontroller impiegano soprattutto quattro tipi di comunicazione seriale, che saranno descritti in seguito e che prendono il nome di:

1°. SPI (Serial Peripheral Interface)

2°. I2C (Inter-Integrated Circuit)

3°. Trasmission/Reception Universal Sincrona/Asincrona basata su USART

4°. RS-485, speciale per comunicazioni industriali su grande distanza.

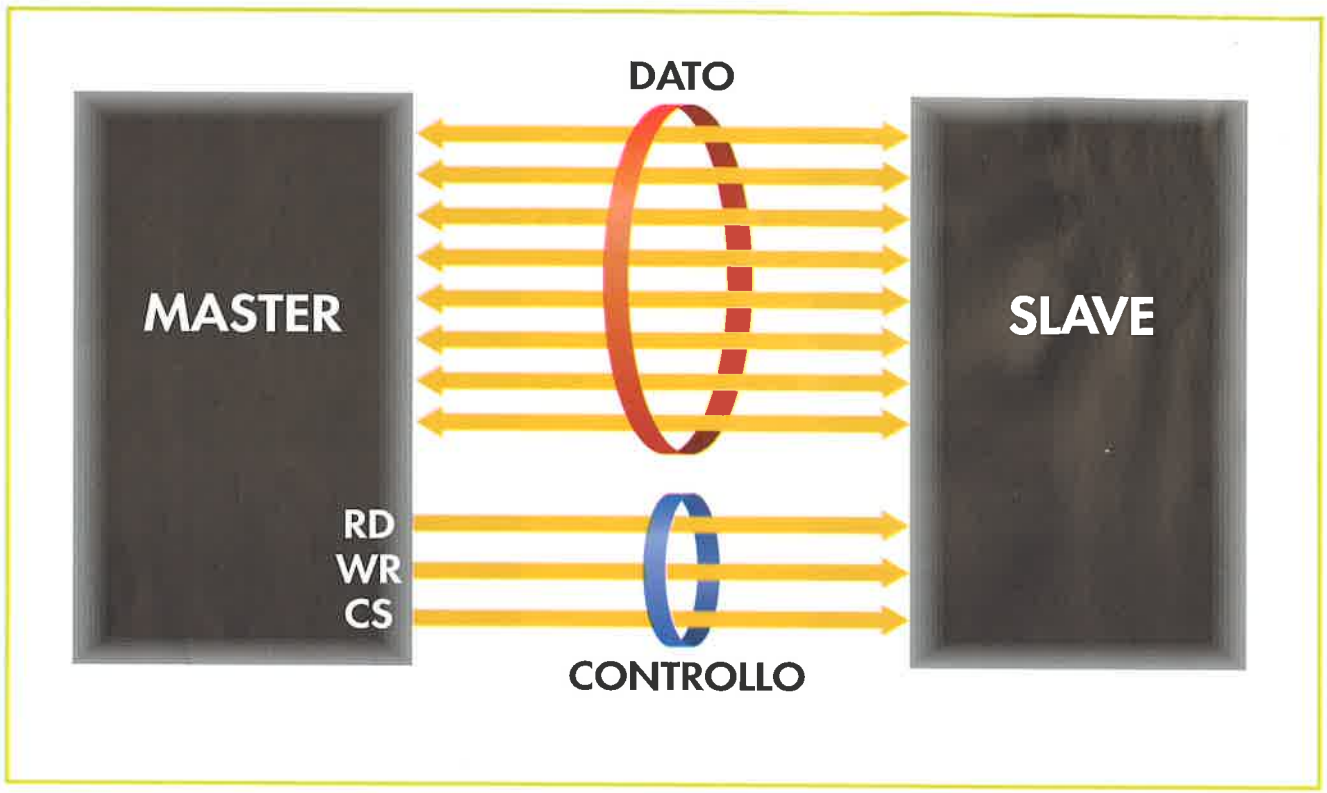
La differenza fra la comunicazione sincrona e quella asincrona è data dalla presenza o assenza del segnale di clock. Nel modo sincrono i bit vengono spostati sulla linea dei dati al ritmo degli impulsi del clock, generato da uno dei due comunicanti, che funziona da "master".

Nella comunicazione asincrona si suppone che non esista segnale di clock, però sia il trasmettitore che il ricevitore devono essere accordati nel momento di depositare e ricevere l'informazione. Per questo il trasferimento si realizza a una frequenza normalizzata in "baude-rate" che può essere di 330, 660, 1.200, 2.400, 4.800, 9.600, 19.200, 38.400 ecc. Questa frequenza coincide sia nel trasmettitore che nel ricevitore, e ognuno di essi dispone del suo generatore indipendente, come si può vedere dalla figura.

I diversi protocolli esistenti utilizzano differenti strategie di comunicazione, che permettono loro di ottimizzare differenti caratteristiche, specializzandoli per risolvere determinate situazioni. Analizzeremo le loro proprietà, insieme ai vantaggi e agli svantaggi.

COMUNICAZIONI

Presentazione



Nella comunicazione parallela il master governa il trasferimento mediante tre linee di controllo.

LA COMUNICAZIONE PARALLELA

Nella famiglia PIC1687X, ci sono dei controllori con una porta parallela slave a 8 bit implementata sul silicio, per supportare una comunicazione parallela bidirezionale con altri dispositivi compatibili. Oltre alle otto linee del bus dei dati, vengono trasmessi i segnali di controllo su altre tre linee: Lettura (RD), Scrittura (WR) e Selezione del chip (CS). Il dispositivo che funziona da master, governa lo stato di queste linee e determina il verso della comunicazione.

Nella fotografia vediamo un PC con le periferiche classiche: tastiera, mouse e stampante. I primi due normalmente comunicano in modo seriale mentre la stampante di solito utilizza la porta parallela.

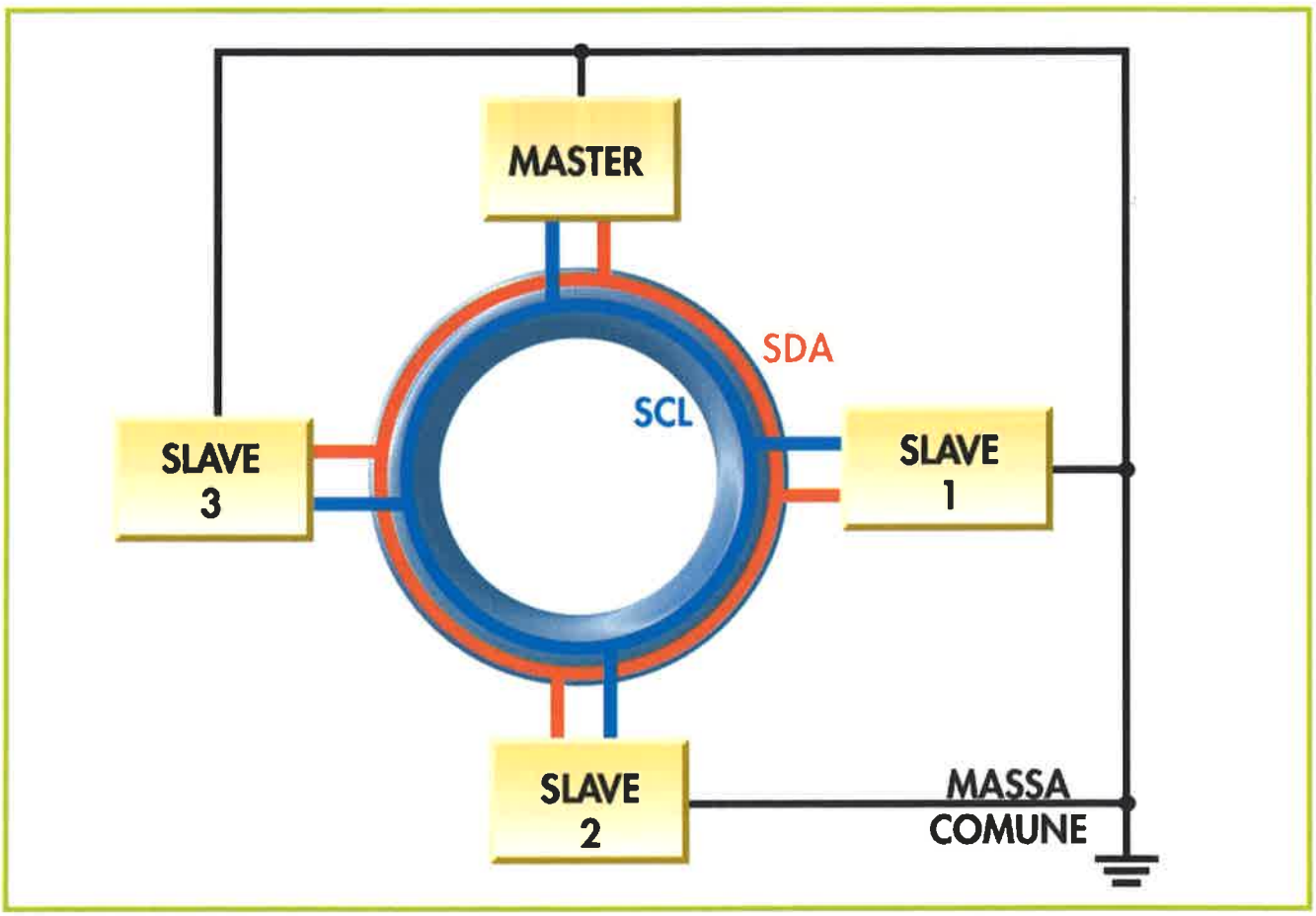
Quando il master attiva i segnali di controllo RD e CS, legge le informazioni che lo slave deposita sulla linea dei dati. Se il master attiva i segnali di controllo WR e CS, deposita un'informazione da 8 bit sulla linea dei dati, in modo che sia letta dallo slave.

Anche se la comunicazione parallela è molto più veloce della seriale, gli inconvenienti dati dalla sua complessità, dal suo costo e dalla sensibilità al rumore e alle interferenze, la rendono sconsigliabile in molte circostanze.

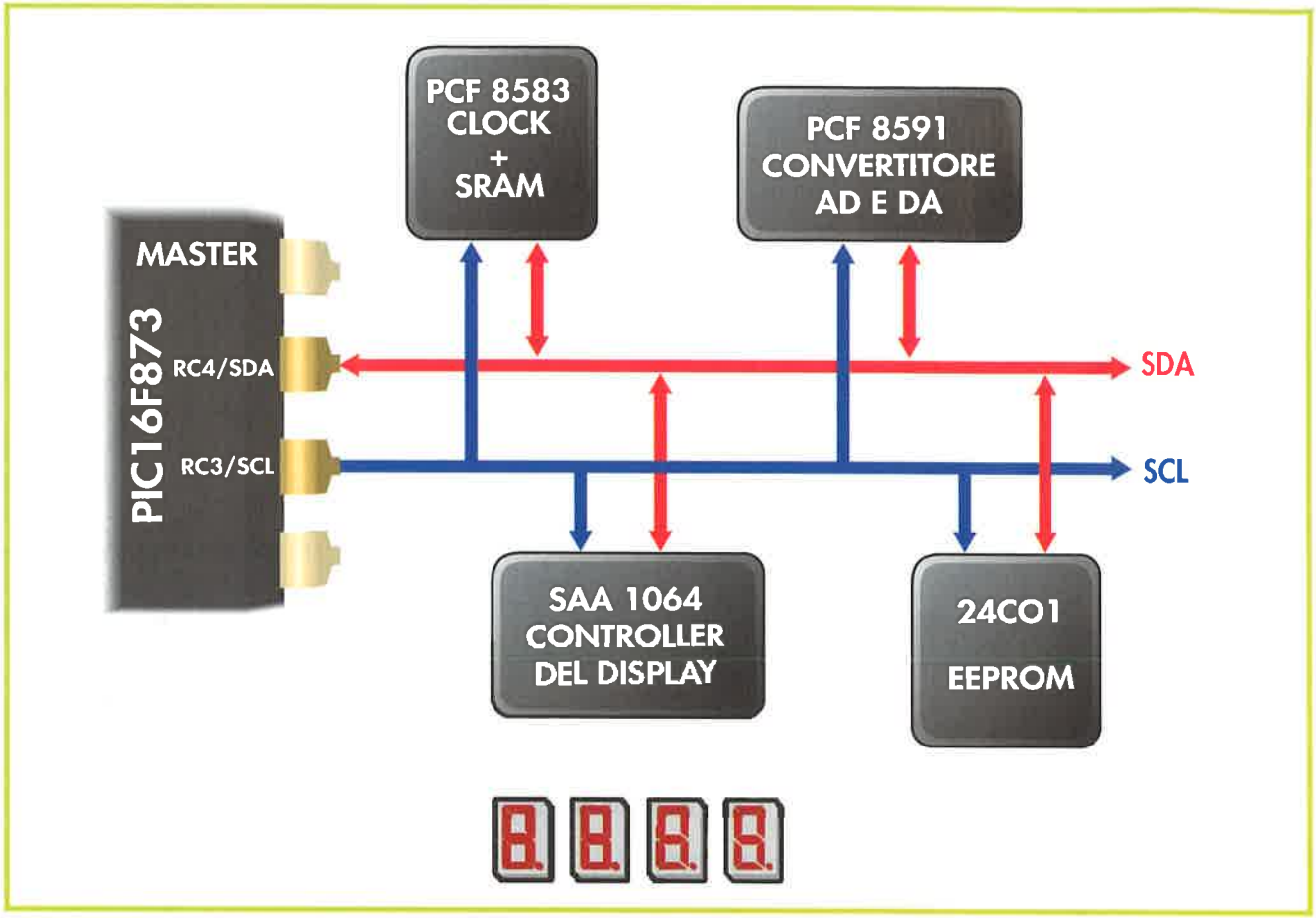
Il bus I2C (I)

Fu sviluppato dalla Philips Corporation per coprire le proprie necessità nell'implementazione di elettrodomestici e di diversi prodotti elettronici, che avevano bisogno di un'elevata interconnessione fra i numerosi circuiti integrati che richiedevano le schede PCB. Il nome I2C significa Inter-Integrated Circuits, cioè Interconnessione di Circuiti Integrati.

Il grande vantaggio di questo bus è la semplicità, dato che utilizza solo due linee per mettere in comunicazione fra loro gli elementi che sono collegati, più una massa comune. Una linea bidirezionale denominata SDA, trasferisce i bit di informazione in serie, e un'altra denominata SCL, supporta gli impulsi del clock. Uno dei dispositivi è costantemente collegato al bus, con funzioni di master, mentre i restanti hanno funzione di slave. Al master compete di determinare le caratteristiche del trasferimento e di generare gli impulsi del clock sulla linea SCL, che è unidirezionale.



Schema dei collegamenti dei dispositivi in un bus I2C.
Uno di essi ha sempre la funzione di master, mentre i rimanenti funzionano come slave.



Utilizzando solo due piedini del PIC si implementa il bus I2C e si espandono facilmente le risorse.

METODO DI LAVORO

Le specifiche originali di Philips permettevano la trasmissione dei dati sino ad una velocità di 100 Kbps, che corrispondono al modo di lavoro "standard". Recentemente sono stati realizzati dei dispositivi I2C che operano in modo "veloce", raggiungendo i 400 Kbps.

Nel protocollo I2C ad ogni slave corrisponde un indirizzo, che è utilizzato dal master quando intende comunicare con uno di essi. Per questo motivo, quando il master inizia la comunicazione con lo slave, invia l'indirizzo del medesimo e dopo specifica se vuole leggerlo o scriverlo. In alcuni dispositivi, il master, dopo queste informazioni ne deve aggiungere altre, ad esempio se desidera leggere una memoria dovrà indicare quanti indirizzi e quale di questi è il primo; infine si trasferiscono i dati.

Tutti i dati o comandi che si trasmettono con il bus I2C devono avere la lunghezza di un byte, non ci sono limiti invece per la quantità di byte che si possono trasferire; inoltre, dopo ogni byte trasferito il ricevitore del medesimo deve generare un bit di RICONOSCIMENTO (ACK#) che indica la ricezione del byte e permette di continuare il trasferimento. Il mancato riconoscimento di un byte, provoca l'interruzione di tutto il trasferimento.

Ci sono microcontroller come il PIC16F87X, che contengono scritto sul silicio il protocollo I2C. Per il loro funzionamento è sufficiente programmare adeguatamente i registri di controllo, e inviare ad un registro i byte da trasmettere, oppure contenere su di un registro i byte ricevuti dalla linea SDA, al ritmo degli impulsi di SCL. Nei microcontroller come il PIC16F84, si può utilizzare il bus I2C riservando solamente due piedini per emulare le linee SDA e SCL, e mediante le routines adeguate, trasmettere o ricevere bit supportando via software la normativa del bus.

L'uso del bus I2C nei microcontroller permette una semplice ed economica espansione delle loro risorse, ipotecando unicamente due dei piedini di I/O. La mag-

gior parte di costruttori di circuiti integrati e di memorie dispongono di famiglie di dispositivi adattabili al bus I2C: in questo modo, abbiamo a disposizione memorie SRAM, EPROM, EEPROM, convertitori AD e DA, clock in tempo reale, controller di LCD, controller di visualizzatori, porte di I/O, ecc., che possono comunicare con il microcontroller master e ampliare le sue risorse interne, come illustrato nella figura della pagina precedente.

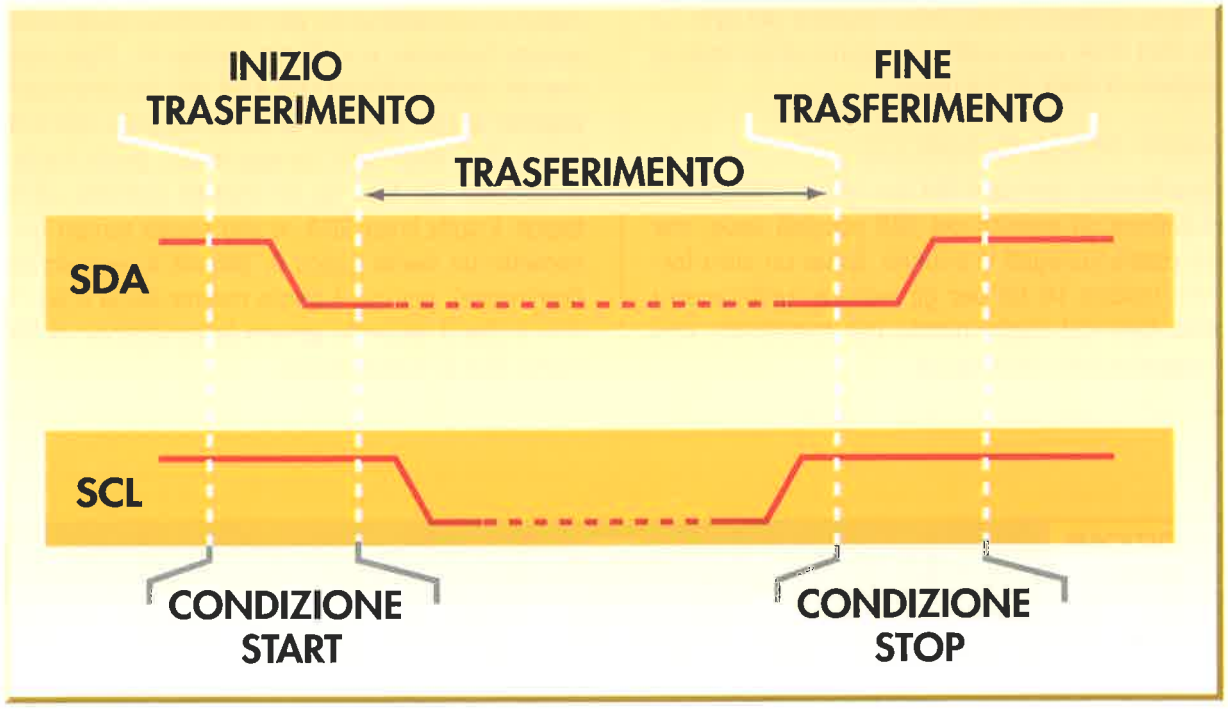
IL PRIMO BYTE DI UN TRASFERIMENTO

Il master inizia e termina tutti i trasferimenti. Per dare l'ordine di inizio, genera la condizione di start, che consiste nel produrre un fronte di discesa sulla linea SDA, mentre la linea SCL mantiene il livello alto. Per concludere un trasferimento genera la condizione di stop, che consiste nel produrre un fronte di salita sulla linea SDA, mantenendo a livello alto la linea SCL, come possiamo vedere nella figura.

Il primo byte di un trasferimento contiene l'indirizzo dello slave con cui il master vuole stabilire la comunicazione. In questo caso si utilizza un indirizzo di sette bit, l'ottavo bit è l'ordine di Lettura/Scrittura (R/W#).

Dopo la condizione di START, gli slave pongono in alta impedenza il piedino SDA, funzionando come ricevitori e restando in attesa di ricevere l'indirizzo dal master. Questi invia bit tramite la linea SDA, al ritmo degli impulsi del clock, generati in SCL. Il bit del dato deve rimanere stabile durante i fronti di SCL, in modo che gli slave possano interpretarlo correttamente. Come possiamo osservare nella figura, nel primo byte il master funziona come trasmettitore, e dopo la condizione di START invia sette bit di indirizzo e un bit finale di R/W#, al ritmo degli otto primi impulsi di clock presenti sulla linea SCL. Inoltre genera un nono impulso di clock, in modo che lo slave a cui si è fatto accesso ponga a livello basso la linea SDA, come riconoscimento della ricezione del byte (ACK#). In questo nono impulso di clock, il master funziona come ricevitore.

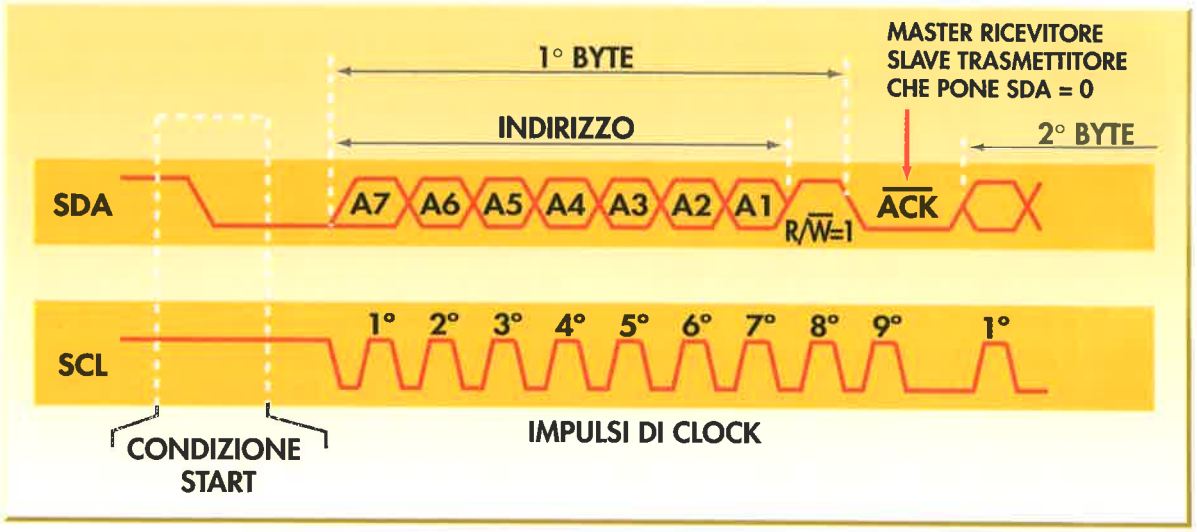
Nel primo byte del trasferimento, gli slave che funzionano come ricevitori mantengono la linea SDA in alta impedenza, in attesa di ricevere l'informazione dal master. Nel nono impulso di clock, il master funziona



Per iniziare e concludere un trasferimento, il master genera la condizione di START e quella di STOP rispettivamente.

COMUNICAZIONI

Presentazione del bus I2C



Nel primo byte il master invia l'indirizzo di 7 bit e il bit di R/W#.
Dopo, nel nono bit, funziona come ricevitore del segnale di riconoscimento dello slave (ACK#).

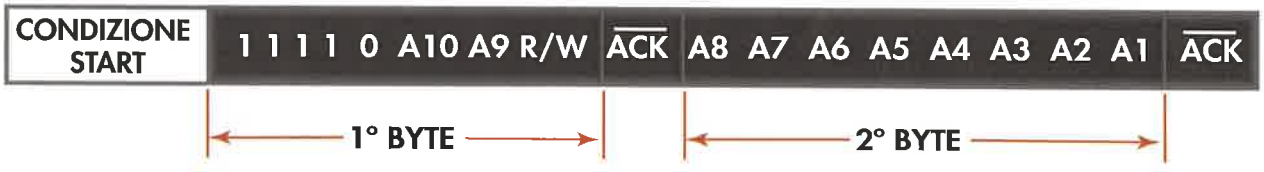
come ricevitore e mette in alta impedenza la linea SDA, in attesa che lo slave interessato la ponga a livello basso, come riconoscimento della ricezione del byte. La linea dei dati SDA, può cambiare di stato nel periodo in cui il segnale di clock SCL è basso.

FORMATO DEGLI INDIRIZZI

Nel procedimento standard del bus I2C si utilizzano 7 bit per definire gli indirizzi dei 128 possibili slave, che possono essere collegati al sistema. Esiste un altro formato che utilizza 10 bit per gli indirizzi, utilizzando i due primi byte del trasferimento per specificarli, così come rappresentato nella figura.

ARBITRAGGIO DEL BUS

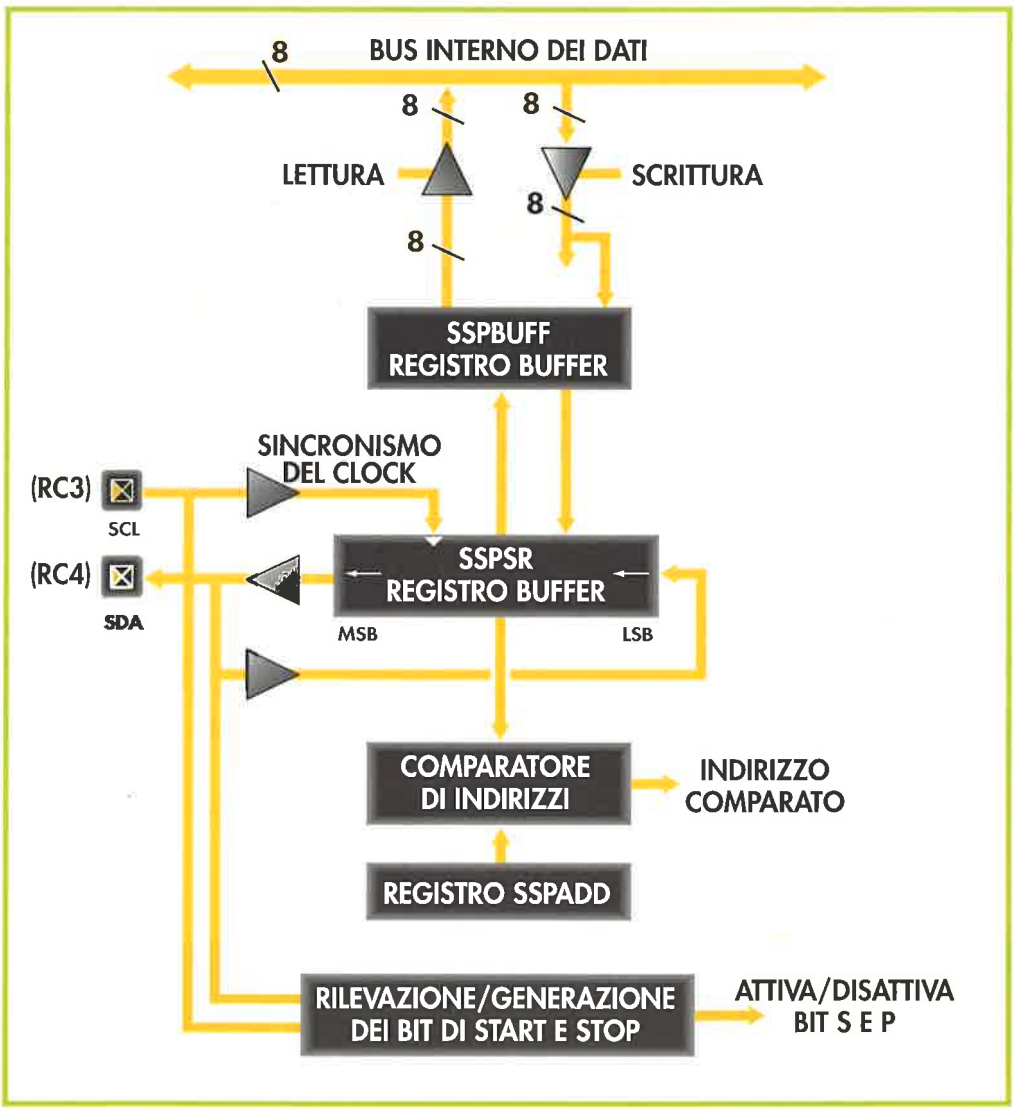
Quando vari dispositivi possono funzionare da master, esiste un procedimento che determina quale realizzerà questa funzione, e in quale momento. Ogni possibile master, deve verificare che il bit dei dati che trasmette insieme al suo impulso di clock coincida con il livello logico della linea SDA. Se non è così, perde l'accesso e il controllo del bus. Se un master immette un livello logico 1 sulla linea SDA, e allo stesso tempo un altro immette un livello logico 0, prevale il secondo bit (bit dominante), per cui il primo master lascia il bus libero sino a che il secondo genera la condizione di STOP e mette fine al trasferimento.



Quando l'indirizzo ha il formato a 10 bit, il master lo invia nei primi due byte del trasferimento.

Il bus I2C (II)

Alcuni modelli di microcontroller PIC hanno il protocollo del bus I2C implementato sul chip, e la sua gestione si realizza programmando i bit di alcuni registri. Questo è il caso dei fratelli maggiori del PIC16F84, ossia i PIC16F87X. Il PIC16F84 non ha il bus I2C implementato sul chip, e per gestirlo bisogna riservare due piedini a questo compito, e realizzare le routines software necessarie per fare in modo che le informazioni seriali entrino e escano da uno di questi, secondo il protocollo, e al ritmo degli impulsi di clock generati con l'altro piedino.



Struttura generale dei registri dei PIC16F87X incaricata di controllare i trasferimenti nel modo I2C.

Nei PIC16F87X due dei suoi piedini, il RC37SCK/SCL e il RC4/SDI/SDA, sono multifunzione e fra le altre cose possono essere utilizzati per supportare i trasferimenti del bus I2C tramite il microcontroller stesso, che può essere programmato in modo da funzionare come master o slave, e per indirizzi da 7 o da 10 bit. Il blocco che contiene l'hardware del PIC destinato al bus I2C si chiama SSP.

C'è un bit (SSPEN) nel registro SSPCON che quando viene messo a 1, determina la configurazione dei piedini RC3/SCL e RC4/SDA per il trasferimento di informazioni sotto il protocollo I2C.

Questi piedini della porta C devono essere precedentemente configurati come ingressi scrivendo i bit corrispondenti del registro TRISC.

Per controllare il modulo SSP nel modo I2C abbiamo a disposizione sei registri:

- **SSPBUF:** Buffer dei dati da inviare o ricevere.
- **SSPSR:** Registro di spostamento che converte l'informazione da seriale a parallela e da parallela a seriale.
- **SSPADD:** Registro che contiene l'indirizzo dello slave.
- **SSPSTAT:** Registro di stato.
- **SSPCON:** Registro principale di controllo.
- **SSPCON2:** Registro ausiliario di controllo.

Nella figura della pagina precedente è mostrata la struttura dei registri che gestiscono l'informazione e il controllo del modulo SSP.

Nello schema della figura il registro SSPBUFF contiene il dato che deve essere trasmesso, o che deve essere ricevuto. Il SSPSR è un registro di spostamento che realizza la conversione da serie a parallelo e viceversa. Quando il SSPSR è stato riempito con gli 8 bit, trasferisce l'informazione al SSPBUFF e si attiva il flag SSPIF. Se si riceve un altro byte in SSPR prima che la CPU abbia letto il SSPBUFF, si attiva il flag di overflow SSPOV, che è il bit 6 di SSPCON. Il registro SSPADD contiene l'indirizzo dello slave. Quando si utilizzano indirizzi da 10 bit, il byte alto dell'indirizzo contiene i due bit più significativi (A9 e A8) e si codifica con il seguente formato: 1-1-1-1-0-A9-A8-0.

IL REGISTRO SSPCON

È il registro di controllo principale per la gestione del modulo SSP nel modo I2C. La distribuzione dei suoi bit è riportata nella figura.

I quattro bit meno significativi di SSPCON, SSPM3 – SSPM0, servono per selezionare il modo di lavoro del microcontroller. Di seguito riportiamo i tre modi principali, ognuno dei quali ha diverse varianti:

1°. Modo slave con 7 bit di indirizzo.
2°. Modo slave con 10 bit di indirizzo.
3°. Modo master.

Le varianti ammesse all'interno del modo master, con i relativi codici, sono riportate nella tabella.

CODICE	MODO LAVORO
1000	Modo master bus I2C. Frequenza = Fosc/4*(SSPADD+1).
1011	Modo master bus I2C controllato da firmware.
1110	Modo master con 7 bit di indirizzo e interrupt.
1111	Modo master con 10 bit di indirizzo e interrupt.

Per attivare il modulo SSP e configurare i piedini RC3/SCL e RC4/SDA per il trasferimento I2C, dobbiamo scrivere SSPEN = 1. Se SSPEN = 0, le linee RC3 e RC4 funzionano come I/O digitali. Se si pone a 0 il bit CKP i livelli di questi piedini si configurano secondo le specifiche I2C e si attiva il clock nel modo slave. SSPOV è il flag di overflow che si pone a 1 quando si riempie il registro SSPR prima di aver letto il registro SSPBUFF, cosa che suppone la perdita dell'ultima informazione ricevuta. Il bit WCOL del registro SSPCON è un rilevatore di collisione in scrittura, e quando vale 1 significa che c'è stato un tentativo di scrivere SSPBUFF senza prima aver letto l'informazione precedente.

IL REGISTRO SSPSTAT

Questo registro, la cui struttura interna è riportata nella figura, permette la lettura di tutti i suoi bit ma solo i due più significativi possono essere scritti. Contiene i flag che ci informano sullo stato del trasferimento dei suoi dati.

REGISTRO SSPCON

WCOL	SSPOV	SSPEN	CKP	SSPM 3	SSPM 2	SSPM 1	SSPM 0

Struttura interna del registro di controllo SSPCON.

REGISTRO SSPSTAT

SMP	CKE	D/A#	P	S	R/W#	UA	BF

Struttura interna del registro di stato SSPSTAT.

Nel modo master il bit SMP vale 1 quando il controllo dei fronti di salita e di discesa è disabilitato (Standard Speed Mode), e vale 0 quando detto controllo è abilitato (High Speed Mode). CKE seleziona i livelli delle linee SDA e SCL nel modo master o multimaster.

Al verificarsi della condizione di inizio, si pone a 1 il bit S e quando si rileva la condizione di STOP si pone a 1 il bit P. Il bit D/A# prende il valore 1 quando il dato ricevuto corrisponde a un'informazione, e ha valore 0 quando si riceve un indirizzo. R/W# determina se si tratta di un'operazione di lettura o di scrittura. Il bit UA segnala la lunghezza dell'indirizzo, se vale 1 è da 10 bit, mentre se vale 0 è da 7 bit.

Infine il bit BF (Buffer pieno) si pone a 1 quando c'è un dato nel registro SPBUFF.

IL REGISTRO SSPCON2

È un registro di controllo ausiliario, i cui bit si possono leggere e scrivere, e dopo un Reset prendono tutti il valore 0.

I compiti dei bit di SSPCON2 sono i seguenti:

GCEN: Ha solo funzione in modo slave.

ACKSTAT: Quando si pone a 1 indica se ha ricevuto un bit di riconoscimento ACK dello slave.

ACKEN: Quando si pone a 1 inizia la sequenza di generazione della condizione di riconoscimento. Si cancella automaticamente via hardware.

RCEN: Si pone a 1 per abilitare il master in modo ricezione.

PEN: Si pone a 1 per generare la condizione di Stop.

RSEN: Si pone a 1 per ripetere la condizione di Start.

SEN: Si pone a 1 per iniziare la condizione di Start o Inizio.

FUNZIONAMENTO DEL MODO MASTER

Questo modo di lavoro si attiva scrivendo nel registro SSPCON il bit SSPEN = 1. Dopo aver realizzato questa operazione, l'utente dispone di 6 opzioni:

REGISTRO SSPCON2

GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN

Struttura interna del registro ausiliario di controllo SSPCON2.

- Generare la condizione di Start.
- Generare la condizione di ripetizione di Start.
- Scrivere su SSPBUFF per iniziare la trasmissione di un dato o di un indirizzo.
- Generare la condizione di Stop.
- Configurare il modo I2C in ricezione.
- Generare la condizione di riconoscimento ACK, alla fine della ricezione di un byte di dati.

Il master genera gli impulsi di clock e le condizioni di Start e di Stop. Quando il master è in trasmissione il primo byte indica l'indirizzo dello slave (7 bit più il bit R/W#). Per ogni byte che è stato ricevuto dal master, si trasmette un bit di riconoscimento ACK. La sequenza dei passi da realizzare in una trasmissione del master sarebbe la seguente:

a. L'utente genera la condizione di Start, ponendo SEN = 1 nel registro SSPCON.

b. Si pone il bit SSPIF = 1 per terminare la condizione di Start e non iniziarne un'altra prima.

c. Si carica in SSPBUFF l'indirizzo dello slave da trasmettere.

d. Si manda in uscita tramite SDA l'indirizzo dello slave.

e. Lo slave genera il bit di riconoscimento ACK, ricevuto dal master in SSPCON2.

f. Si genera un interrupt alla fine del nono impulso di clock e SSPIF = 1.

g. L'utente carica in SSPBUFF un dato.

h. Si trasmette il byte del dato tramite SDA.

i. Si riceve il bit di riconoscimento dello slave.

j. Si genera un interrupt nel nono impulso di clock e si pone SSPIF = 1.

k. Si genera la condizione di Stop ponendo PEN = 1 in SSPCON2.

l. Dopo aver completato la condizione di Stop, si genera un interrupt.

Nella figura è riportato il cronogramma di una tipica trasmissione di un PIC16F87X quando opera in modo master.

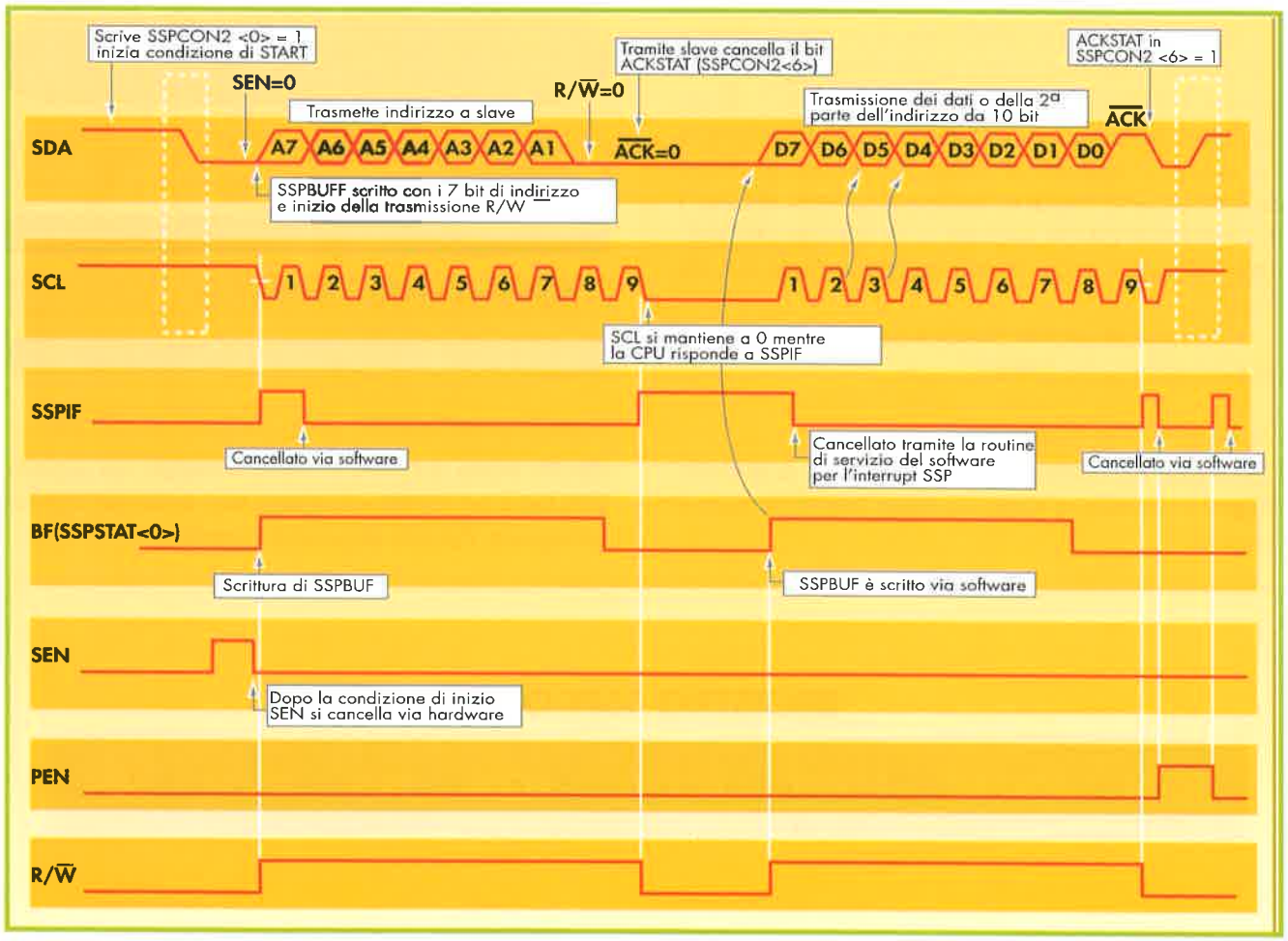


Diagramma dei tempi di una trasmissione di un PIC16F87X in modo master.

Il bus I2C (III)

Sempre più costruttori di circuiti integrati inseriscono nei loro cataloghi componenti che integrano dispositivi I2C, il che significa che questi componenti sono in grado di trasferire informazioni secondo la normativa del bus I2C, e possono essere collegati a reti con questo protocollo rispondendo agli indirizzi con cui sono stati programmati.

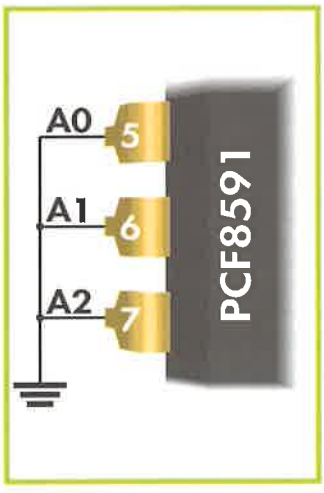
Con l'obiettivo di collegare al bus I2C diversi dispositivi uguali, nel caso che uno non sia sufficiente, gli indirizzi di questi componenti non sono totalmente definiti in fabbrica, ma gli ultimi bit sono lasciati liberi e sono collegati a piedini che devono essere collegati a terra o al positivo, per prendere valore 0 o 1. Questo è anche il caso del componente PCF8591, che contiene un convertitore A/D e uno D/A, e di cui parleremo. L'indirizzo di 7 bit a cui risponde ha i 4 bit più significativi già programmati, mentre i 3 meno significativi sono determinati dall'utente collegando i piedini A0, A1 e A2 a massa o al positivo. I 4 bit più significativi sono 1-0-0-1, quindi se i 3 piedini menzionati sono collegati a massa come mostrato nella figura, l'indirizzo completo sarà: 1-0-0-1-0-0-0.

In seguito descriveremo le principali caratteristiche del PCF8591, che è un componente molto rappresentativo fra quelli fabbricati con l'interfaccia I2C.

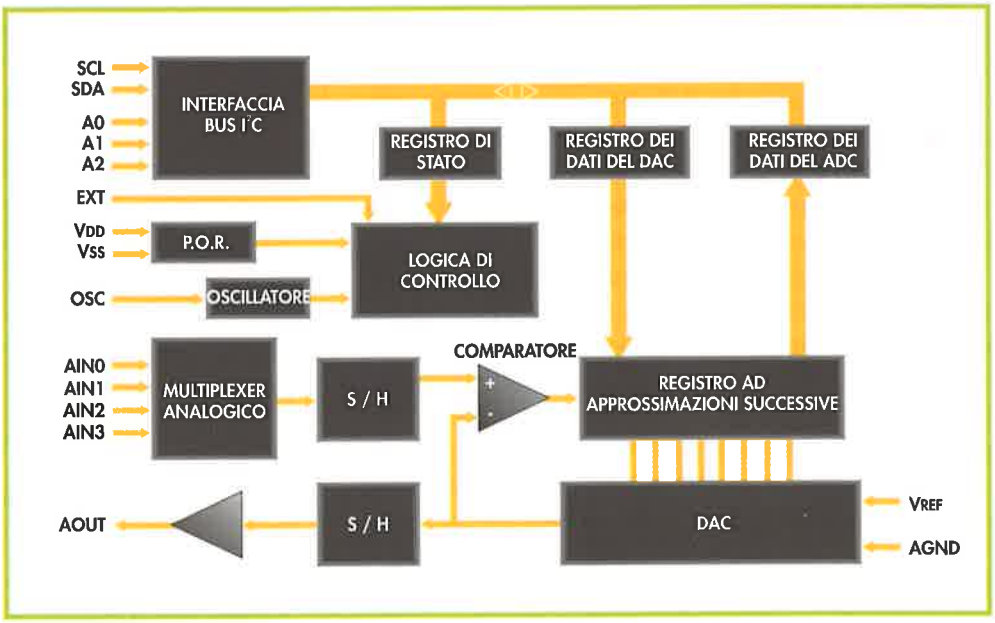
Sia questo circuito che il PCF8574A, che è una porta di I/O digitali, il PCF8583, che è un clock/Calendario + RAM e il SAA1064, che è un controllore di display a 7 segmenti, sono utilizzati nella scheda MICRO'PIC TRAINER PLUS, che verrà descritta in seguito, e che permette l'espansione del sistema di sviluppo MICRO'PIC TRAINER della ditta Ingenieria de Microsistemas Programados S.L.

PCF8591: CONVERTITORE A/D E D/A

Si tratta di un circuito integrato CMOS a basso consumo, che dispone di un'interfaccia con il bus I2C e ha tre linee per determinare via hardware uno degli indirizzi fra gli 8 possibili a cui rispondere nella rete. Ha quattro



Collegando i piedini A0, A1, e A2 a massa o al positivo, si possono collocare sino a 8 dispositivi PCF8591 nel bus I2C. In questo caso l'indirizzo della figura è 1-0-0-0-1-0-0-0.



Schema a blocchi del dispositivo I2C PCF8591 che contiene un convertitore A/D e uno D/A.

Nº PIN	SEGNALE	DESCRIZIONE
1-4	AIN0-AIN4	Ingresso analogico per l'ADC
5-7	A0-A2	Linee degli indirizzi hardware
8	Vss	Negativo dell'alimentazione
9	SDA	Linea dei dati del bus I²C
10	SCL	Linea del clock del bus I²C
11	OSC	I/O dell'oscillatore
12	EXT	Selezione dell'oscillatore: "0" interno; "1" esterno
13	AGND	Massa del segnale analogico
14	Vref	Ingresso della tensione di riferimento
15	Aout	Uscita analogica del DAC
16	Vdd	Positivo dell'alimentazione

Piedinatura del PCF8591 e descrizione dei piedini.

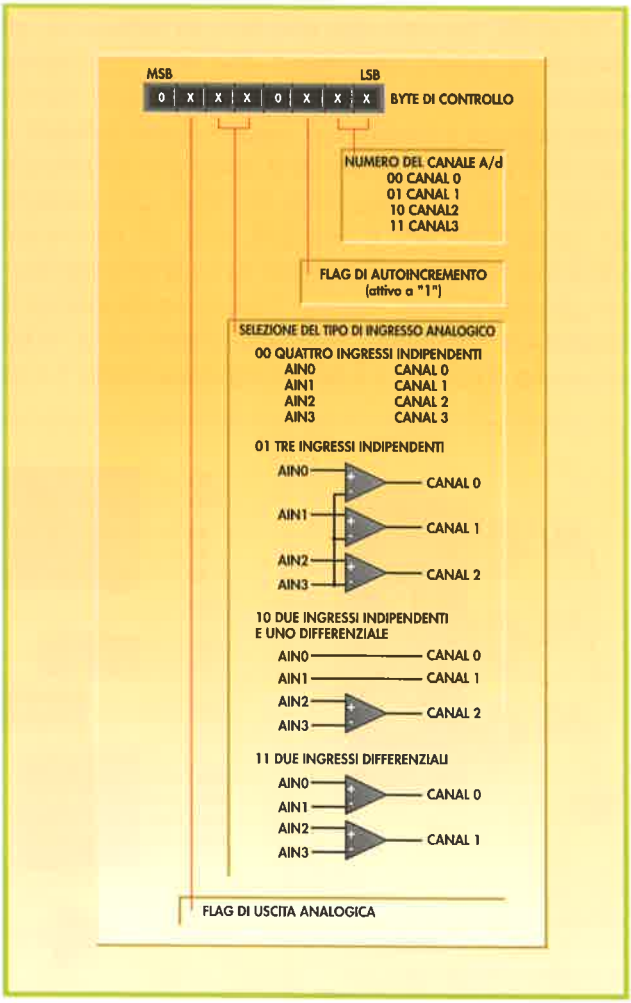
Byte di indirizzo corrispondente al PCF8591.

ingressi analogici per il convertitore A/D, e un segnale analogico per il convertitore D/A. Il margine di tensione analogica può variare fra 0 V e la tensione di alimentazione. Il PCF8591 si attiva quando riceve tramite il bus I2C l'indirizzo a cui è stato programmato. I piedini A0, A1 e A2 determinano via hardware l'indirizzo valido, secondo quanto mostrato nella figura.

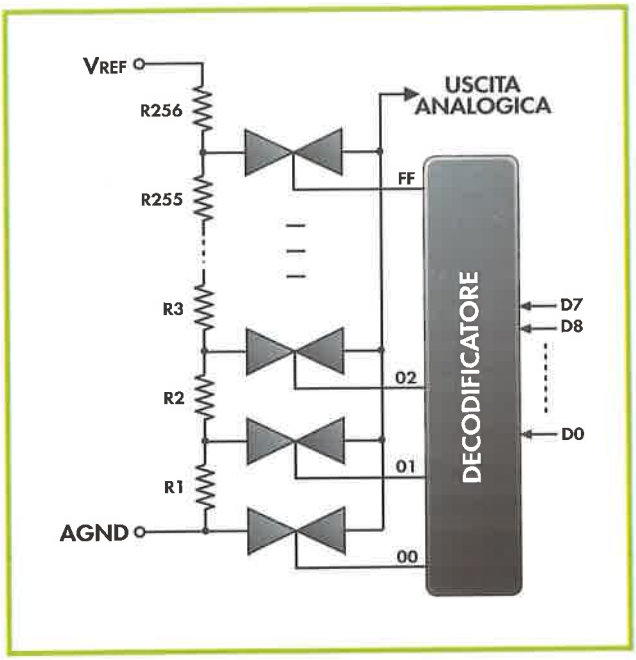
Dopo il byte che contiene l'indirizzo di 7 bit e quello di Lettura/Scrittura (R/W#), il master invia un secondo byte di controllo, che si carica nel registro di controllo del convertitore. I 4 bit più significativi di questo byte servono per attivare l'uscita analogica del CD/A e programmare gli ingressi del CA/D. I 4 bit meno significativi selezionano un canale di ingresso, e il numero del canale si autoincrementerà alla fine della conversione.

FUNZIONAMENTO DEL CONVERTITORE D/A

In caso di utilizzo del convertitore D/A, il terzo byte che invia il master è il valore digitale che si vuole convertire



Il master invia al PCF8591 un secondo byte di controllo, i cui bit governano le parti fondamentali dell'operazione.



Il codice binario del terzo byte che invia il master seleziona una delle 256 resistenze, ottenendo il segnale analogico equivalente.

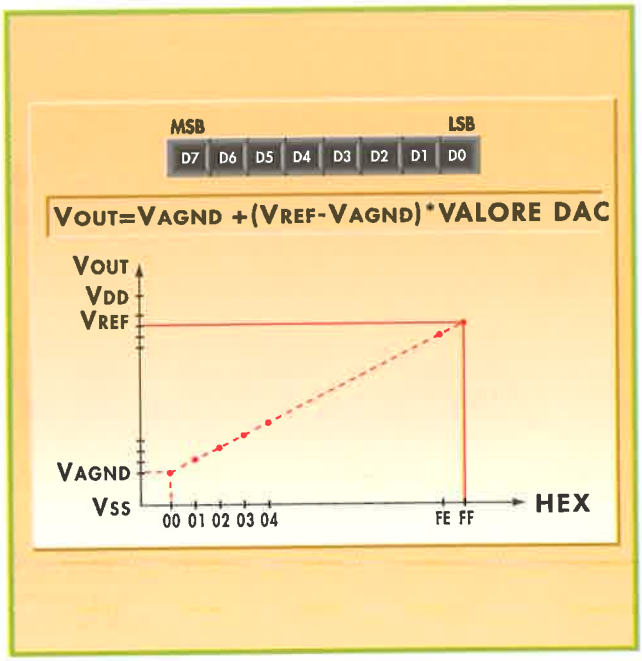
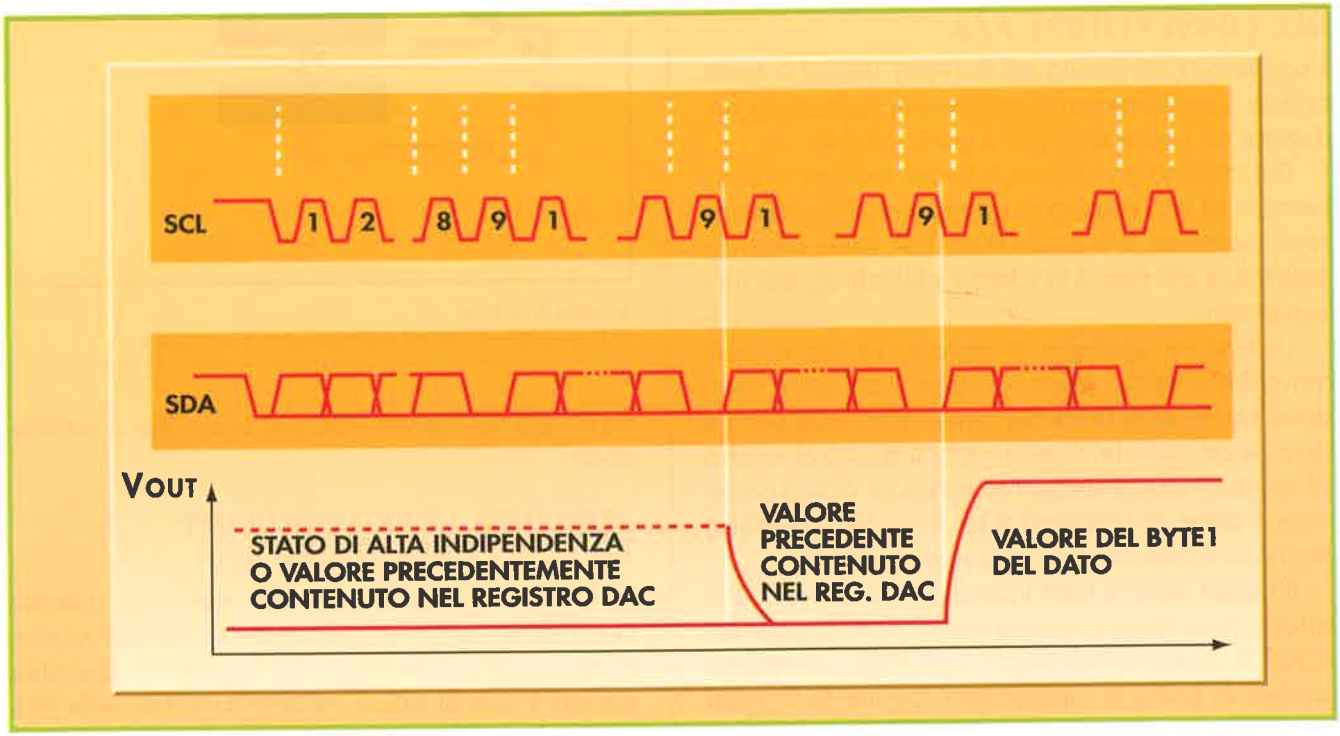


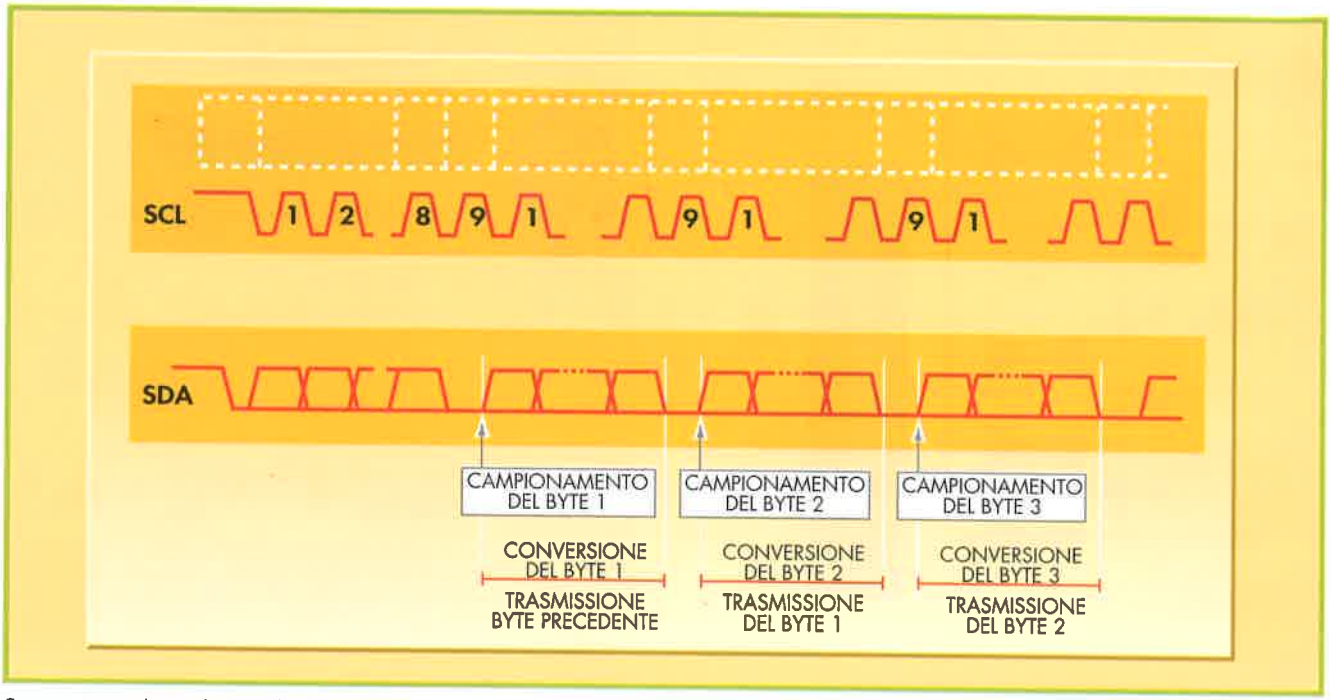
Grafico che rappresenta l'ottenimento del segnale analogico di uscita, in funzione del codice binario o esadecimale.

in tensione analogica. Il convertitore D/A è composto da un divisore di tensione formato da 256 resistenze, a cui si applica la tensione di riferimento. Un decodificatore seleziona uno di questi elementi a seconda del codice binario ricevuto, ottenendo il segnale analogico equivalente.

Il segnale analogico può essere amplificato prima di essere mandato al piedino Aout, dove verrà mantent-



Sequenza della conversione D/A.



Sequenza corrispondente ad una conversione A/D.

stabile sino alla prossima conversione. Il convertitore D/A è utilizzato anche nella conversione che realizza il convertitore A/D.

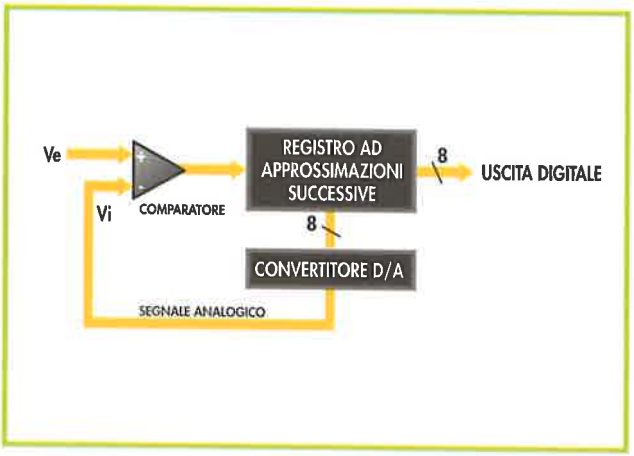
FUNZIONAMENTO DEL CONVERTITORE A/D

Il convertitore contenuto nel PCF8591 utilizza la tecnica delle approssimazioni successive per realizzare la traduzione del valore analogico a quello digitale.

Durante uno dei suoi cicli di lavoro, impiega un comparatore ad alto guadagno, un registro ad approssimazioni successive, e il convertitore D/A come mostrato nella figura che riporta lo schema a blocchi del suo funzionamento.

La tensione analogica di ingresso Ve che si vuole convertire a digitale, viene comparata con quella fornita dal convertitore D/A e dal comparatore esce 0 o 1 a seconda del risultato. Questa uscita si applica al registro ad approssimazioni successive, e in ogni ciclo si determina il valore di uno degli 8 bit di uscita. Al termine degli 8 cicli si ottiene il segnale digitale.

Il master inizia la CAD inviando l'indirizzo e il bit di lettura, dopo riceve il riconoscimento ACK da parte del PCF8591. Quest'ultimo invia il byte della conversione precedente prima di campionare nuovamente il canale selezionato, per realizzare una nuova conversione. Al termine di questa si ottiene il codice a 8 bit che viene



Schema a blocchi del convertitore A/D ad approssimazioni successive.

depositato sul registro ADC per la successiva elaborazione.

SEQUENZA CORRISPONDENTE AD UNA CONVERSIONE A/D

Se è stato attivato il bit 2 del registro di controllo (autoincremento), si seleziona il canale successivo automaticamente. Il primo byte che trasmette il dispositivo durante il ciclo di lettura contiene il risultato della conversione precedente. Se c'è stato un Reset per connessione dell'alimentazione, il byte letto è 80 Hex.

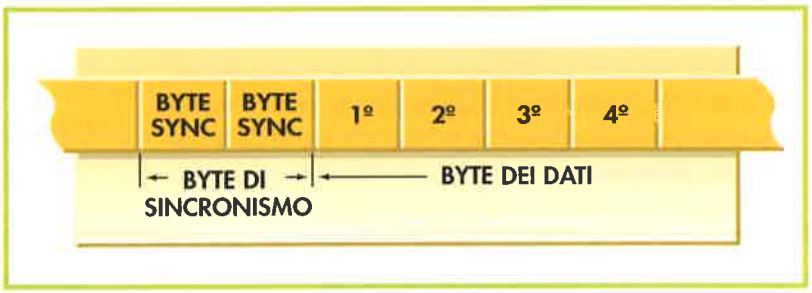
RS-232-C (I)

L'utilizzo diffuso della comunicazione seriale iniziò con l'adattamento di numerose periferiche tipiche dei PC, dall'umile mouse sino al più complesso modem. A fronte di un così grande quantitativo di dispositivi candidati ad utilizzare questo tipo di trasferimento che accomunava i vantaggi della semplicità, economia e sicurezza, furono creati diversi organismi che cercarono di normalizzare i protocolli per poter omologare i prodotti che li utilizzavano. Fra questi ricordiamo negli USA l'EIA, che propose l'interfaccia RS-232-C, considerata uno standard industriale per il suo impiego nelle porte seriali dei PC. In Europa il CCTT propose l'interfaccia V-24 che è identica alla RS-232-C.

Esistono due tipi di comunicazione seriale: sincrona e asincrona, e quest'ultima è la più utilizzata.

ASINCRONA: con questo sistema i bit da trasferire in serie sono divisi in piccoli pacchetti o gruppi, che si inviano preceduti da un segnale di inizio e seguiti da uno di fine. Ogni gruppo di bit, in generale 8, è una parola. Nel caso più semplice gli 8 bit da trasferire sono preceduti da un bit di inizio (Start) e al termine degli 8 bit si pone un bit di fine (Stop), che segnala appunto il termine del trasferimento, così come riportato nel grafico.

SINCRONA: in questo tipo di trasferimento seriale l'emettitore invia i byte dei dati in sequenza, uno dietro l'altro sino al termine. I byte dei dati sono preceduti da uno o due byte di sincronismo. Il ricevitore deve autosincronizzarsi con i bit che riceve; nella figura riportiamo il formato generale della comunicazione seriale sincrona.



Nella comunicazione seriale sincrona i byte dei dati si trasferiscono uno di seguito all'altro, preceduti da alcuni bit di sincronizzazione.

STRUTTURA GENERALE DELLA TRASMISSIONE ASINCRONA

Come abbiamo già spiegato, in questa comunicazione ogni gruppo di bit dei dati deve essere preceduto e seguito da segnali di sincronizzazione. Come minimo davanti abbiamo un bit di Start e dietro uno di Stop.

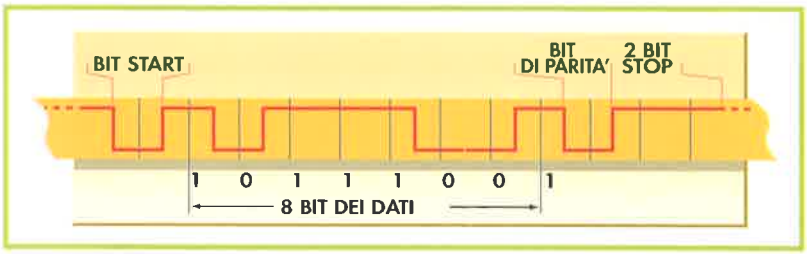
Un parametro fondamentale di tutta la comunicazione è la velocità, che determina il periodo di tempo in cui ogni bit permane sulla linea di trasmissione. L'unità di misura è il "baud", che generalmente significa un bit al secondo. Nel protocollo RS-232-C, la frequenza in baud deve avere un valore normalizzato: 330, 600, 1200, 2400, 4800, 9600, 19.200, 38.400, ecc.

Un'altra caratteristica importante di un protocollo è il numero di bit che contiene ogni pacchetto o parola, che deve essere compreso fra 5 e 9. Anche se non è obbligatorio, è molto frequente includere in ogni trasferimento un bit di parità, dopo i bit dei dati e prima del bit di Stop. Questo bit si genera e si rileva per indicare e conoscere la parità o disparità del numero di bit a 1 che contiene la parola inviata o ricevuta, rispettivamente. È

Nella comunicazione seriale asincrona di base, ogni pacchetto di bit dei dati deve essere preceduto da un bit di Inizio (Start) e seguito da un bit di Fine (Stop).

COMUNICAZIONI

Comunicazione seriale sincrona e asincrona



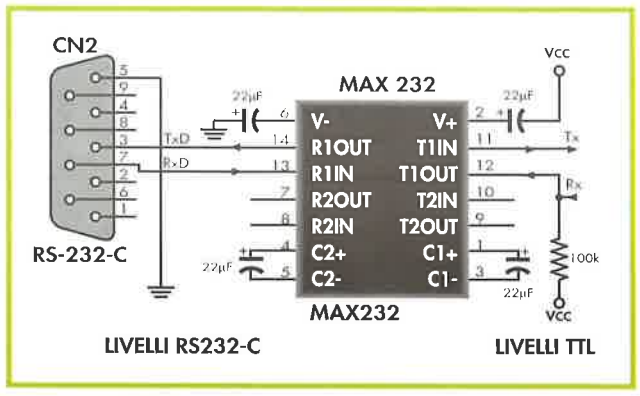
Il cronogramma rappresenta il formato RS232-C del trasferimento di una parola di 8 bit, che deve essere preceduta dal bit di Start e seguita dal bit di parità e da 2 bit di Stop.

uno strumento molto efficace e molto semplice per sapere se il trasferimento è stato realizzato normalmente o si sono verificati errori, che la maggior parte delle volte sono provocati da un solo bit che cambia valore, cambiando la parità. Se nella trasmissione si verifica un errore, cioè un bit cambia valore, cambierà la parità, e questo fatto sarà rilevato dal bit di parità.

Alla fine della trasmissione di una parola si inviano i bit di Stop, che possono essere 1, 1.5 o 2 e la cui funzione principale è determinare il tempo minimo fra due trasferimenti di parole consecutive.

Nella figura si mostra il diagramma temporale della trasmissione di una parola di 8 bit, secondo il protocollo RS-232-C, il cui codice è 10111001. Il bit di parità vale 0 e indica che la parità del dato è dispari, ovvero che ci sono cinque bit a 1. Infine abbiamo 2 bit di Stop ad indicare che la parola successiva da trasferire deve attendere un minimo di due periodi di clock dopo la fine di quella descritta.

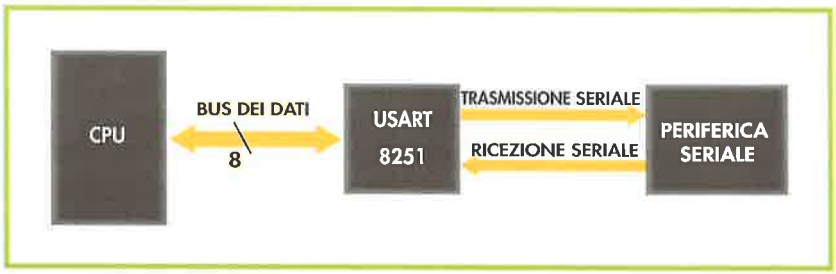
Nella normativa di questo protocollo, sono stabiliti i range della tensione DCV fra cui devono essere compresi i due stati logici. Il livello alto fra +5 e +15 VDC, e il livello basso fra -5 e -15 VDC per i segnali di uscita. Se i segnali sono di ingresso i range variano fra + 3 e + 15 VDC per il livello alto e fra -3 e -15 VDC per il livello basso. Dato che questi livelli non sono compatibili con quelli TTL dei microcontroller e dei microprocessori del PC, sono necessari circuiti integrati specializzati, come quello mostrato in figura, che ha la sigla MAX 232, che hanno il compito di adattare i livelli in entrambi i versi.



Il circuito integrato MAX 232 trasforma i livelli RS232-C in livelli TTL e viceversa.

L'USART

Si tratta di un circuito integrato progettato e costruito da Intel, che, come dice il suo nome, è un Trasmettitore-Ricevitore Sincrono–Asincrono Universale. La sua funzione è quella di convertire l'informazione parallela, che arriva dal microcontroller o dal PC, in seriale per le periferiche utilizzate, e viceversa. Comunica con il microcontroller o il PC tramite un bus da 8 linee, mentre per la comunicazione con le periferiche utilizza la linea o le linee necessarie a seconda del protocollo impiegato.



La USART trasforma l'informazione seriale in parallela e viceversa.

L'architettura interna dell'USART, mostrata nella figura è composta da 4 blocchi:

- *Sezione trasmissione:*

Invia, tramite la linea TxD, dati in serie alle periferiche. Quando la sezione è vuota lo segnala la linea TxE. La frequenza di lavoro di questa sezione è ricevuta tramite la linea TxC#.

- *Sezione ricezione:*

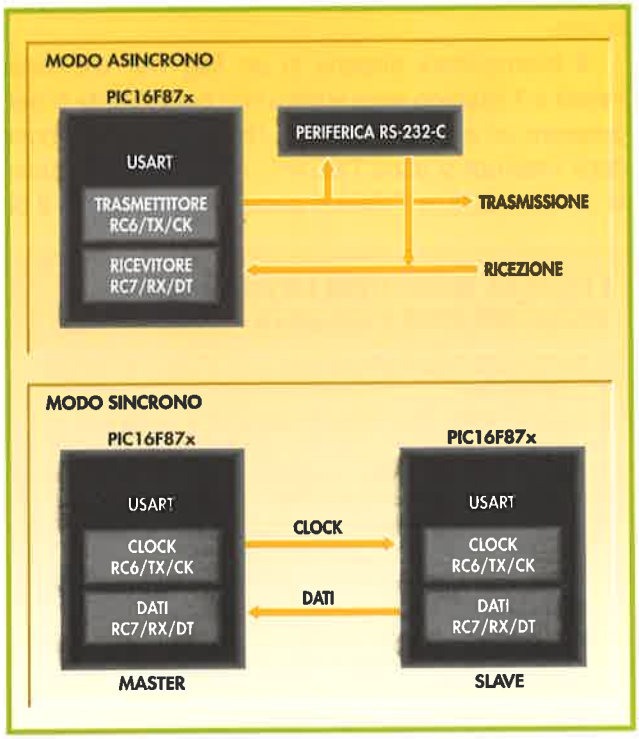
Riceve i dati seriali tramite la linea RxD, e li trasforma in paralleli, per inviarli alla CPU mediante il bus dei dati da 8 bit. La linea RxRDY avvisa quando ha ricevuto un dato, e tramite RxC# si applica la frequenza di lavoro, che deve essere uguale a quella di trasmissione.

RS-232-C (II)

Alcuni modelli di microcontroller contengono implementati sul silicio diversi tipi di porte di comunicazione, tra cui un USART, per permettere la comunicazione con il protocollo RS-232-C, fra questi il PIC16C7X e il PIC16F87X.

In riferimento alla famiglia PIC16F87X, abbiamo già visto che dispone di un modulo MSSP che supporta due formati di comunicazione sincrona, con il segnale di clock, in modo SPI, e in modo bus I2C. Inoltre possiedono un modulo SCI (Serial Comunication Interface) che contiene un USART capace di lavorare in tre modi diversi:

1°. Asincrono (Full duplex bidirezionale).
2°. Sincrono Master (Half duplex unidirezionale).
3°. Sincrono Slave (Half duplex unidirezionale).



Comportamento dei PIC16F87X in modo seriale asincrono e sincrono.

Nella figura è riportato lo schema del comportamento dell'USART dei PIC in modo asincrono e sincrono.

Nel primo il trasferimento si realizza sulle linee Tx e Rx, dalle quali escono ed entrano i bit seriali, alla velocità della frequenza di lavoro. Nel modo sincrono la linea DT è bidirezionale, e trasla i dati nei due sensi, alla frequenza degli impulsi di clock che arrivano dalla linea CK del Master. Queste linee sono implementate sui piedini multifunzione RC7/RX/DT e RC6/TX7CK.

Quando si lavora con il protocollo RS-232-C la parola può essere di 8 o 9 bit - abitualmente è 8 bit - i quali sono preceduti dal bit di Start e seguiti dal bit di Stop. I PIC16F87X non supportano il bit di parità, quindi se la periferica da controllare lo richiede, bisognerà crearlo via software, scrivendolo sul nono bit del dato.

ARCHITETTURA INTERNA

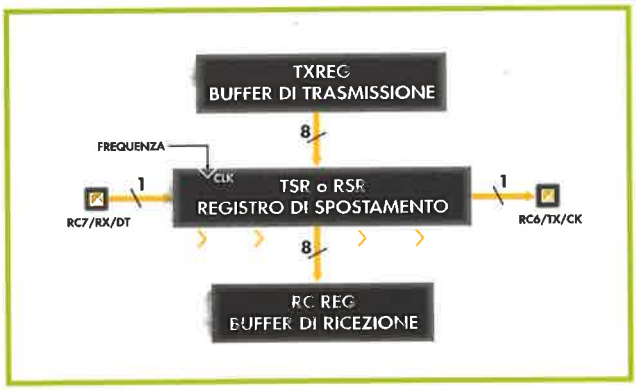
L'USART dei PIC16F87X è composta da 4 blocchi:

1°. Circuito di campionamento: ha il compito di campionare i bit che arrivano dalla linea RC7/RX/DT, e realizza tre campionamenti per ogni bit, decidendo il valore a maggioranza.

2°. Generatore di baud: ha il compito di generare gli impulsi della frequenza di lavoro.

3°. Trasmettitore asincrono.

4°. Ricevitore asincrono.



Architettura interna dell'USART dei PIC16F87X.

Nella figura è riportata la struttura interna dell'USART, dove notiamo due registri: quello di spostamento, TSR e il buffer per la memorizzazione del dato parallelo, TXREG in trasmissione e RCREG in ricezione.

IL GENERATORE DI BAUD

I trasferimenti nel protocollo RS-232-C si effettuano ad una frequenza normalizzata in baud (330, 600, 1.200, 2.400, 4.800, 9.600, 19.200, 38.400 ecc.). Questa frequenza è prodotta dal generatore di baud BRG, il cui valore è controllato dalle impostazioni del registro SPBRG.

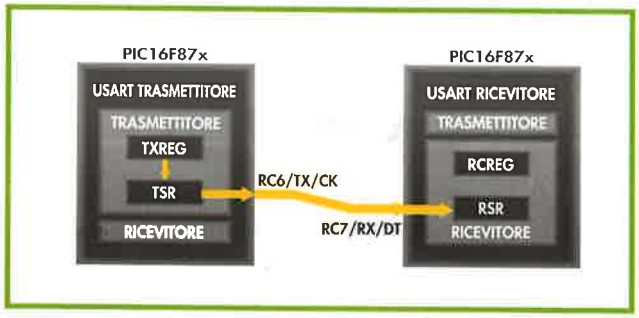
La frequenza del generatore di baud dipende dal valore X caricato nel registro PSBRG, e dal valore del bit BRGH (TXSTA). Se BRGH = 0 si lavora con bassa velocità, e la costante K della formula vale 64.

Quando BRGH = 1 il valore K = 16.

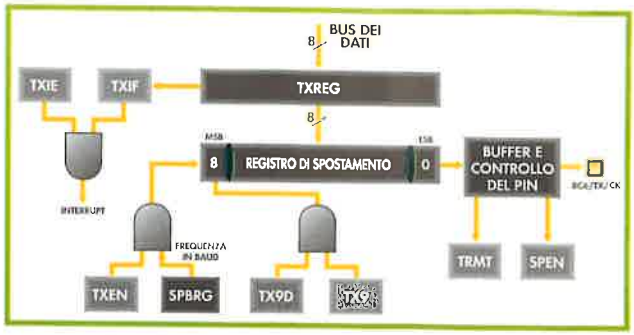
Frequenza in baud = Fosc / K * (X + 1)
X: Valore caricato nel registro SPBRG
K = 64 se BRGH = 0
K = 16 se BRGH = 1

L'USART COME TRASMETTITORE ASINCRONO

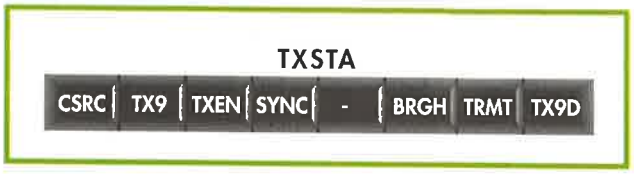
Nella figura è riportato il collegamento fra due PIC16F87X tramite l'USART di cui dispongono entrambi, sotto il protocollo RS-232-C. Uno dei due funziona come trasmettitore, e l'altro come ricevitore. Il dato da inviare tramite l'USART di trasmissione è depositato sul registro TXREG, e poi è passato al registro di spostamento TSR, che manda in uscita i bit in modo sequenziale. Inoltre prima di trasmettere i bit dei dati, genera il bit di Start e alla fine introduce un bit di Stop. L'USART ricevitore riceve questi bit, elimina il bit di Start e quello di Stop, e quando il registro RSR dispone di tutti i bit della parola la trasla automaticamente al registro buffer RCREG.



Collegamento di due PIC16F87X tramite il loro modulo USART in funzionamento asincrono.



Schema a blocchi del trasmettitore asincrono dell'USART.



Struttura interna del registro di stato del trasmettitore dell'USART del PIC16F87X.

Lo schema a blocchi del trasmettitore asincrono della figura, mostra gli elementi fondamentali e il compito di alcuni bit di controllo del registro di stato del trasmettitore.

Il registro di stato del trasmettitore si chiama TXSTA e possiamo vedere la sua struttura interna nella figura in alto.

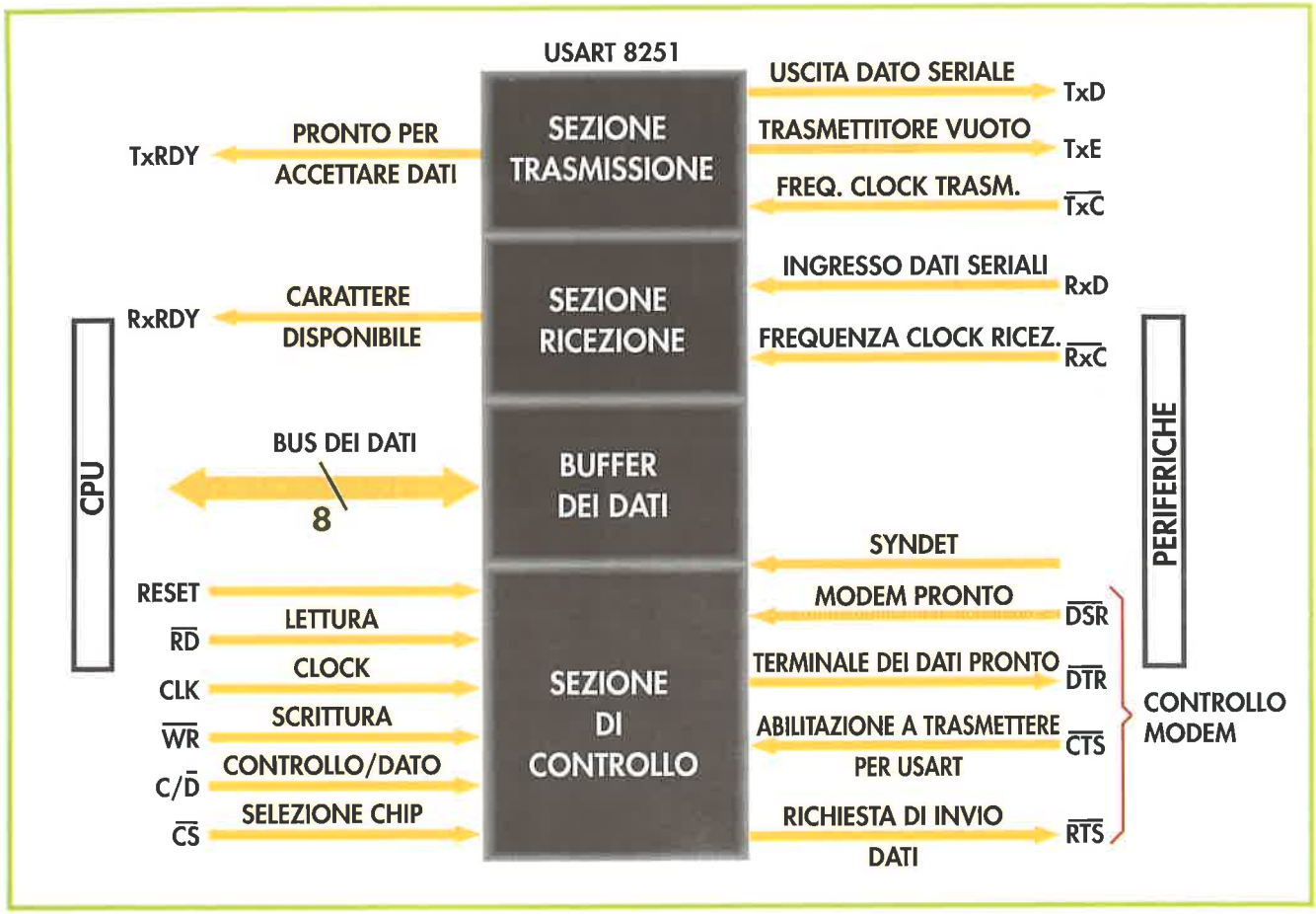
Il trasmettitore dispone di un flag TXIF che viene messo a 1 quando resta vuoto, e in quell'istante si può generare un interrupt nel caso che il bit di abilitazione degli interrupt si pone TXIE = 1. Al momento di scrivere un altro dato in TXREG si pone TXIF = 0. Inoltre il bit

I PASSI DA SEGUIRE PER LA TRASMISSIONE ASINCRONA SONO I SEGUENTI:

1. Configurare la linea RC6/TX/CK come uscita e la linea RC7/RX/DT come ingresso.
2. Impostare SPEN = 1 per attivare l'USART e SYNC = 0 per farlo funzionare in modo asincrono.
   Entrambi i bit si trovano nel registro di stato TXSTA.
3. Impostare TXIE = 1 se si desidera lavorare con gli interrupt.
4. Quando il dato è di 9 bit si imposta il bit TX9D = 1 e il valore del nono bit si deposita in TX9D del registro TXSTA.
5. Si carica il registro SPBRG con il valore X adatto per lavorare alla frequenza selezionata.
   Controllare il bit BRGH per lavorare con la frequenza alta o bassa, e definire la costante K della formula della frequenza in baud.
6. Impostare il bit TXEN = 1 per attivare la trasmissione.
   Appena TXREG è vuoto il flag TXIF passa a 1.
7. Caricare il dato da trasferire in TXREG, TXIF passa a valore 0, e inizia la trasmissione.

COMUNICAZIONI

Comunicazione seriale sincrona e asincrona



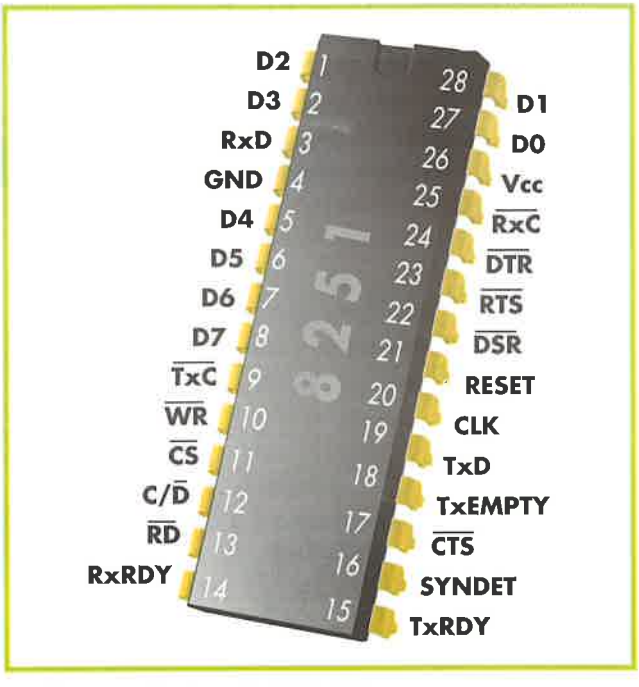
Architettura base dell'USART con le sue principali linee di ingresso e uscita.

• *Buffer dei dati:*
Quando la CPU esegue un'istruzione di uscita dei dati (OUT), carica un registro fondamentale di questa sezione con 8 bit, per procedere successivamente ad inviarli uno a uno alla periferica. Se si esegue un'istruzione di ingresso (IN), la CPU legge il registro del buffer dove sono caricati gli 8 bit che sono arrivati in modo seriale dalla periferica.

• *Sezione di controllo:*
Riceve e invia i segnali di controllo necessari per supportare la comunicazione e per adattarsi ad un modem, che è un dispositivo capace di trasformare dati digitali seriali, in dati analogici, per poi trasmetterli su una linea telefonica.

L'USART ha un contenitore a 28 pin per supportare tutti i segnali necessari per adattarsi da un lato una CPU a 8 bit, e dall'altro alle periferiche seriali e al modem.

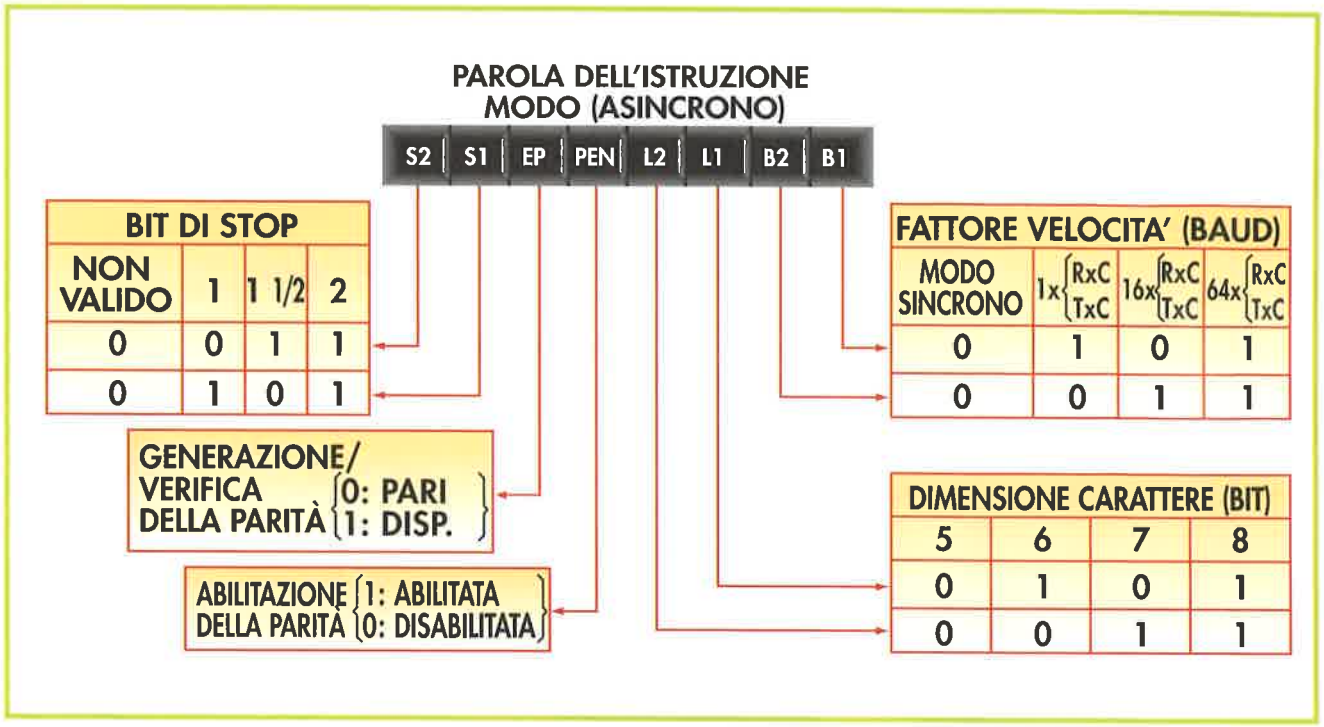
La piedinatura del componente è riportata nella figura a lato.



Piedinatura a 28 pin dell'USART 8251.

COMUNICAZIONI

Comunicazione seriale sincrona e asincrona



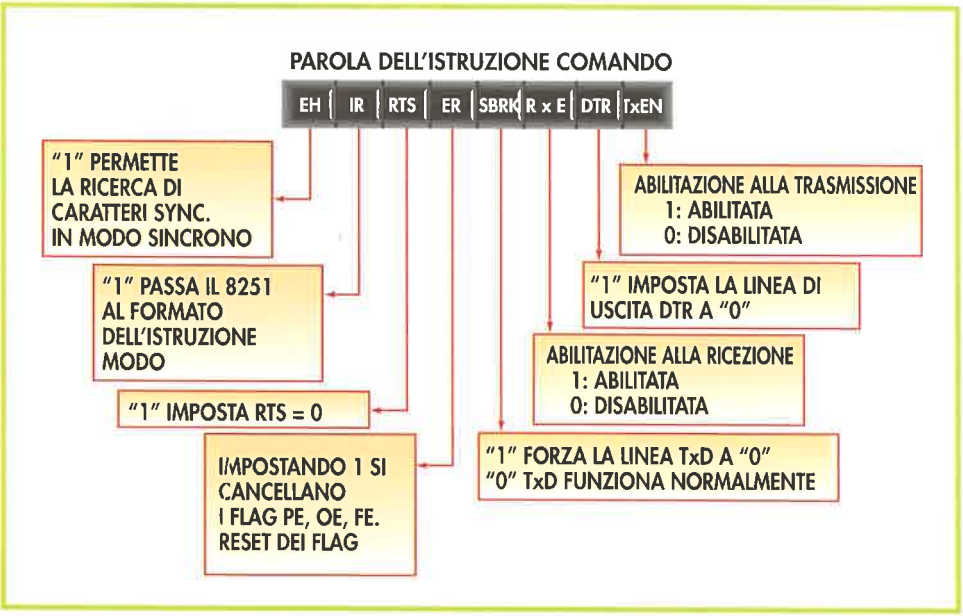
Compiti dei bit della parola "istruzione modo".

PROGRAMMAZIONE DELL'USART

Questo circuito è programmabile, quindi caricando correttamente i bit dei registri di controllo, si stabiliscono le caratteristiche del suo funzionamento, e di conseguenza quelle della comunicazione. Tra le caratteristiche programmabili ricordiamo: selezione del trasferimento sincrono e asincrono, velocità di trasferimento, numero di bit per parola, esistenza o meno del bit di parità, numero di bit di Stop, ecc.

L'USART dispone di un solo registro di controllo per la programmazione, però è multifunzione, in modo che quando si inizializza la configurazione dopo un reset, deve essere scritto due volte consecutivamente. Prima si scrive un byte che si chiama "istruzione modo", e dopo un altro che si chiama "istruzione comando". Con il primo si definiscono le caratteristiche principali del funzionamento dell'USART e con il secondo le funzioni ausiliarie, come illustrato nella figura, che descrive il compito di ognuno dei bit e di queste due parole.

La cosa interessante di questo circuito integrato chiamato USART è che molti modelli di microprocessori lo contengono integrato sul silicio, questo li rende adatti a una connessione diretta con tutti i tipi di periferiche che comunicano tramite il protocollo RS-232-C.



Compiti dei bit della parola "istruzione comando".

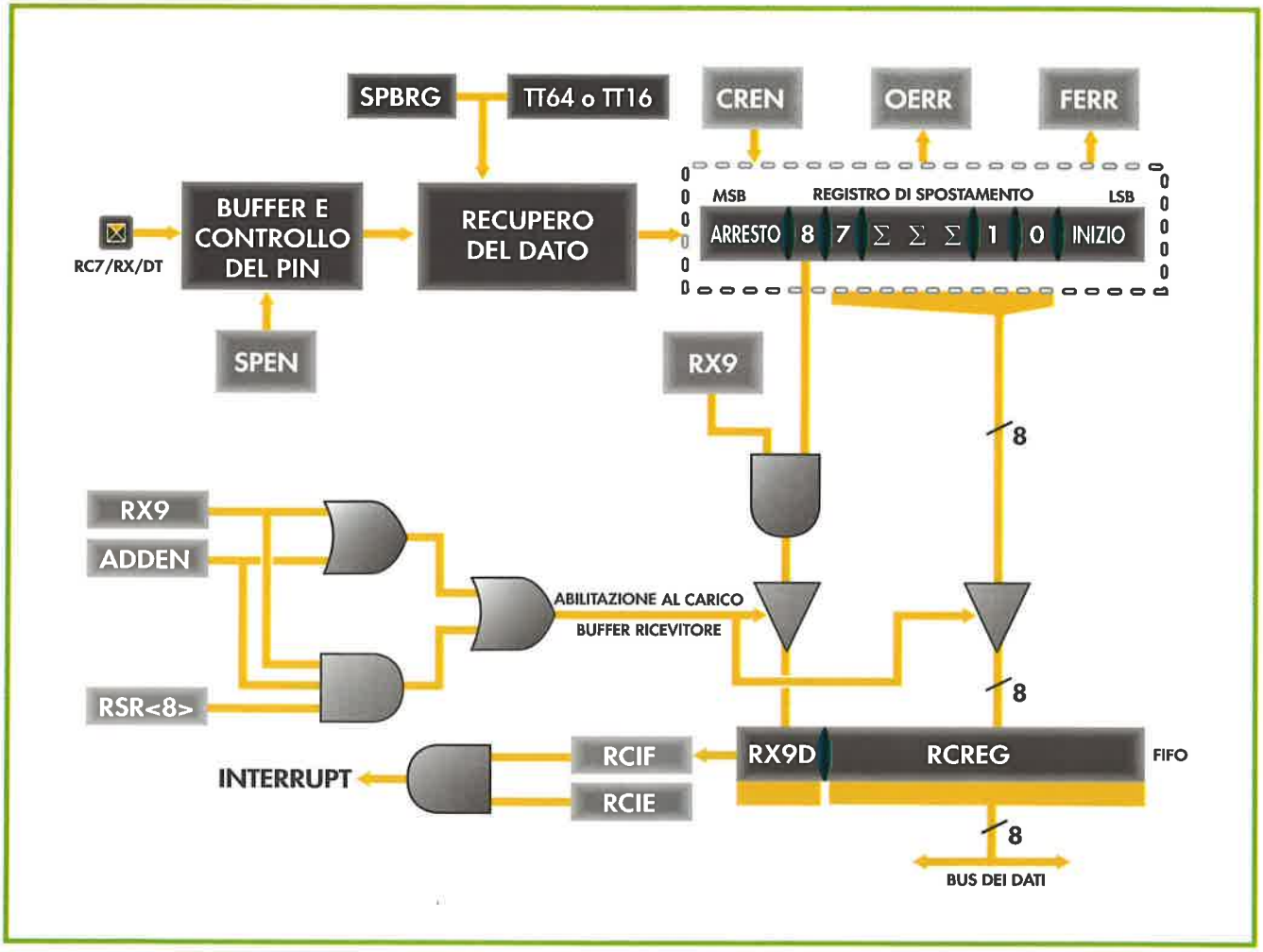
TRMT del registro TXSTA vale 1 quando il trasmettitore è vuoto. Il bit CSRC definisce nel modo sincrono quando l'USART lavora come Master (1) o come Slave (0).

L'USART COME RICEVITORE ASINCRONO

Il dato ricevuto via seriale dal pin RC7/RX/DT entra nel registro RSR che funziona ad una frequenza 16 volte superiore a quella di lavoro. Quando si riempie RSR e si riceve il bit di Stop si sposta l'informazione al registro RCREG. Il registro di controllo e di stato del ricevitore si chiama RCSTA e la sua struttura interna è riportata nella figura.

Quando il dato ha 9 bit bisogna scrivere RX9 = 1, e il nono bit si colloca in RX9D del registro RCSTA. Quando il bit ADDEN = 1 si rileva l'indirizzo e si scarica il buffer di ricezione quando si attiva RSR, mentre se

Struttura interna del registro RCSTA per il ricevitore asincrono dell'USART.



Schema a blocchi del ricevitore dell'USART.

COMUNICAZIONI

Un USART all'interno dei PIC

I PASSI DA SEGUIRE PER REALIZZARE UNA RICEZIONE ASINCRONA SONO I SEGUENTI:

1. Si carica il registro SPBRG con il valore X adatto a lavorare alla frequenza desiderata.
   Dobbiamo anche controllare il valore del bit BRGH che determina la costante K.
2. Impostare SPEN = 1 per attivare l'USART e SYNC = 0 per farlo funzionare in modo asincrono.
3. Se vogliamo che si produca un interrupt all'arrivo dell'ultimo bit della parola dobbiamo impostare il bit di abilitazione RCIE = 1.
4. Quando si lavora con parole di 9 bit si imposta RX9 = 1.
5. Per rilevare l'indirizzo si imposta ADDEN = 1.
6. Si abilita l'USART in ricezione con CREN = 1.
7. Terminata la ricezione di una parola RCIF passa a 1 e si produrrà un interrupt se il bit RCIE = 1.
8. Si legge RCSTA e si verifica se ci sono stati errori.
9. Si leggono gli 8 bit di RCREG per determinare se il dispositivo è stato indirizzato.
10. Se ci sono stati errori porre CREN = 0.
11. Se il dispositivo è stato indirizzato dobbiamo porre ADDEN = 0, per permettere la ricezione del dato.

ADDEN = 0 si ricevono tutti i bit e il nono può essere utilizzato come bit di parità. Il bit CREN abilita la ricezione continua dei dati quando vale 1 e la disabilita quando vale 0. Il bit SPEN abilita (1) o disabilita (0) la porta seriale. SPEN non influisce sul funzionamento in modo asincrono.

Quando FERR = 1 indica che è stato prodotto un errore di trama. Se OERR = 1 significa che è stato prodotto un errore di sovrapposizione.

Nello schema a blocchi del ricevitore dell'USART si può vedere il funzionamento dei bit del registro RCSTA. Nella tabella della figura sono raccolti i registri che si utilizzano nell'USART dei PIC16F87X, con le caratteristiche più rilevanti, e la loro struttura interna.

INDIRIZZO	NOME	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	VALORE DOPO LA CANCELLAZIONE	VALORE NEL RESTO DEI RESET
8Ch	PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
0Ch	PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	--	BRGH	TRMT	TX9D	0000 -010	0000 -010
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
1Ah	RCREG	Registro di ricezione dell'USART								0000 0000	0000 0000
19h	TXREG	Registro di trasmissione dell'USART								0000 0000	0000 0000
99h	SPBRG	Registro generatore di baud								0000 0000	0000 0000

Principali caratteristiche e struttura interna dei registri che utilizza l'USART nei PIC16F87X.

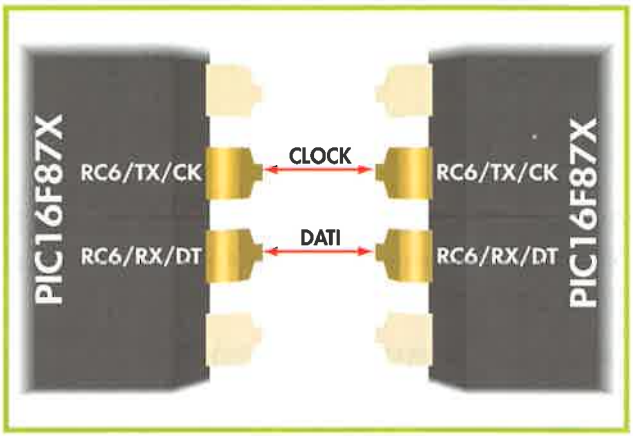
Comunicazione seriale sincrona

Ci sono diversi dispositivi e periferiche che lavorano con il protocollo seriale sincrono semiduplex, tipo memorie EEPROM seriali, convertitori AD e DA, ecc. In questo modo ogni elemento può lavorare come master o slave.

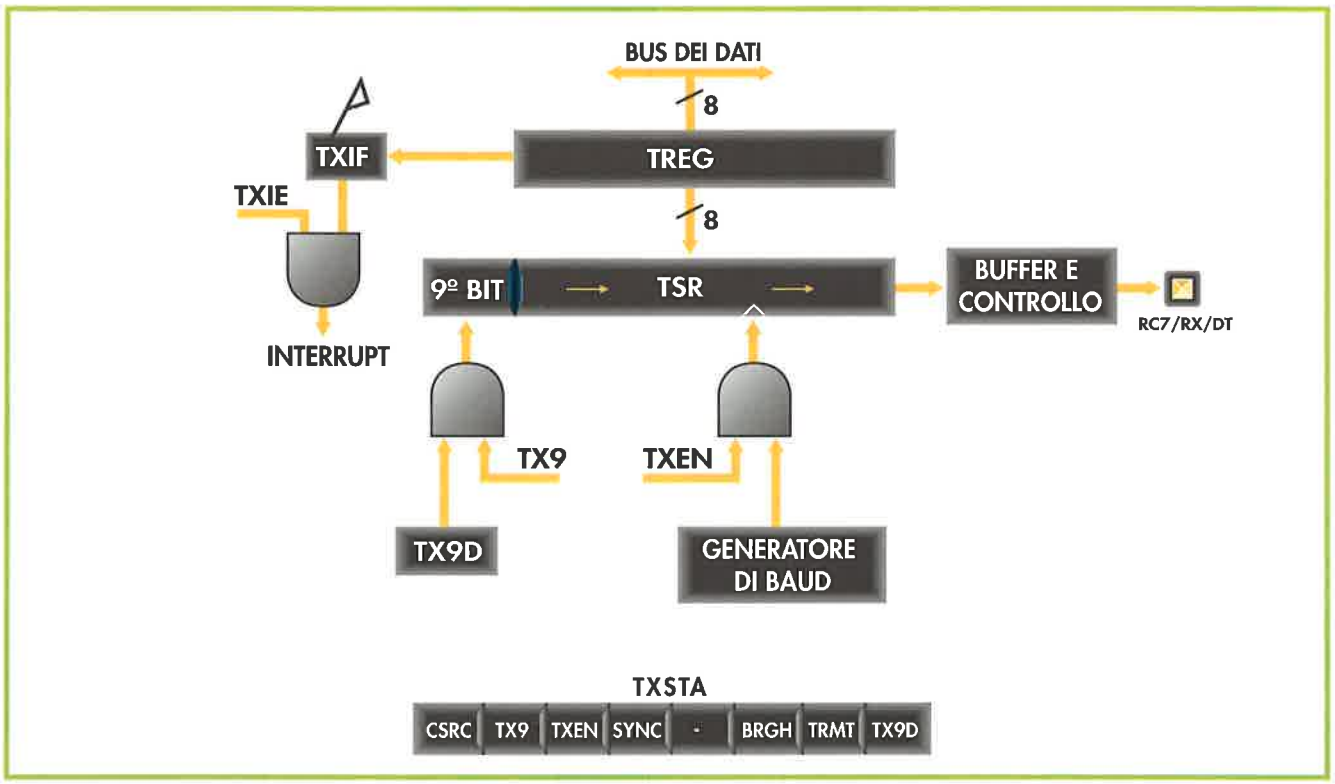
I bit sono trasmessi nei due versi tramite la linea dei dati RC7/RX/DT. Il segnale di clock è sempre generato dal master tramite il pin di uscita RC6/TX/CK, che nel caso dello slave è un piedino di ingresso.

Quando il master trasmette, la ricezione in esso rimane inibita, e viceversa.

Ricordate che nella comunicazione sincrona, il trasferimento dei dati è continuo, e ogni blocco di infor-



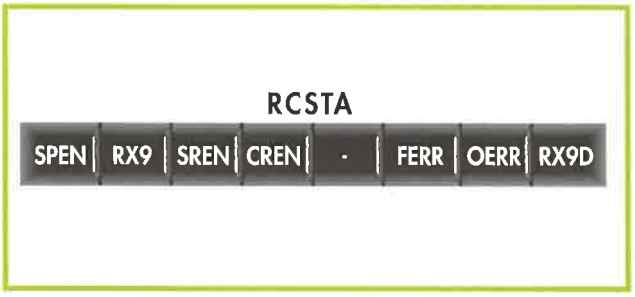
Nel modo sincrono dell'USART, la frequenza di trasferimento dell'informazione è generata dal master.



Schema a blocchi del trasmettitore dell'USART e struttura interna del registro TXSTA.

COMUNICAZIONI

L'USART in modo sincrono



Struttura interna del registro RCSTA incaricato del controllo della ricezione dell'USART.

mazione va preceduto da uno o due byte che di solito sono caratteri ASCII SYNC.

Inoltre nel modo sincrono bisogna programmare l'USART per selezionare i parametri di funzionamento, fra cui la frequenza in baud e il numero di bit della parola, che possono essere 8 o 9 ecc.

MASTER IN MODO SINCRONO

Il master come trasmettitore utilizza, così come in modo asincrono, il registro TXREG come buffer dei dati, e il TSR come registro di spostamento. Il registro TSR riceve il dato dal TXREG, che non si carica sino a che non è uscito l'ultimo bit dal TSR. Al termine della trasmissione e rimanendo vuoto il TXREG, si può generare un interrupt, se il bit di abilitazione TXIE vale 1, in questo modo la CPU salta alla routine di interrupt che torna a caricare il registro TXREG, per continuare immediatamente dopo con la trasmissione del dato in serie.

Nella figura riportata in basso nella pagina precedente, è illustrato lo schema a blocchi del master trasmettitore, insieme alla struttura interna del registro TXSTA, i cui bit controllano e monitorizzano questa operazione.

Per abilitare la trasmissione si pone TXEN a 1. Il bit TX9 è a 1 quando le parole sono di 9 bit, depositando il nono bit in TX9D. Quando SYNC vale 1 si sceglie il modo sincrono. Il bit CSRC determina in modo sincrono quando il funzionamento è in modo master (1) o in modo slave (0).

Nel primo caso la frequenza di lavoro la produce il Generatore di Baud, in funzione del valore caricato nel registro SPBRG. Quando il bit TRMT vale 1 il registro TSR è vuoto. Infine il bit BRGH seleziona il valore della costante K che interviene nella formula per il calcolo della frequenza in baud, visto durante lo studio del modo asincrono.

I passi da seguire per realizzare la trasmissione dell'USART come master in modo sincrono sono i seguenti:

1°. Caricare il registro SPBRG con il valore appropriato di X, per lavorare alla frequenza scelta.

2°. Porre il bit SYNC a 1 per scegliere il modo lavoro sincrono, CSRC a 1 per far lavorare l'USART come master, e il bit SPEN a 1 per abilitare la porta seriale.

3°. Collocare TXIE a 1 se si desidera che si generi un interrupt al termine della trasmissione.

4°. TX9 è 1 quando la parola è di 9 bit.

5°. Porre TXEN a 1 per abilitare la trasmissione.

6°. Caricare il nono bit della parola in TX9D.

7°. Iniziare la trasmissione caricando la prima parola nel registro TXREG.

Per controllare la ricezione nel modo master si utilizzano i bit del registro RCSTA.

Quando si lavora con parole di 9 bit dobbiamo porre RX9 a 1 e il nono bit che si riceve si carica sul bit RX9D. Con SPEN a 1 si abilita la porta seriale. Con SREN a 1 si abilita il master alla ricezione e con CREN a 1 si permette la ricezione continua. Quando FERR è a 1 si rileva un errore di trama e con OERR che vale 1 l'errore rilevato è di sovrascrittura.

Nella figura in alto a sinistra si fornisce un quadro con la struttura interna, indirizzi e valori che prendono i registri dell'USART in modo sincrono dopo un Reset.

Per abilitare la ricezione nel master si pone SREN a 1. I bit entrano in modo seriale tramite il pin RC7/RX/DT con il fronte di salita dell'impulso di clock. Se solo SREN vale 1 si riceve una parola solamente, ma se è a 1 anche CREN si ricevono parole in modo consecutivo.

Ogni volta che si riempie RSR, il valore si trasferisce al RCREG.

COMUNICAZIONI

L'USART in modo sincrono

I passi da seguire nella ricezione sincrona dell'USART sono i seguenti:

1°. Si carica il valore adeguato di X nel registro SPBRG, per selezionare la frequenza di lavoro.
2°. Porre il bit SYNC a 1 per il modo sincrono, SPEN a 1 per abilitare la porta seriale, CSRC a 1 per far lavorare l'USART come master.
3°. Verificare che: CREN = SREN = 0.
4°. Quando si vuole produrre un interrupt al termine della ricezione si pone RCIE a 1, che è il bit di abilitazione.
5°. Lavorando con parole da 9 bit, RX9 deve essere a 1.
6°. Se desideriamo realizzare solo la ricezione di una parola imposteremo SREN = 1. Quando si vuole realizzare una ricezione continua di parole imposteremo anche CREN a 1.
7°. Al termine della ricezione imposteremo a 1 il flag RCIF, producendo un interrupt se avremo posto RCIE a 1.
8°. Si legge il registro RCSTA e il RX9D se si lavora con 9 bit.
9°. Verificare se si è prodotto qualche errore nella ricezione, testando il valore dei flag FERR e OERR.
10°. Se si è verificato qualche errore cancellarlo ponendo CREN a 0.
11°. Lettura del registro RCREG.

SLAVE IN MODO SINCRONO

Le due differenze più importanti dello slave e del master quando lavorano in comunicazione sincrona sono le seguenti:

1°. Lo slave riceve tramite il piedino RC6/TX/CK la frequenza degli impulsi del clock che arrivano dal master.
2°. È possibile che lo slave stia lavorando mentre il PIC si trova in modo Riposo, situazione che si origina dopo l'esecuzione di una istruzione SLEEP.

Per lavorare nel modo Riposo si scrivono due parole nel registro TXREG, e dopo si esegue l'istruzione SLEEP. Successivamente si trasferisce al registro TSR la prima parola e si trasmette via seriale. Quando termina la trasmissione della prima parola TXIF non viene messo a 1 e la seconda parola passa da TXREG a TSR.

Dopodiché si porrà TXIF a 1, e se TXIE era stato impostato a 1 si produrrà un interrupt che sveglierà il microcontroller dal modo Riposo, mandandolo alla routine di interrupt che inizia all'indirizzo 0004 H.

I passi da seguire per la trasmissione serie sincrona dello slave sono i seguenti:

INDIRIZZO	NOME	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	VALORE DOPO LA CANCELLAZIONE	VALORE DOPO GLI ALTRI RESET
8Ch	PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
0Ch	PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	--	BRGH	TRMT	TX9D	0000 -010	0000 -010
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
1Ah	RCREG	Registro di ricezione dell'USART								0000 0000	0000 0000
99h	SPBRG	Registro generatore di baud								0000 0000	0000 0000
19h	TXREG	Registro di trasmissione dell'USART								0000 0000	0000 0000

Struttura interna, indirizzi e principali caratteristiche dei registri dell'USART che intervengono nel modo sincrono.

COMUNICAZIONI

L'USART in modo sincrono

1°. Abilitare la porta seriale impostando SPEN a 1 e il modo sincrono con SYNC a 1. Per fare in modo che lavori come slave si pone CSRC a 0.

2°. Porre CREN = SREN = 0

3°. Se desideriamo generare un interrupt alla fine della trasmissione impostiamo TXIE a 1.

4°. Se vogliamo trasmettere parole di 9 bit impostiamo TX9 a 1.

5°. La trasmissione si attiva con TXEN a 1.

6°. Se si opera con 9 bit caricheremo il nono in TX9D.

7°. Iniziare la trasmissione caricando i dati in TXREG.

Per fare in modo che lo slave funzioni in ricezione imposteremo CREN a 1. Se il PIC è in modo Riposo, quando si riceve una parola sul registro di spostamento RSR, al momento di spostarla su RCREG, se erano stati abilitati gli interrupt con RCIE = 1, si genererà l'interrupt e il microcontroller uscirà dal modo Riposo.

I passi per fare in modo che uno slave compia un'operazione di ricezione sono i seguenti:

1°. Abilitare la porta seriale ponendo SPEN a 1. Porre in modo sincrono con SYNC a 1 e in modo slave con CSRC = 1.

2°. Se si desidera generare un interrupt al termine della ricezione RCIE deve essere a 1.

3°. Con parole di 9 bit RX9 varrà 1.

4°. La ricezione si abilita quando RCEN vale 1.

5°. Al termine della ricezione RCIF è uguale a 1 e si genera un interrupt se CRIE è a 1.

6°. Si legge il registro RCSTA per conoscere il possibile nono bit, e se si sono verificati errori nella ricezione.

7°. Si legge il registro RCREG.

8°. Se si è verificato qualche errore si cancella il flag ponendo CREN = 0.

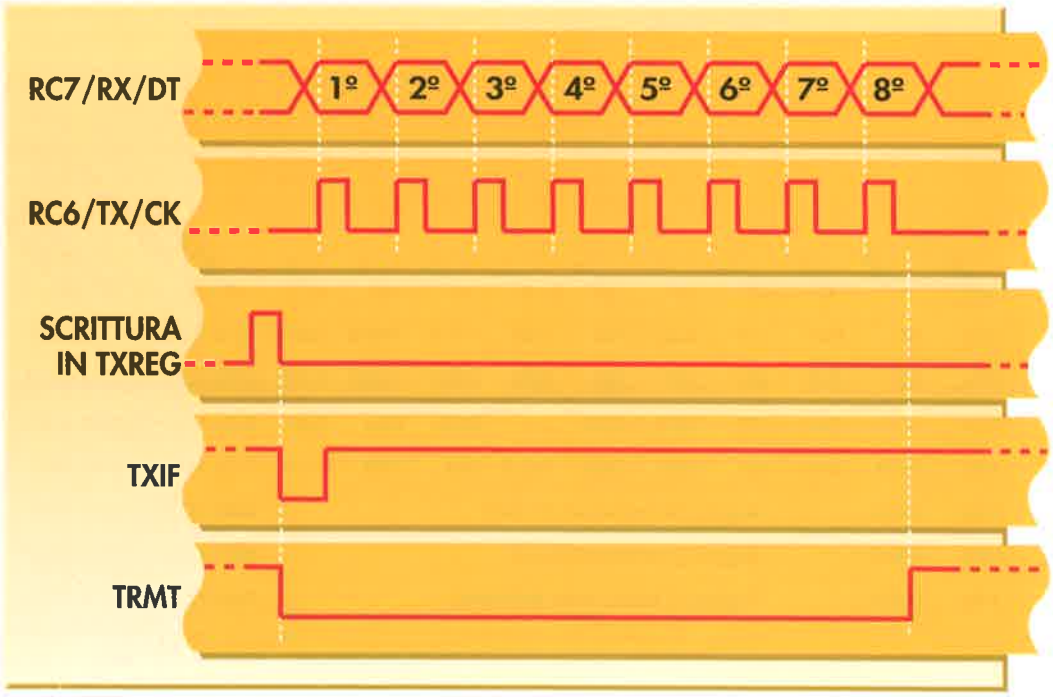
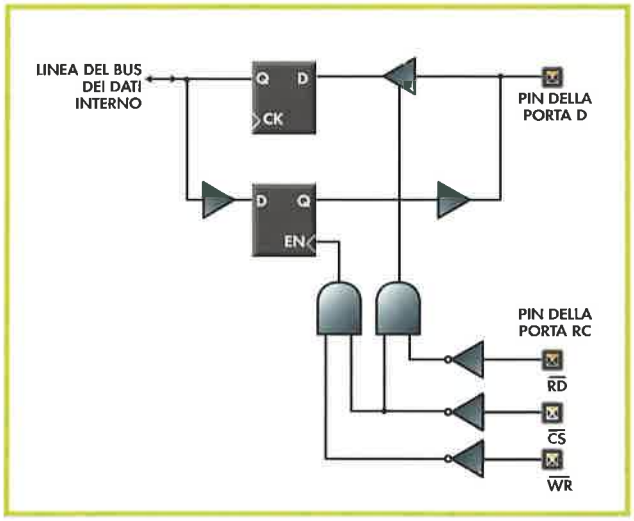


Diagramma dei tempi di una trasmissione sincrona di parole di 8 bit.

Comunicazione parallela

I PRO E I CONTRO DELLA COMUNICAZIONE PARALLELA

Il trasferimento dell'informazione in modo parallelo è consigliato per i dispositivi veloci che si trovano fisicamente vicini, per evitare rumori e interferenze.



Schema semplificato che riporta il controllo delle linee della porta slave parallela che possiedono alcuni microcontroller PIC.

A parte la rapidità, la trasmissione parallela ha solo degli svantaggi; la maggiore quantità di linee di cui necessita suppone un costo maggiore, e il fatto che le linee su cui viaggia il segnale siano così vicine fra di loro dà origine a rumori e a interferenze, dovute anche all'elevata frequenza di funzionamento.

Nei microcontroller PIC troviamo alcuni modelli che dispongono di una porta parallela slave implementata sul silicio. Si tratta dei modelli di PIC che dispongono di cinque porte di I/O e che ne dedicano due a supportare questo tipo di comunicazione. La porta D supporta le 8 linee incaricate del trasferimento dell'informazione e la porta E le 3 linee di controllo necessarie. I segnali di controllo necessari sono RD# (lettura), WR# (scrittura) e CS# (selezione del chip).

Ogni linea del bus dei dati del microcontroller comunica con un pin esterno tramite due flip-flop tipo D, uno per la lettura e uno per la scrittura. I segnali di controllo permettono il funzionamento di un flip-flop alla volta, come mostrato nella figura.

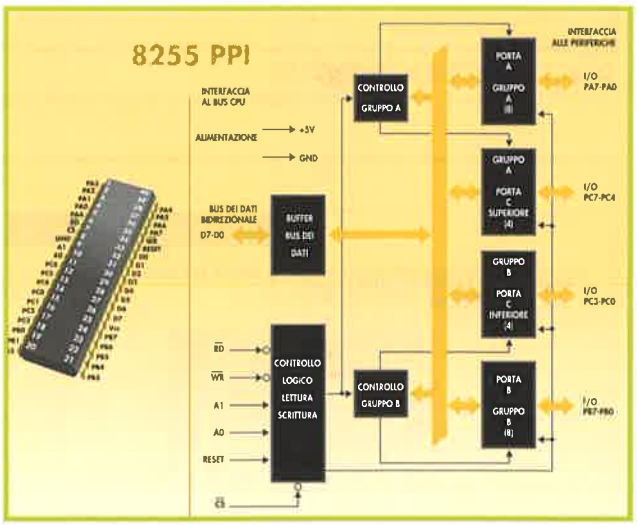
CIRCUITI INTEGRATI SPECIFICI

Per facilitare il collegamento dei sistemi basati su microprocessore, con le periferiche parallele, sono stati sviluppati una serie di circuiti integrati specifici, uno dei più conosciuti e utilizzati è il PPI 8255, prodotto da Intel.

È stato progettato per processori con il bus dei dati da 8 bit, per cui trasferisce informazioni su 8 linee. Il PPI è composto da tre porte da 8 linee cadauna, e che si chiamano PA (porta A), PB (porta B) e PC (porta C); è programmabile e il suo funzionamento si controlla mediante i bit che si scrivono sul registro di controllo.

ARCHITETTURA INTERNA E PIEDINATURA

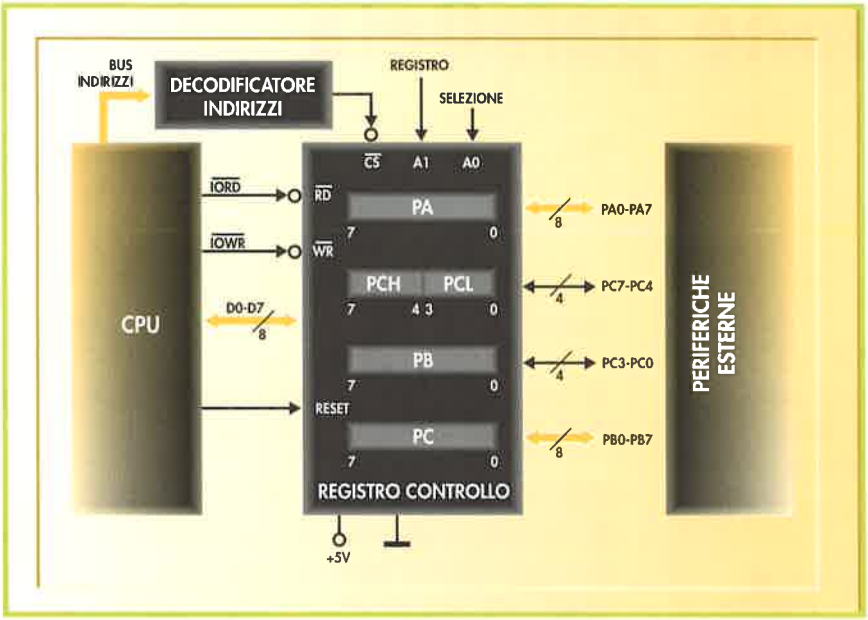
Il PPI ha un contenitore da 40 pin, la sua struttura interna e la piedinatura sono mostrate nella figura.



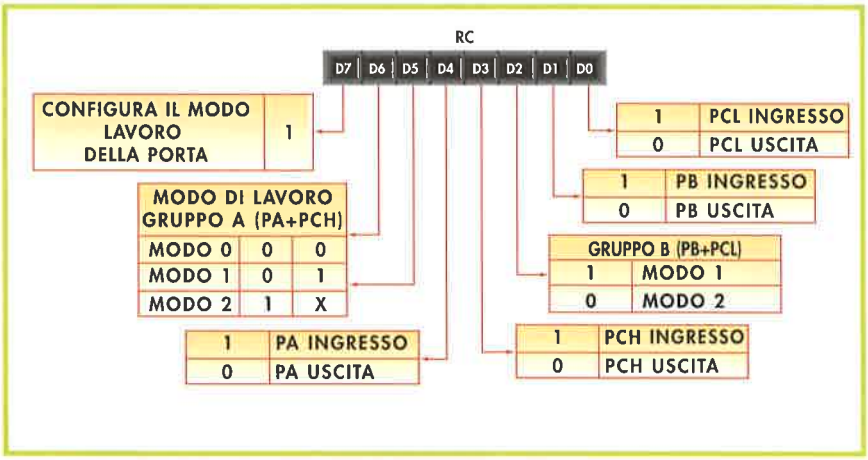
Struttura interna a blocchi del PPI e piedinatura del componente.

COMUNICAZIONI

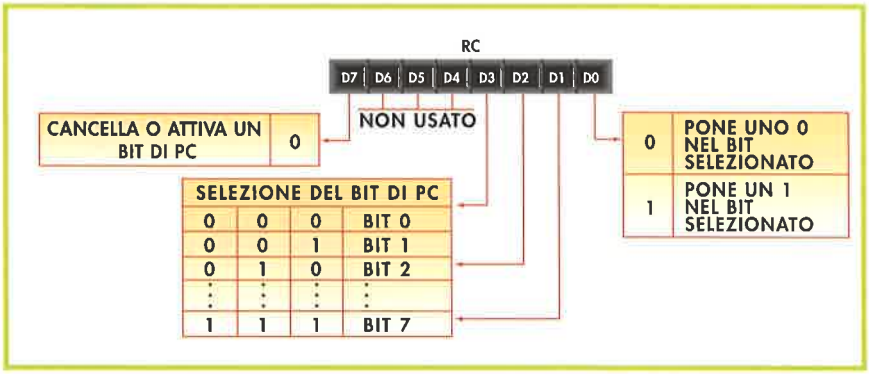
Caratteristiche e funzionamento del PPI



Il PPI si collega da una parte con le periferiche esterne e dall'altra con la CPU.



Quando il bit più significativo di RC si carica con un 1, i restanti bit determinano il modo lavoro delle tre porte.



Quando il bit più significativo di RC si carica con uno 0, i rimanenti determinano per quale delle linee della porta C esce un bit 1 o 0.

L'interno del PPI è composto fondamentalmente da quattro registri da 8 bit. Tre di questi corrispondono con i dati che entrano ed escono dalle porte A, B e C, mentre il quarto è il registro di controllo RC.

Nella figura è riportato il collegamento del PPI da un lato con le periferiche esterne, e dall'altro con la CPU con cui comunica tramite le otto linee del bus dei dati e con le linee di controllo RD# (lettura), WR# (scrittura) e CS# (selezione del chip). Quest'ultimo segnale serve per fare in modo che il PPI funzioni quando, tramite il bus degli indirizzi, arriva quello che corrisponde al PPI stesso.

Ognuna delle porte è composta da otto linee bidirezionali che ricevono il nome di PA0-PA7, PB0-PB7 e PC0-PC7. La porta C di solito lavora in due sezioni, PCH e PCL, ognuna di quattro linee. Le linee D0-D7 mettono in comunicazione il bus dei dati della CPU con i quattro registri del PPI, e le sue informazioni si scrivono o si caricano su una delle porte, selezionata con le linee di indirizzo A1 e A0. Se A1 = 0 e A0 = 0 si accede alla porta A; se A0 = 1 e A1 = 0 si accede alla porta B; se A1 = 1 e A0 = 0 si accede alla porta C e se A1 = A0 = 1 si accede al registro di controllo RC.

PROGRAMMAZIONE DEL PPI

Scrivendo nel registro di controllo si può programmare il PPI in modo dinamico, cioè si può modificare il funzionamento del PPI durante l'esecuzione di un programma, in qualsiasi momento. La scrittura del registro di controllo e programmazione del PPI permette di realizzare due azioni fondamentali:

1ª) Se il bit più significativo di RC si imposta a 1, i restanti bit di questo registro determinano il modo di lavoro delle tre porte. Esistono tre modi di lavoro: Modo 0, Modo 1 e Modo 2 e

le porte possono funzionare come ingresso e come uscita.

2°) Se si carica uno 0 nel bit più significativo di RC, i bit rimanenti servono per indicare per quale delle linee della porta C esce un bit impostato a 1 oppure a 0, come si può vedere dall'ultima figura della pagina precedente.

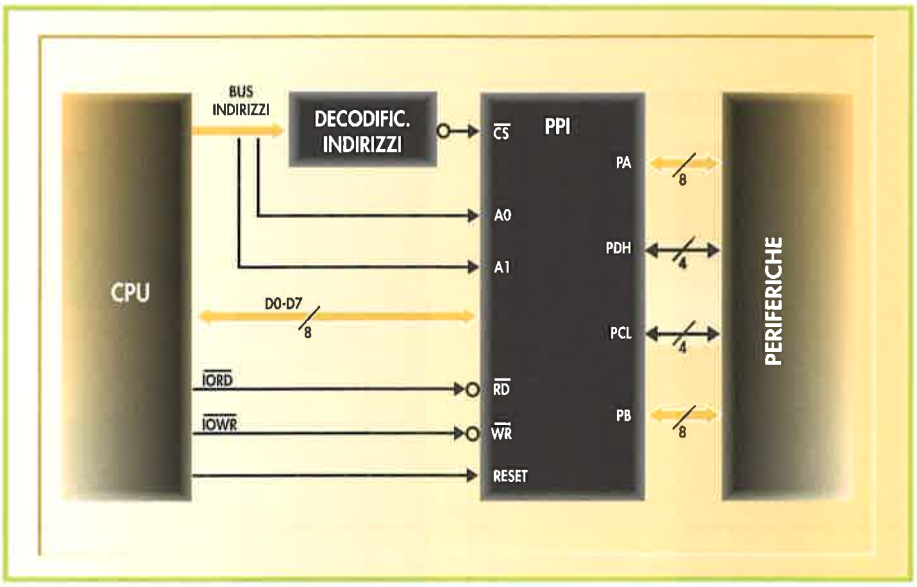


Diagramma dei collegamenti del PPI quando funziona in Modo 0, senza segnali di dialogo.

MODI DI FUNZIONAMENTO

Il PPI ammette tre modi di funzionamento delle sue porte. Il Modo 0, il Modo 1 e il Modo 2. Il modo di lavoro selezionato dipende dalle periferiche che deve pilotare. Esistono periferiche passive, o "non intelligenti", che non necessitano di segnali di dialogo, ad esempio un insieme di otto diodi LED; altri dispositivi, definiti "intelligenti" necessitano di segnali di dialogo per realizzare i trasferimenti, qualcosa come segnali del tipo "adesso ti mando un dato" e "il dato è stato ricevuto". Di seguito descriveremo il comportamento di ognuno dei modi di lavoro:

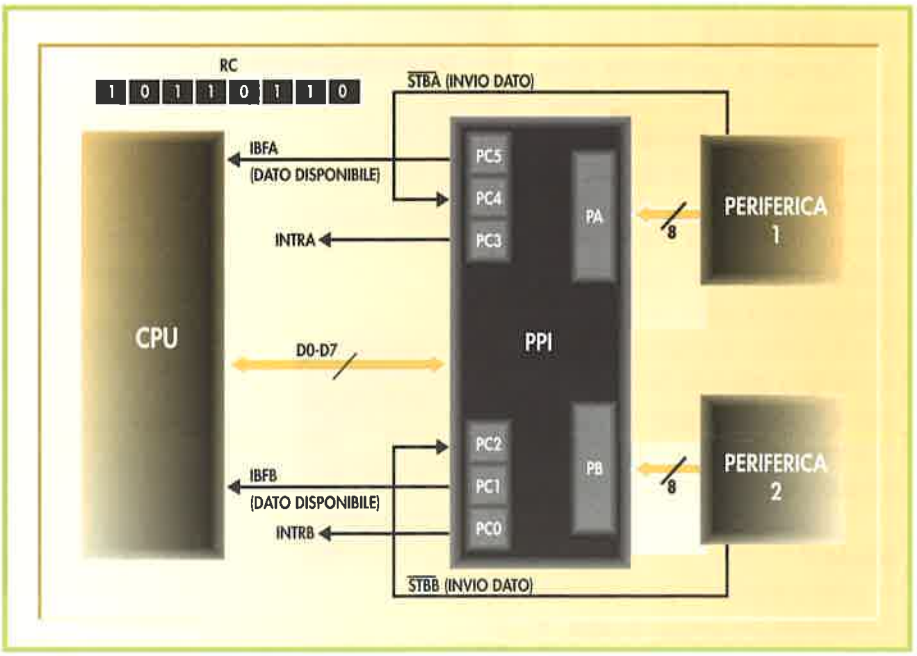
Modo 0

È il più semplice di tutti, dato che non gestisce segnali di dialogo con le periferiche, che si considerano "non intelligenti". Ad esempio, una periferica non intelligente di ingresso è un insieme di 8 interruttori, e se è di uscita un insieme di 8 diodi LED.

Dobbiamo programmare ognuna delle tre porte per farla lavorare come ingresso o come uscita, scrivendo adeguatamente i bit di RC. In questo modo possono lavorare tutte le porte PA, PB, PCH e PCL. Nella figura è riportato lo schema di collegamento del PPI nel Modo 0.

Modo 1

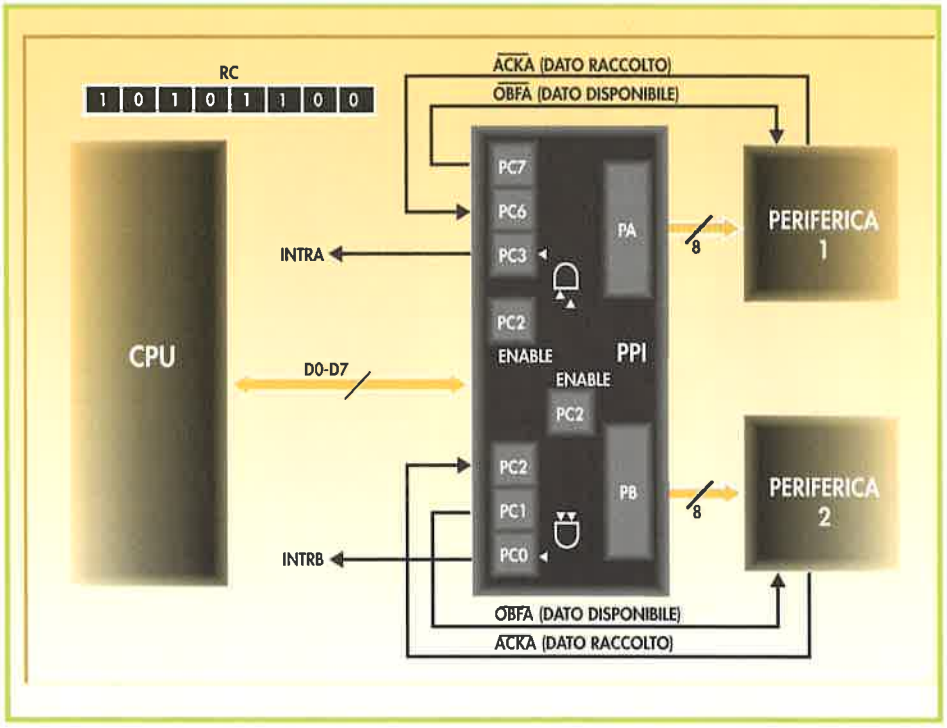
Il trasferimento di informazione è controllato dai segnali di dialogo fra il PPI, le periferiche e la CPU. Possono solo lavorare in questo modo le porte A e B dato che la porta C fornisce le sue linee per supportare i segnali di dialogo che sono necessari. Quando si utilizza una periferica di ingresso, e si carica una porta con un dato, si attiva il segnale STB# allora il PPI attiva automaticamente un segnale chiamato IBF (Buffer di Ingresso Pieno) per avvisare la CPU che c'è un dato da leggere.



Schema dei collegamenti del PPI quando lavora in Modo 1 con due periferiche di ingresso.

COMUNICAZIONI

Caratteristiche e funzionamento del PPI



Collegamenti della CPU a due periferiche di uscita tramite il PPI lavorando in Modo 1.

Inoltre in questo modo si può provocare una richiesta di interrupt alla CPU mediante il segnale INTR, per fare in modo che il segnale sia letto immediatamente. Nella figura si mostra lo schema dei collegamenti del PPI in Modo 1 con due periferiche di ingresso sulle porte A e B.

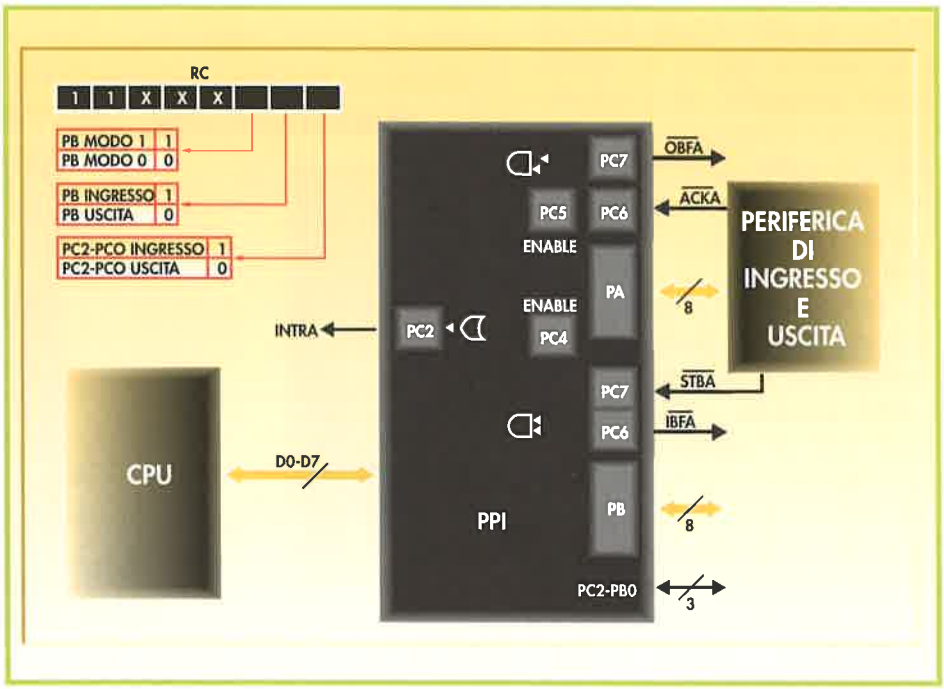
Quando le periferiche sono di uscita, i segnali di dialogo sono differenti, anche se sono sempre gestiti dalla porta C. In questo caso uno dei segnali si chiama OBF#, e il PPI lo attiva automaticamente quando la CPU scrive un dato per la porta A o B. L'altro segnale si chiama ACK# e lo attiva la periferica di uscita quando legge la porta, l'operazione comporta la disabilitazione automatica di OBF#.

Modo 2

In questo modo lavoro può solo funzionare la porta A che è supportata da cinque linee della porta C per gestire il dialogo con la periferica. Nel Modo 1 la porta A e la porta B potevano funzionare solo come ingressi, o come uscite e per ogni caso avevano bisogno di tre linee di controllo, contando anche quella di interrupt.

Nel Modo 2 la porta A può lavorare in modo bidirezionale ed esige 5 linee di dialogo, per cui alla porta B rimangono a disposizione solo 3 linee libere sulla porta C, rimanendo limitata a lavorare in Modo 1.

Quando PA funziona in Modo 2 come ingresso, utilizza i segnali IBF# e STB#, mentre se funziona come uscita utilizza OBF# e ACK#. Inoltre necessita della linea INTR per la possibile richiesta di interrupt alla CPU. Se la periferica introduce un dato in PA si attiva STB# e IBF# si pone a 1 automaticamente, in modo che quando la CPU legge il dato si disattiva IBF. Se la CPU scrive un dato in PA si attiva OBF# e quando la periferica raccoglie questo dato attiva ACK# e si disattiva OBF#.



Schema dei collegamenti fra la CPU e una periferica di ingresso/uscita tramite il PPI in Modo 2.

Comunicazione parallela

GLI INCONVENIENTI DELLA COMUNICAZIONE SERIALE RS-232-C

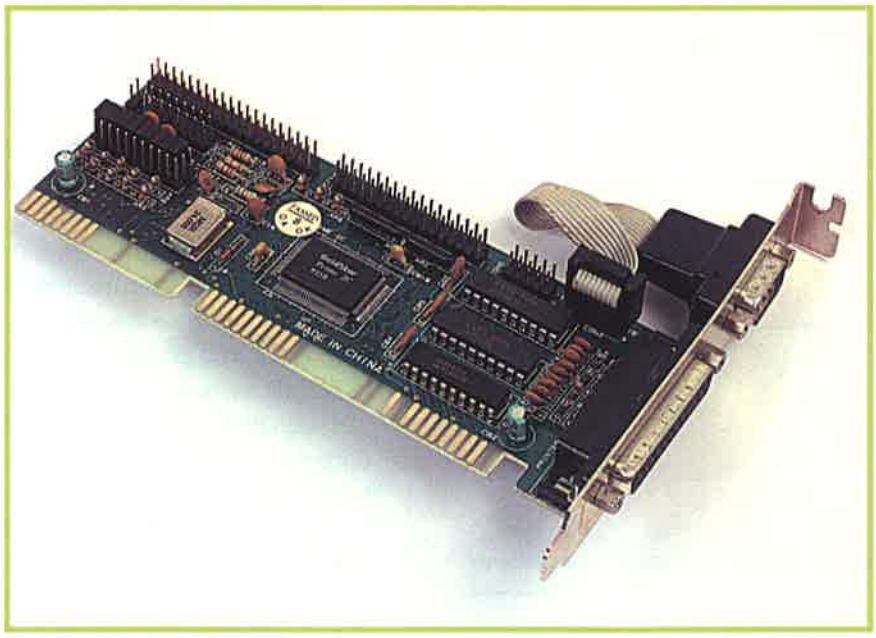
La comunicazione dei dati in serie secondo la normativa RS-232-C fu introdotta nel 1962, e permette il trasferimento dei dati fra un trasmettitore e un ricevitore. Sin dai primi anni '90, la maggior parte dei personal computer o PC, e anche computer di categorie diverse, comprendevano almeno una porta seriale con interfaccia RS-232-C, secondo le norme dell'EIA dell'anno 1969. Anche se questo standard è ancora abbastanza diffuso, ha una serie di limitazioni che impediscono la sua applicazione in molti campi, fra esse ricordiamo:

1ª. La velocità massima è limitata a 20 Kbits/s, il che ne impedisce l'uso nelle applicazioni dove sia necessario trasferire grandi quantità di dati.

2ª. La distanza massima fra l'emettitore ed il ricevitore è di 15 metri.

3ª. Nel trasferimento possono essere coinvolti solo un trasmettitore e un ricevitore.

4ª. Il rumore e le interferenze elettromagnetiche influenzano molto questo tipo di trasferimento, e il motivo principale risiede nel fatto che si utilizza il "modo comune", che consiste nell'inviare segnali il cui valore ha come riferimento la massa, quindi costantemente influenzati da disturbi.



Una tipica scheda con porta seriale del PC.

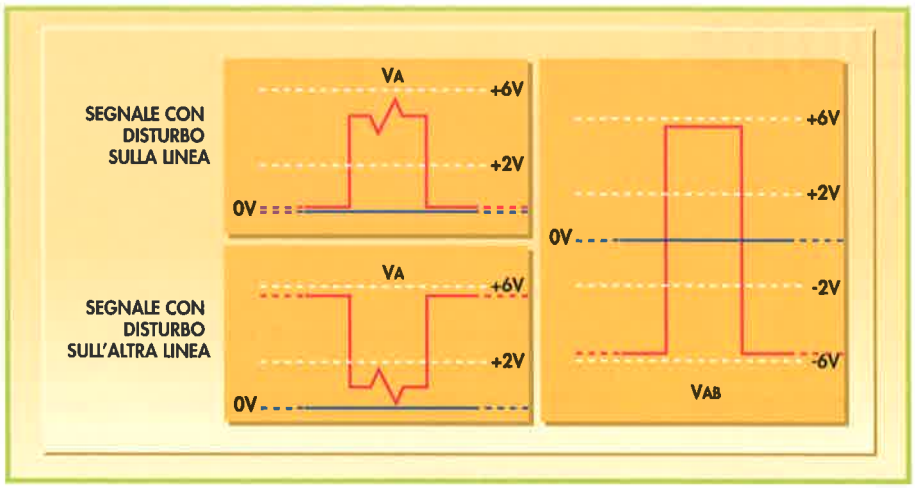
NUOVE NORME DI COMUNICAZIONE

Le restrizioni della comunicazione RS-232-C hanno dato luogo allo sviluppo di nuove norme, destinate a coprire aree specifiche. Così ad esempio l'ultima revisione della norma RS-232-C, chiamata EIA-232-F, permette la comunicazione fra due elementi ad una distanza di 15 metri con una velocità massima di 19,2 Kbit/s.

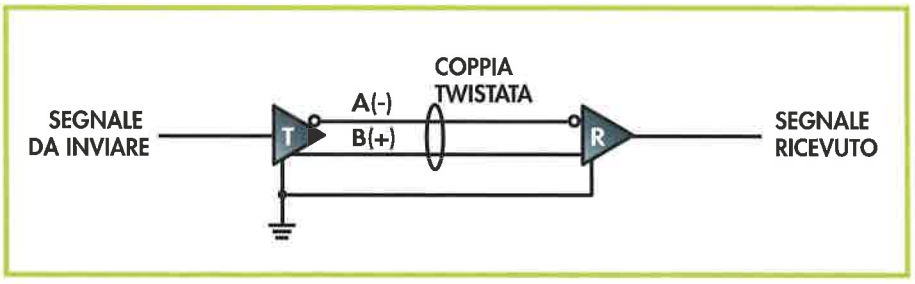
Nella norma EIA-423-B l'emettitore lavora in modo comune, però il ricevitore lo fa in modo differenziale, che praticamente annulla gli effetti dovuti al rumore e alle interferenze. Il modo differenziale verrà spiegato più avanti, però fondamentalmente è basato sulla trasmissione di segnali come differenza di due tensioni al posto di una tensione rispetto a massa. Nel modo differenziale si utilizzano due cavi twistati, quindi ogni segnale ha bisogno di due conduttori. Il livello logico alto e basso dei bit dei dati, è determinato dalla differenza di potenziale fra i due conduttori, e non da quello di una linea rispetto a massa. Nella norma EIA-423-B è previsto un solo trasmettitore, ma sono possibili diversi ricevitori differenziali. La distanza massima può arrivare a 1200 metri con una velocità di 1 Kbit/s. Se si eleva la frequenza di trasmissione diminuisce la distanza, così quando si trasferisce informazione a 100 Kbit/s la distanza massima è ridotta a 12 metri. Esistono anche altre norme, come la EIA-561, 562, 694 e 723, però le più importanti probabilmente sono la RS-422 e la RS-485, che saranno descritte in modo più approfondito.

COMUNICAZIONI

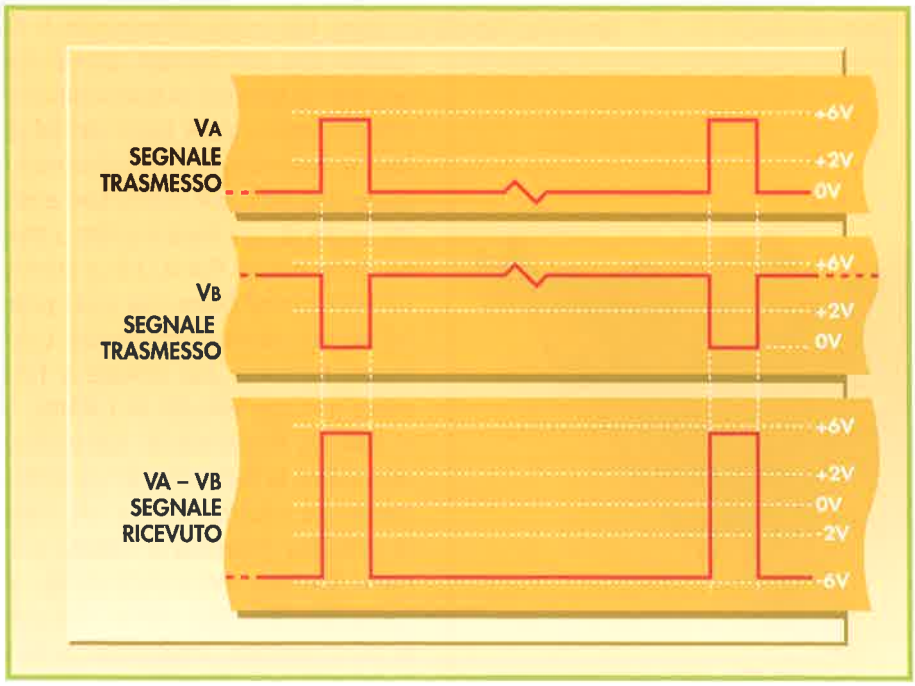
Altre regole della comunicazione seriale



La differenza di potenziale fra le due linee elimina le interferenze.



Collegamento di un trasmettitore e di un ricevitore in modo differenziale



Il segnale trasmesso sui due cavi viene sottratto per ottenere il segnale ricevuto senza rumore.

LA TRASMISSIONE IN MODO DIFFERENZIALE

L'obiettivo di una buona norma è trasferire dati a frequenza elevata fra punti molto distanti. In ambiente industriale con alti livelli di rumore e interferenze, è difficile far convivere i due parametri principali: frequenza e distanza. Dato che la normativa RS-232-C non raggiunge questo obiettivo sono state sviluppate norme che utilizzano il modo differenziale invece del modo comune.

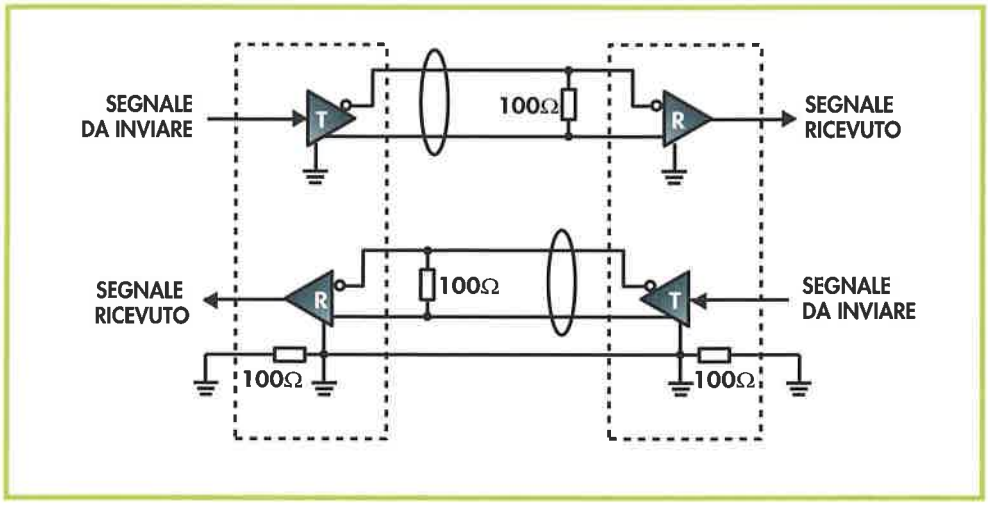
Nel modo comune, i livelli logici alto e basso che rappresentano i bit dei dati e di comando, sono determinati dalla tensione esistente sulla linea rispetto alla massa comune. Nel modo differenziale, i segnali non circolano su un cavo solo ma su due, e i livelli logici sono determinati dalla differenza di tensione fra essi. Dato che il rumore influenza in modo uguale le due linee, la differenza fra esse lo annulla e lo elimina, come si può vedere dalla figura.

La comunicazione in modo differenziale di un trasmettitore e di un ricevitore è rappresentata graficamente con la simbologia implementata nella figura. Il segnale da inviare e il segnale ricevuto possono essere di livello TTL, dopo i due dispositivi trasmettono e ricevono questa informazione in modo differenziale tramite i due cavi, A e B, che prendono il nome di coppia twistata.

I segnali sui due cavi del modo differenziale sono invertiti (notate i cerchietti) e la sottrazione fra loro non solo azzera il rumore e le interferenze, ma raddoppia il livello del segnale risultante, come si può vedere graficamente nella figura.

COMUNICAZIONI

Altre regole della comunicazione seriale



Collegamento fra due dispositivi che utilizzano la norma RS-422.

Maggiore è la distanza che separa l'emettitore e il ricevitore, minore sarà la velocità di trasferimento. A 1200 metri la velocità massima è di 100 Kbit/s, a 120 metri è di 1 MHz/s e a 12 metri può arrivare a 10 Mbit/s. Con l'obiettivo di poter collegare fra loro vari dispositivi in rete, è necessario che la sezione di trasmissione di ogni dispositivo sia ad alta impedenza, quando non è in funzione. In questo modo si può trasmettere con un solo dispositivo e averne un insieme di 10 in ascolto.

NORMA RS-422

Un collegamento secondo la norma RS-422 necessita di cinque linee, o cavi: due per trasmettere l'informazione in modo differenziale, altri due per riceverla nello stesso modo, e l'ultima per la massa comune. La tensione di ogni cavo in modo comune può essere compresa fra +7 V e -7 V rispetto a massa.

NORMA RS-485

È molto simile alla RS-422, di seguito riportiamo alcune fra le principali differenze:

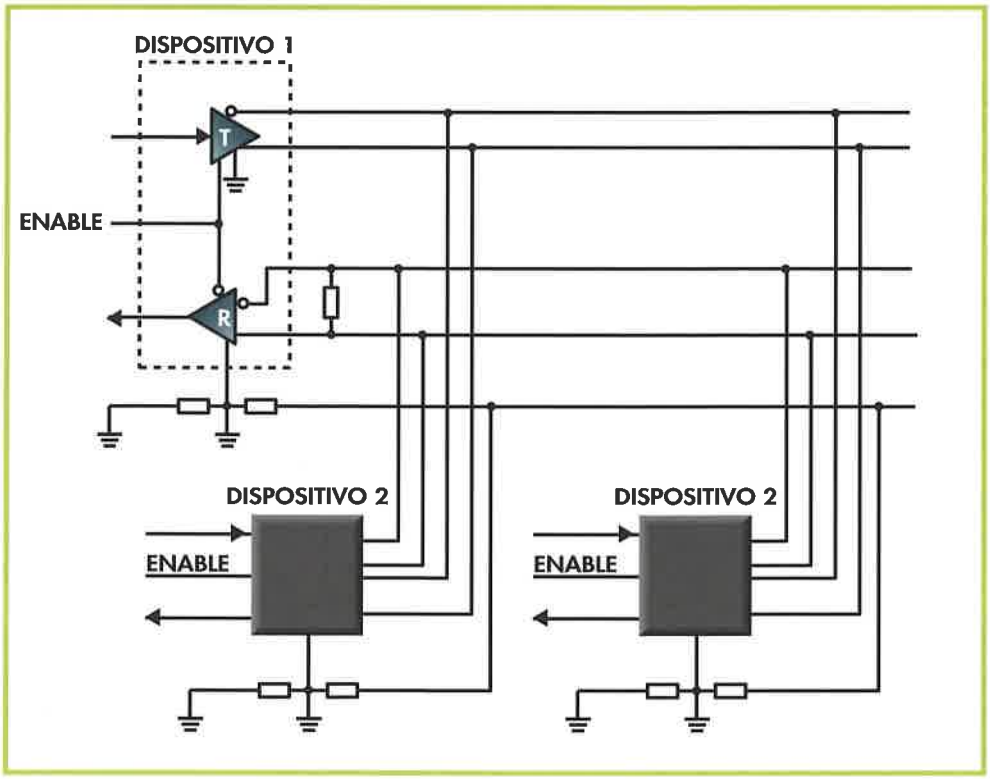
1ª. Nella stessa rete possono funzionare sino a 32 sistemi emettitore/ricevitore.

2ª. Il margine di tensione in modo comune può andare da +12 V a -7 V.

3ª. Se si cortocircuitano i due cavi twistati A e B, o se si applica fra essi una tensione fra +12 V e -7 V, la corrente massima che circola sarà inferiore a 250 mA per evitare che si danneggino.

4ª. Il trasmettitore e il ricevitore, nella normativa RS-485, dispongono di un segnale di ENABLE che abilita o disabilita il funzionamento. In questo modo i dispositivi che non stanno emettendo, o ricevendo, possono essere scollegati dalla rete.

5ª. L'impedenza minima di ingresso deve essere superiore a 10,56 KΩ.



Collegamento di vari dispositivi RS-485 in rete.

COMUNICAZIONI

Altre regole della comunicazione seriale

PARAMETRO	RS-232_C	RS-422	RS-485
Massima distanza (metri)	15	1.200	1.200
Massima velocità (Kbit/s)	20	10.000	10.000
Modo di funzionamento	COMUNE	DIFFERENZIALE	DIFFERENZIALE
Numero emettitori/ricevitori	1/1	1/10	1/32
Massima tensione di uscita	±25V	–0,25V +6V	–7 +12V
Sensibilità di ingresso del ricevitore	±3V	±200mV	±200mV
Resistenza di ingresso del ricevitore	3k - 7k	4k min	12k min

Principali specifiche delle norme RS-232, RS-422 e RS-485.

LA RETE INDUSTRIALE RS-485

Poiché la distanza che permette questa rete può superare il chilometro, è molto adatta alle installazioni industriali per la raccolta dei dati e l'attivazione di differenti attuatori.

Se non si utilizzano ripetitori intermedi si possono collegare sino a 32 unità di carico, con due resistenza di terminazione da 120 Ω. Questo suppone che ogni emettitore RS-485 debba fornire una corrente di 57 mA, così le 32 unita di carico insieme alle resistenze di terminazione consumano 1 mA ognuna. L'impedenza massima di ingresso sarà superiore a 10,56 KΩ, in modo che applicando all'ingresso una tensione di 12 V, la corrente non sia superiore a 1 mA.

Quando l'ambiente è molto ostile, si raccomanda di utilizzare del cavo schermato, con la calza collegata a terra da uno degli elementi della rete.

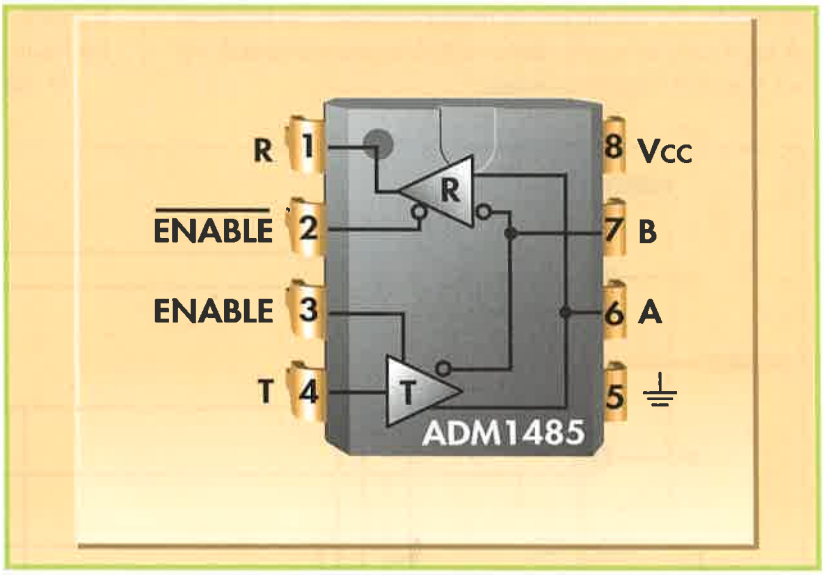
Il numero di dispositivi RS-485 nella rete, si può aumentare in modo considerevole con l'utilizzo di ripetitori e ricevitori ad alta impedenza.

Nella tabella in alto si pone in rilievo la differenza fra i diversi parametri delle norme RS-232, RS-422 e RS-485.

CIRCUITI INTEGRATI SPECIFICI

Dato il diffuso utilizzo di dispositivi accoppiati allo standard RS-485, i costruttori di circuiti integrati, dispongono di una linea di prodotti destinati a risolvere l'adattamento dei segnali TTL/CMOS a questa normativa, e uno dei componenti più rappresentativi è il modello ADM1485 di Analog Devices. Si tratta di un circuito integrato a 8 piedini che permette di soddisfare le norme RS-485 e RS-422, e la cui piedinatura è riportata nella figura riprodotta qui sotto.

L'ADM1485 è un trasmettitore/ricevitore in modo differenziale, progettato per l'accoppiamento alle norme RS-485 e RS-422. Ognuna delle sue sezioni si può abilitare in modo indipendente con il segnale ENABLE. Lavora con un range di tensione compreso fra +12 V e -7 V. Permette velocità di trasferimento che arrivano sino a 30 Mbit/s, e si alimenta con una tensione unica da +5 VDC. Il tempo di propagazione del trasmettitore è di 10 ns e quello del ricevitore di 25 ns. Permette la connessione diretta di 32 dispositivi in rete, anche se è possibile attivare un solo trasmettitore per volta.



Piedinatura del circuito integrato ADM1485.

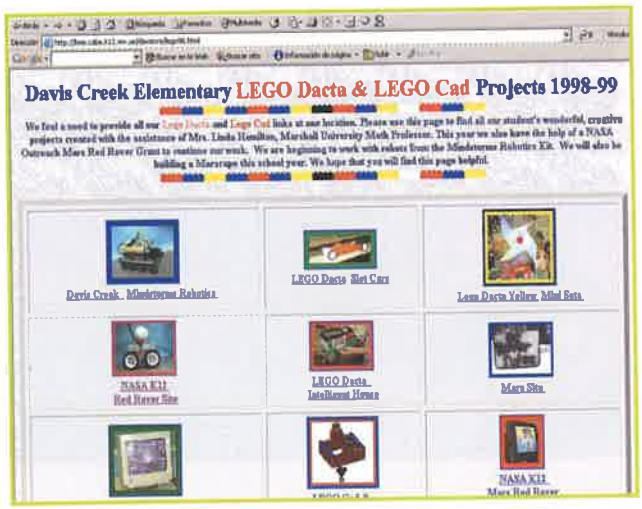
Indirizzi utili di Internet

GRUPPI DI NOTIZIE, FORUM E LISTE RELATIVE ALLA ROBOTICA E ALLA PROGRAMMAZIONE

http://www.cs.uu.nl/cgi-bin/faqwais.mailgate.org/comp/comp.robotics.misc
http://www.mailgate.org/sci/sci.electronics.basics
http://www.robot.ireq.ca/CRR/Archives/1995/author.html
http://ftp.ncnu.edu.tw/Documentation/faq/COMP-ANSWERS/
http://www.cs.uu.nl/cgi-bin/faqwais
http://www.redeya.com/foros/
http://www.elistas.net/lista/todoelectronica

PROGETTI DI MICROROBOTICA E SVILUPPO CON PIC

http://boe.cabe.k12.wv.us/daviscre/lego98.html
Progetti con Lego dalla Marshall University.



http://robotics.eecs.berkeley.edu/~ronf/milli-robot.html
Progetti di microrobot dell'Università di Berkeley.

http://people.cs.uchicago.edu/~firby/aap/
Progetto di intelligenza artificiale dell'Università di Chicago.

http://www.gsystech.de/
Progetti per il PIC.

http://cherokee.iespana.es/cherokee/Microbotica_Prog.htm
Sezione in cui si pubblicano informazioni relative alla programmazione di microcontroller, oltre a varie utility ed esempi di programmazione. In questo momento possiamo trovare un manuale di JAL (Just Another Language), un linguaggio di programmazione di alto livello per i microcontroller PIC che assomiglia molto al Pascal.

http://www.al-williams.com/wd5gnr/stampfaq.htm
http://www.stampsenclase.com/html_files/basic_stamp.asp
Pagine in cui si possono trovare manuali, e risposte alle domande più frequenti (FAQ) sui moduli Basic Stamp, basati sui microcontroller PIC e molto utilizzati nella costruzione di microrobot.

CONCORSI

http://www.sia.eui.upm.es/championbot
Universidad Politecnica de Madrid.

KITS DI MICROROBOT

http://www.robix.com/
Kits per la costruzione di robot, dal livello scolastico secondario al livello universitario.

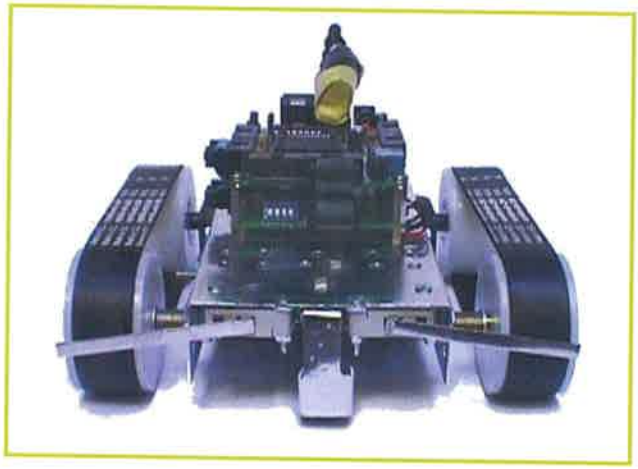
COMUNICAZIONI

Internet

http://www.robotics.com
Kits di robot, notizie, documenti.

http://robotikitsdirect.com/index.html
Kits di robot.

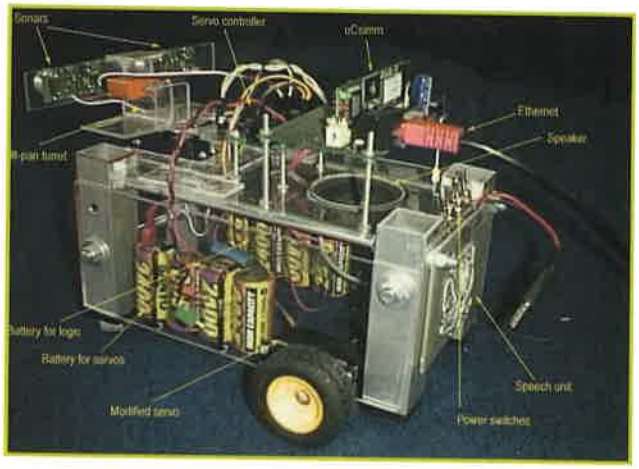
http://www.irobot.com/home/
Sistemi robotici reali.



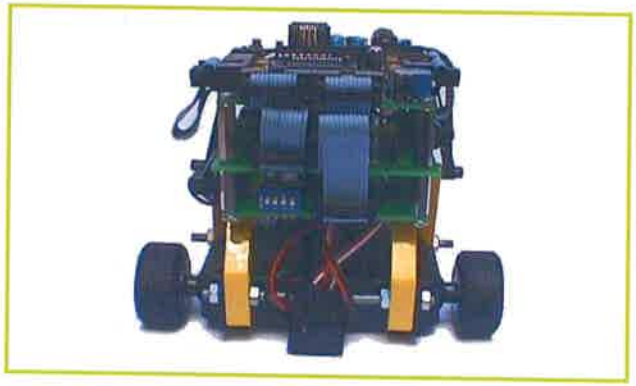
http://www.joker-robotics.com/

http://www.lynxmotion.com/
Kits di robot e braccia robotiche.

http://www.microbotica.es/tower.htm
http://www.microbotica.es/flota.htm
Schede e microrobot completi basati sul 68hc11 di Motorola.

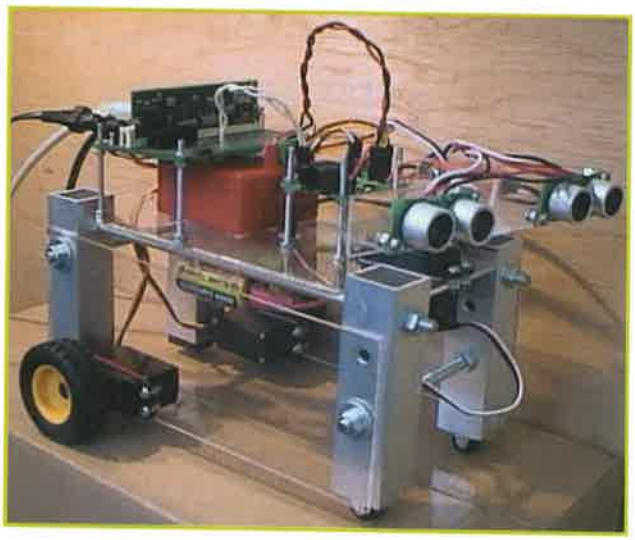


http://www.stampsenclase.com/
Prodotti di robotica, manuali ecc. basati sul Basic Stamp di Parallax.



http://www.sciencekits.com/robots.htm
Kits di robot per differenti livelli.

http://www.robotstore.com
Offerta di numerosi kit per differenti livelli di conoscenza su hardware, elettronica e programmazione. Possiamo trovare da microrobot con gambe, ali e braccia mobili, sino a robot con telecamere di visione o funzionanti a pannelli solari. Inoltre sono in vendita kit singoli, come ad esempio il sonar, adatti ad essere montati su diversi microrobot.

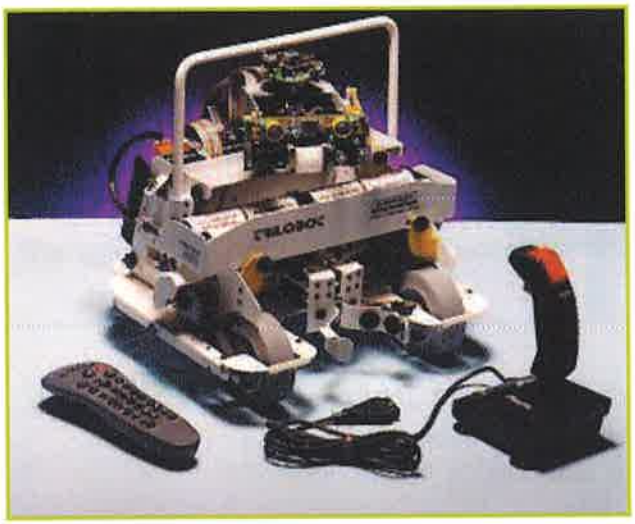


http://www.robot-electronics.co.uk/
Svariati modelli di moduli: ad ultrasuoni, scrittori di PIC, ecc. Componenti: servomotori, sensori, ecc.

GRUPPI DI ROBOTICA

http://www.eupmt.es/cra/
Gruppo di robotica di Matarò. EUPMt.

http://www.robotgroup.org/
Pagina di un gruppo con diverse cose e links.



http://www.dprg.org/
Club di Dallas.

http://www.cs.columbia.edu/robotics/
Gruppo di robotica dell'Università della Columbia

http://www.dai.ed.ac.uk/groups/mrg/MRG.html
Gruppo dell'Università di Edimburgo

UNIVERSITÀ E ISTITUTI DI RICERCA CHE LAVORANO NEL CAMPO DELLA ROBOTICA

http://www.robotics.uc.edu/
Università di Cincinnati.

http://www.ri.cmu.edu/
Istituto di robotica di Carnegie Mellon.

http://www.cs.uchicago.edu/research/ai/index.html
Università di Chicago.

http://www.eng.fiu.edu/me/robotics/
Laboratorio dell'Università della Florida.

http://www.ual.es/~jcmoreno/CRUAL2001.html
Accesso condizionato da autorizzazione
Università di Almería.

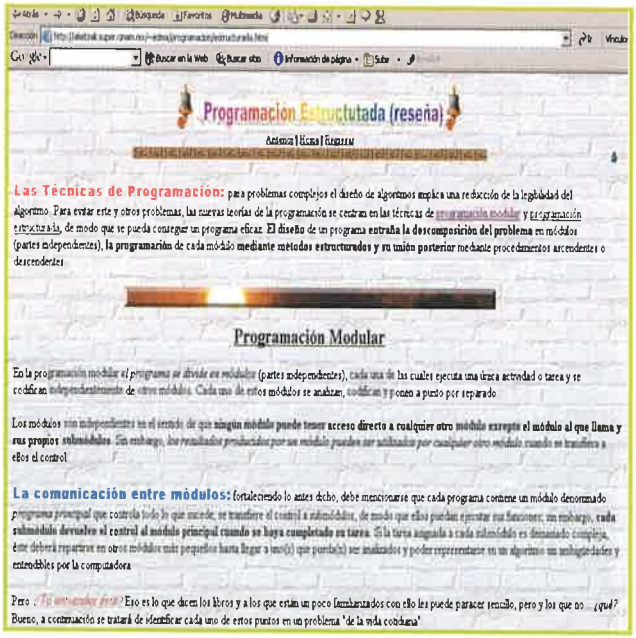
http://obelix.umh.es/picbot/index.htm
reindirizzato all'indirizzo seguente

http://apolo.umh.es
Università Miguel Hernández (Escuela Politecnica de Elche).

CORSI E CONCETTI DI ROBOTICA E PROGRAMMAZIONE

Breve spiegazione riguardo alla programmazione strutturata.Spiega con un semplice e conciso esempio in cosa consiste la programmazione strutturata.

http://www. infor.uva.es/~descuder/docencia/pd/node59.html
Argomenti di un corso di introduzione alla programmazione strutturata.



http://www.puc.cl/curso_dist/cbc/textos/teoria/lengua1.html
Accesso condizionato da autorizzazione partire dall'indirizzo seguente:

http://puc.cl
Informazioni complete sui linguaggi di alto e basso livello, programmazione strutturata, struttura di un linguaggio.

http://www.coqui.lce.org/cadiaz/CEDU5120V/cedu5120S.htm
Interessante pagina dove si illustra un corso su un linguaggio di programmazione strutturata, tenuto dal professor Carlos A. Diaz Aponte.

http://www.itlp.edu.mx/publica/tutoriales/pascal/u1_1_4.html
Descrive le strutture di programmazione più comuni.

http://mailweb.pue.udlap.mx/~asanchez/is150/is/19.htm
Accesso condizionato da password. Tutti i passaggi per realizzare un programma

http://www.abcdatos.com/tutoriales/ofimatica.html
In questo sito si può trovare un corso di metodologia della programmazione inerente alle strutture, agli operatori ecc., che tratta sia gli aspetti teorici sia quelli pratici.

STRUMENTI DI SVILUPPO PER IL PIC

http://www.microchip.com
Costruttore dei microcontroller PIC.

http://www.arroweurope.com/
Distributore di prodotti per PIC in Europa.

http://www.parallaxinc.com/
Strumenti di sviluppo per il BASIC Stamp, basati sul PIC.

http://www.siriusmicro.com/
Azienda che produce strumenti di sviluppo per microcontroller PIC.

http://www.rentron.com/
Strumenti per il PIC.

http://www.controlled.com/pic.html
Lista di indirizzi di aziende fornitrici di prodotti relativi al PIC.

http://www.vmdesign.com/
Simulatore di microcontroller.