Tecniche Automatiche di Acquisizione Dati

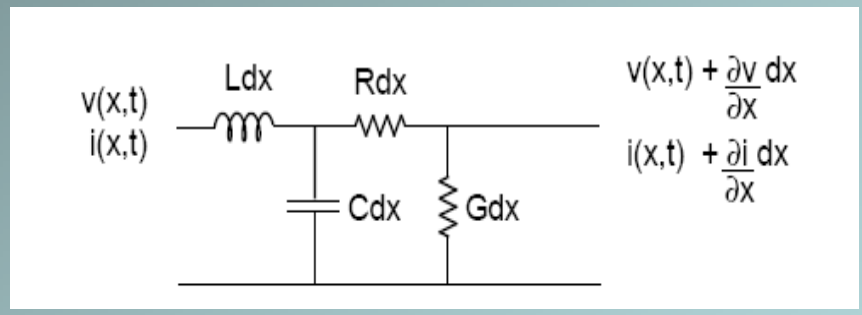
Trasmissione delle informazioni

Generalità

- La trasmissione delle informazioni tra i componenti di un medesimo elaboratore o fra più elaboratori avviene tramite un mezzo fisico (cavi di rame, fibra ottica, radio, etc.) mediante il quale si ottiene una connessione (link).
- Il modo in cui i segnali ed i dati sono organizzati e si propagano attraverso il link definisce un **protocollo** di comunicazione.
- I protocolli possono sommarsi e dar vita a protocolli più complessi (pile di protocollo) e permettere all'utente finale di trasmettere dati nello stesso modo su differenti link fisici (un esempio di pila di protocolli e TCP/IP su Ethernet, o su Fibre Channel)

Le linee di trasmissione

- Per linea di trasmissione si intende in genere una coppia di conduttori attraverso cui si propaga un segnale. I due conduttori possono essere paralleli (come in una piattina), coassiali, o composti da un conduttore ed un piano di massa (come nei circuiti stampati.
- Un tratto infinitesimo di linea si può schematizzare come in figura, in cui i parametri L, R, C e G sono intesi per unità di lunghezza



Dette Z=R+iωL e Y=G+iωC ed applicando le leggi di Kirkhoff otteniamo una coppia di equazioni delle onde:

$$\frac{\partial^2 V}{\partial x^2} = \gamma_0^2 V; \quad \frac{\partial^2 I}{\partial x^2} = \gamma_0^2 I; \quad \gamma_0 = \sqrt{ZY}$$

Linee di trasmissione II

- Se R e G sono trascurabili (linea non dissipativa):

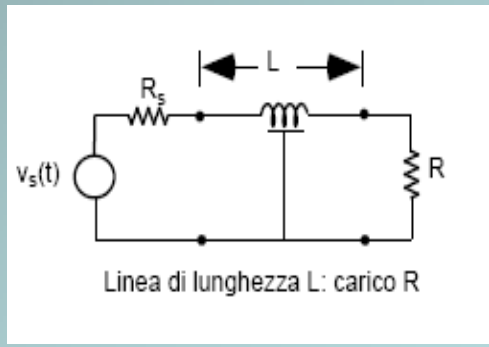
$$\frac{\partial^2 \mathrm{v}}{\partial x^2} = LC \frac{\partial^2 \mathrm{v}}{\partial t^2}$$

Quindi un equazione delle onde con velocità u=1/√LC con soluzioni del tipo:

$v(x, t)= f_1(x-ut) + f_2(x+ut)$

$i(x, t)= 1/R_0[f_1(x-ut) - f_2(x+ut)]$

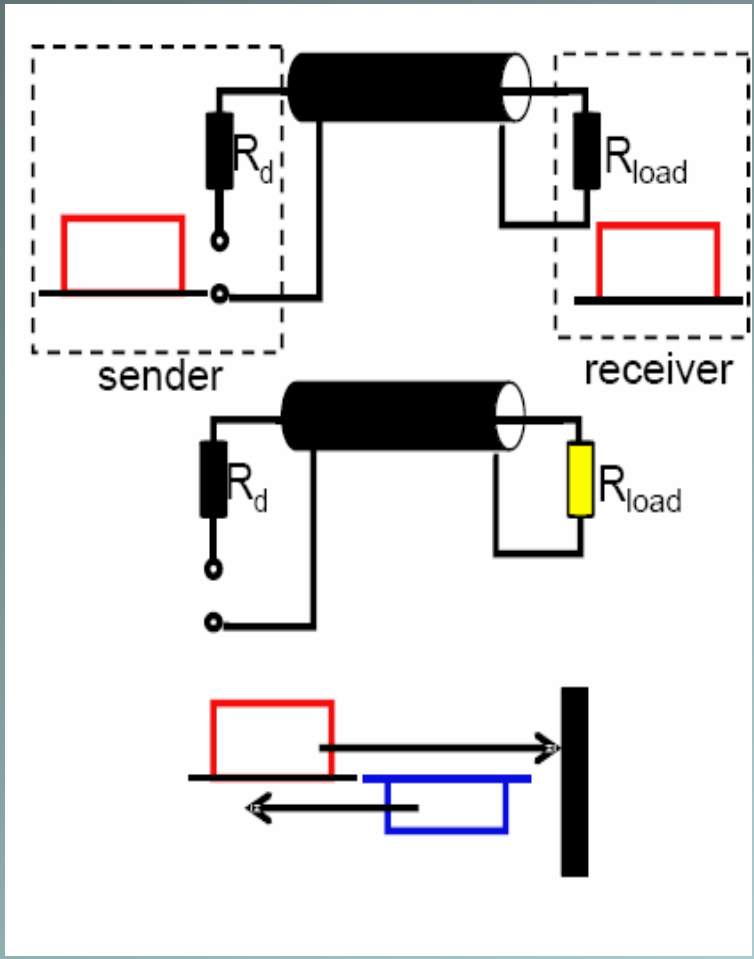
Ove $R_0= \sqrt{(L/C)}$ è l'impedenza caratteristica della linea.



Linea di lunghezza L: carico R

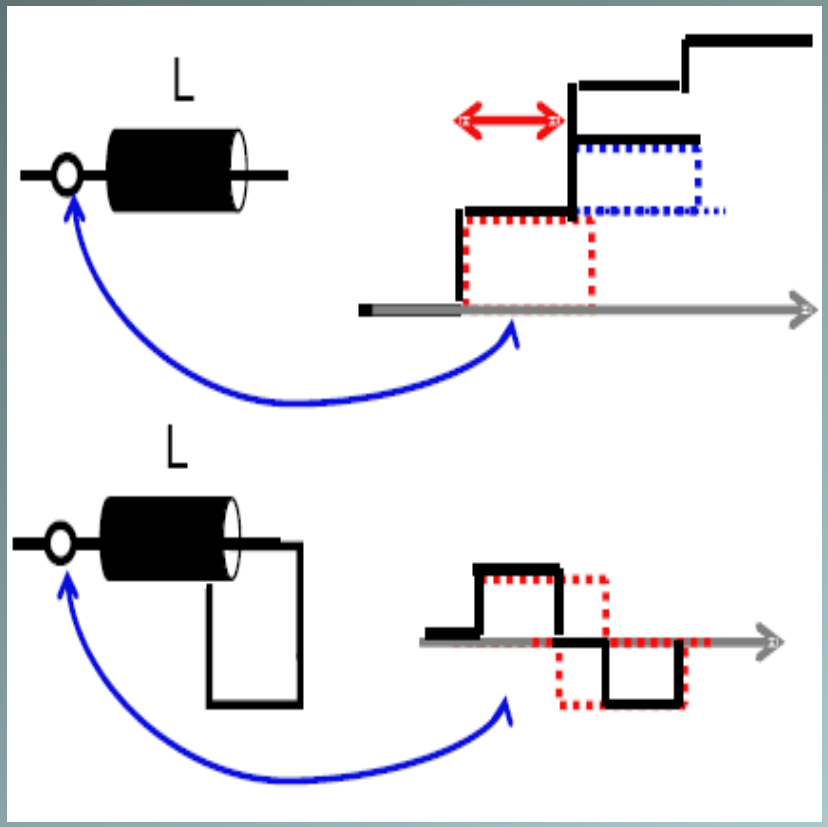
Se chiudiamo la linea su una resistenza R, nel punto alla fine della linea, la corrente sarà v/R => $i(L, t)= 1/R_0[v_s- \rho v_s] = 1/R[v_s+ \rho v_s]$
Da cui si ricava il coefficiente di riflessione:
$\rho= (R-R_0)/(R+R_0)$.
Se $R=R_0$ non c'è riflessione:**Linea adattata**
R è la **terminazione**

Adattamento di impedenza



- Se l'impedenza è adattata il cavo appare infinitamente lungo: nessun segnale riflesso
- Se l'impedenza è disadattata un segnale riflesso al termine della linea si sovrappone al segnale inviato.

Ancora adattamento



- Linea aperta: $R_{load}=\infty$ l'ingresso e la riflessione hanno uguale polarità.
- Si sovrappongono per $t=2T=2l/c$

- Linea cortocircuitata: $R_{load}=0$ l'ingresso e la riflessione hanno opposta polarità.
- Si sovrappongono per $t=2T=2l/c$
- Segnale bipolare

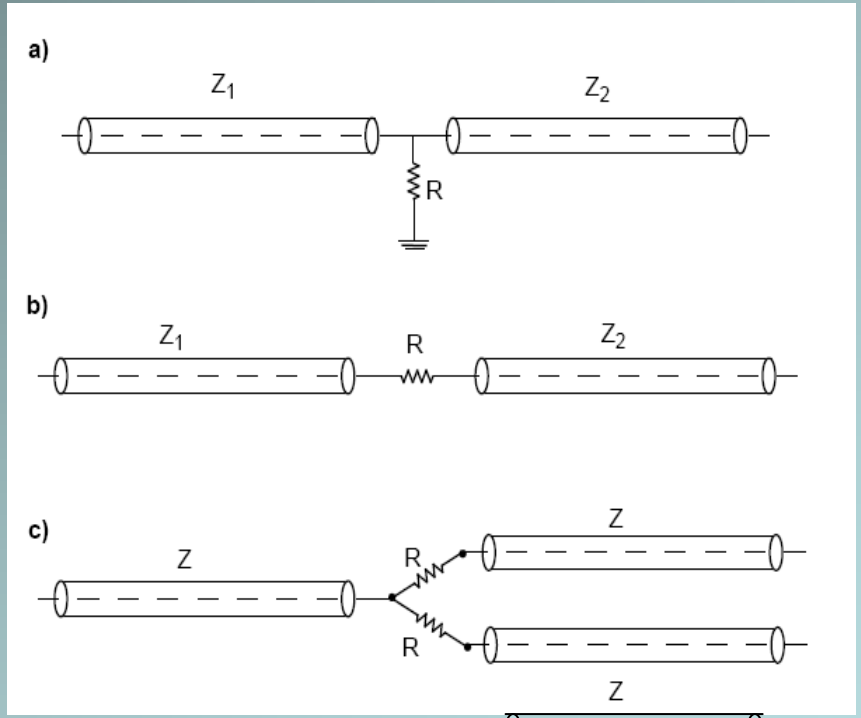
Giunzioni e divisioni di linee

a) $Z_1<Z_2$ $R=Z_1Z_2/(Z_1+Z_2)$
b) $Z_1>Z_2$ $R=Z_2-Z_1$
c) Se la divisione è in n rami bisogna imporre che $(R+Z)/n=Z$ => $R=(n-1)Z$.

Per n=2; R=Z

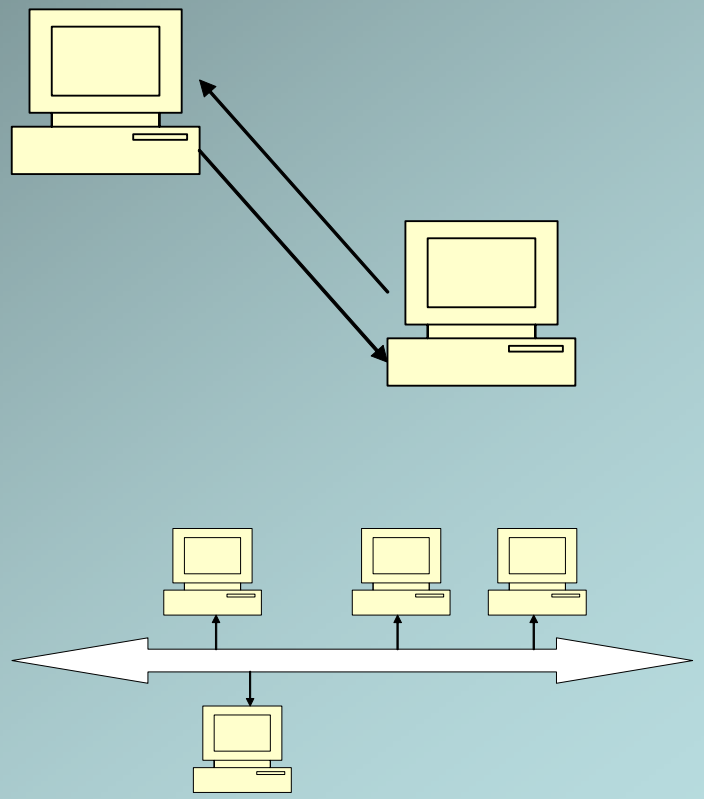
Però in questo caso le due linee aggiunte non vedranno un'impedenza Z. Va usato uno splitter simmetrico con 3 resitenze uguali: R=Z/3 (per Z=50Ω, R=17Ω)

In generale R=Z(N-1)/(N+1)



Trasmissione punto-punto e bus

- Nella trasmissione punto-punto due dispositivi sono connessi fra loro e con nessun altro attraverso il link (esempio: un monitor con la scheda grafica attraverso il cavo VGA)
- I bus connettono più dispositivi tra loro mediante il medesimo link (esempio: più dischi con il computer e fra loro tramite il bus SCSI o IDE).



Trasmissione seriale e parallela

- La trasmissione dei dati si dice seriale, se questi sono trasmessi attraverso il link un bit alla volta, in sequenza. È questo il caso, per esempio, quando è presente un solo cavo per la trasmissione.
- La trasmissione si dice parallela quando tutti i bit di una parola sono trasmessi contemporaneamente, in parallelo. Per ottenere ciò, è necessario avere a disposizione tanti cavi (o canali) quanti sono i bit della parola.

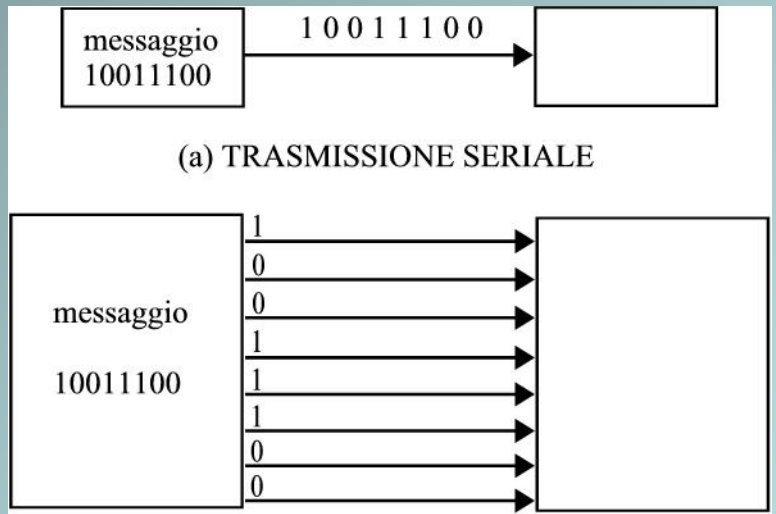


Figura 2 - Trasmissione seriale e parallela

Full duplex/Half duplex

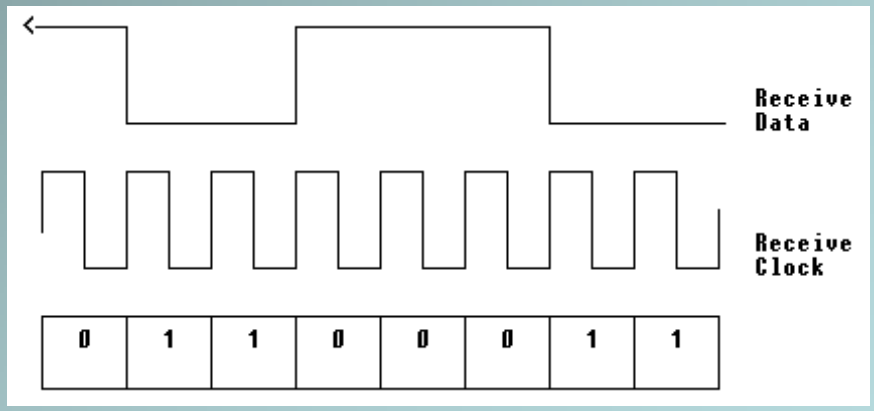
- Una linea di comunicazione di dice *full duplex* se è in grado di ricevere e trasmettere contemporaneamente – ci sono due canali: uno in ingresso l'altro in uscita.
- Si dice *half duplex* se non può trasmettere e ricevere allo stesso tempo. C'è un solo canale su cui i dati possono viaggiare in un senso o nell'altro.

Rivelazione degli errori – bit di parità

- Può capitare che i bit trasmessi risultino alterati al momento della ricezione, a causa di interferenze o errori nella trasmissione.
- C'è bisogno di un metodo per permettere la rivelazione di errori.
- Si possono trasmettere assieme ai dati uno o più bit di controllo.
- Un metodo comunemente usato è il bit di parità: esso vale 1 se il numero di bit con valore 1 compreso il bit di parità è pari (parity even), 0 se dispari (parity odd).
- Se in ricezione il bit di parità calcolato è diverso da quello ricevuto, un bit deve essere cambiato nel tragitto o erroneamente trasmesso .
- Non funziona se cambia più di un bit ma la probabilità di due errori nella trasmissione di un singolo byte è bassa.

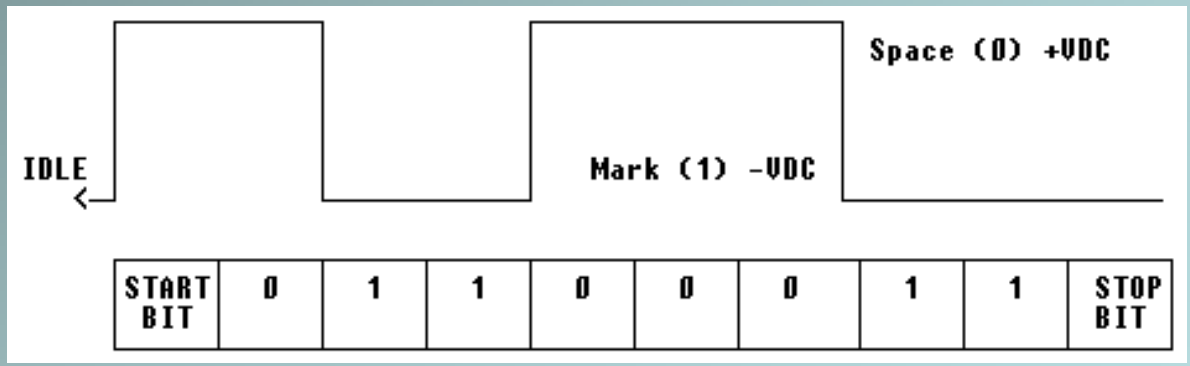
Sincronizzazione

- Sia nella trasmissione seriale sia in quella parallela, spesso viene trasmesso un segnale di sincronizzazione su un canale dedicato (clock).
- Il clock consente al ricevitore di decodificare i segnali nello stesso tempo in cui vengono inviati. Il ricevitore ed il trasmettitore sono, così, sincronizzati nel tempo di clock.
- La comunicazione, in questo, caso si dice **sincrona.**
- La frequenza del clock definisce anche la velocità della comunicazione che si misura in bit x s=**bps** o **baud** (in realtà baud indica la frequenza con cui la linea cambia stato che non è necessariamente pari a bps).
- Il vantaggio di questo tipo di trasmissione è che qualsiasi drift nella frequenza di trasmissione è riconosciuto dal ricevitore. Sono, inoltre, possibili frequenze molto alte.



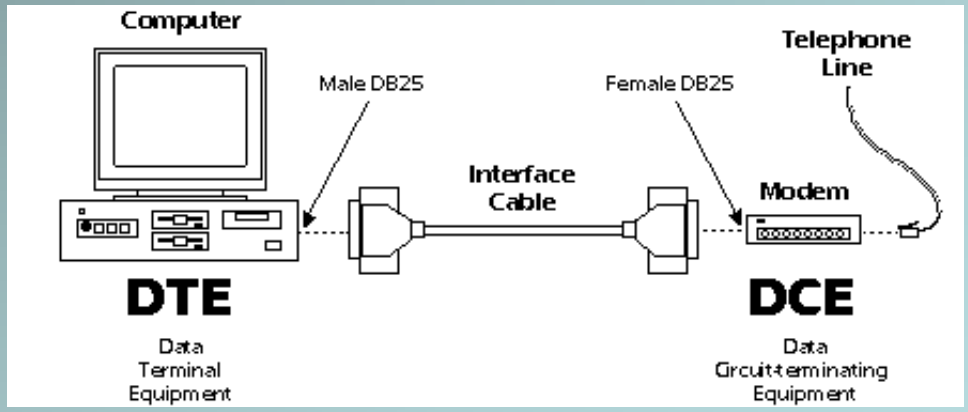
Comunicazione asincrona

- Nella comunicazione asincrona, non viene trasmesso alcun segnale di clock.
- Ai dati vengono aggiunti in testa ed in coda una serie di bit, detti di start e stop, che istruiscono il ricevitore dell'inizio e della fine di una parola.
- Il vantaggio è che trasmettitore e ricevitore hanno clock indipendenti e le parole possono essere separate da un qualsiasi intervallo di tempo.



Linee seriali RS232

- RS-232 è uno standard per le comunicazioni seriali definito dalla Electronic Industries Association (EIA). RS sta per Recommended Standard.
- Esiste in tre diverse versioni A, B o C che definiscono i differenti voltaggi per i livelli *on* ed *off.*
- La versione più utilizzata è RS-232C che definisce un bit *on* (mark) come una tensione tra -3V e -12V e *off* (space) tra +3V e +12V.
- Lo standard definisce anche che la massima distanza lungo la quale questi segnali possono essere trasmessi è 8m (25 ft).
- Le porte o i dispositivi seriali sono etichettati come Data Communication Equipment (DCE) o Data Terminal Equipment (DTE). I segnali di trasmissione e ricezione fra queste due tipologie sono invertiti di posto.



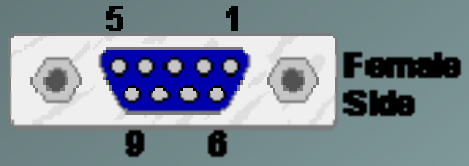
RS232 – schema del connettore

- Oltre ai cavi per la trasmissione e la ricezione dei dati, RS232 necessita di altri cavi per il controllo del flusso di dati.
- Il connettore utilizzato è un connettore a “D” a 25 poli su cui coesistono un canale primario ed uno secondario.
- Il segnale FG non è quasi mai utilizzato,per cui spesso è sufficiente un connettore a 9 poli.
- Eliminando il segnale DSR si può usare anche un connettore a 8 poli come quelli in uso per il networking (RJ45)

DB-25 Pin#	DB-9 Pin#	Nome comune	Direzione DTE-DCE	Nome formale
1		FG	-	Frame Ground
2	3	TD	→	Transmitted Data, TxD
3	2	RD	←	Received Data, RxD
4	7	RTS	→	Request to Send
5	8	CTS	←	Clear To Send
6	6	DSR	←	Data Set Ready
7	5	SG	-	Signal Ground, GND
8	1	DCD	←	Data Carrier Detect
9		--	-	+P
10		--	-	-P
11		--	-	Non assegnato
12		SDCD	←	Secondary Data Carrier Detect
13		SCTS	←	Secondary Clear To Send
14		STD	→	Secondary Transmitted Data
15		TC	←	Transmission Signal Element Timing
16		SRD	←	Secondary Received Data
17		RC	→	Receiver Signal Element Timing
18		--	-	Non assegnato
19		SRTS	→	Secondary Request To Send
20	4	DTR	→	Data Terminal Ready
21		SQ	←	Signal Quality detector
22	9	RI	←	Ring Indicator
23		--	→	Data Signal Rate Selector
24		--	←	Transmitter Signal Element Timing
25		--	-	Non assegnato

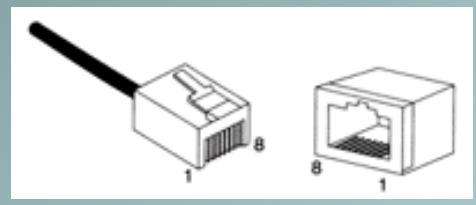
RS232 Connettori ridotti

DB9



1	DCD
2	RXD
3	TXD
4	DTR
5	GND
6	DSR
7	RTS
8	CTS
9	RI

RJ45



1	RI
2	DCD
3	DTR
4	GND
5	RXD
6	TXD
7	CTS
8	RTS

RS232 Segnali di controllo I

- **GND - Logic Ground:** Riferimento di tensione per tutti gli altri segnali.
- **TXD - Transmitted Data:** Linea di trasmissione dei bit di informazione dal DTE (periferica) a DCE (computer). Il DTE mantiene tale linea al valore logico 1 quando *non* ci sono dati da trasmettere; la trasmissione del dato su questa linea è possibile solo se i segnali *Request To Send*, *Clear To Send*, *Data Set Ready* e *Data Terminal Ready*, quando presenti, assumono valore logico 0.
- **RXD - Received Data:** Linea di trasmissione dei bit di informazione dal DCE (computer) a DTE (periferica).Il dato (bit) primario viene inviato su questa linea dal DCE al DTE. Questo segnale viene mantenuto ad un valore logico 1 quando DCE non trasmette dati e viene portato a 0 per un breve intervallo di tempo dopo una transizione della linea *Request To Send* da 1 a 0, per consentire il completamento della trasmissione.
- **DCD - Data Carrier Detect:** Il segnale DCD indica che il computer o il dispositivo sono connessi o on line. DCD non è sempre usato o disponibile.
- **DSR – Data set Ready:** Su questa linea il DCE dice al DTE che il canale di comunicazione è disponibile, ma non indica che effettivamente sia stato stabilito un link con un dispositivo remoto.

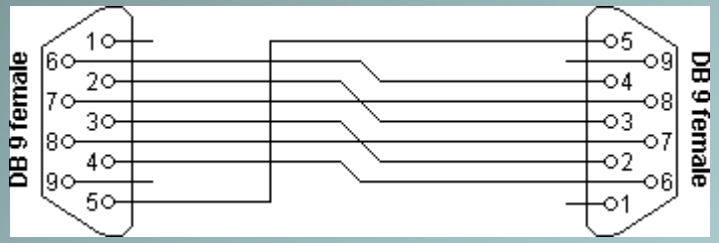
RS232 Segnali di controllo II

- **DTR - Data Terminal Ready:** Se questo segnale è a livello logico 1, DCE viene informato che DTE è pronto per la ricezione. Il segnale DTR deve essere attivo prima che DCE attivi il segnale Data Set Ready, indicando così di essere connesso al canale di comunicazione. Se il segnale DTR assume il valore logico 0, DCE interrompe la trasmissione in corso.
- **CTS - Clear To Send:** Segnale di risposta a DTE. Quando attivo, indica a DTE che può dare inizio alla trasmissione (linea TXD)
- **RTS - Request To Send:** dal DTE al DCE, disponibilità a trasmettere; quando attivo questo segnale informa il DCE che il DTE è pronto a spedire un byte.
- **Transmitter Signal Element Timing:** Linea usata da DTE per inviare a DCE un segnale di clock. La transizione da 1 a 0 indica il punto centrale del tratto di segnale corrispondente ad un bit sul *Transmitted Data.*
- **Receiver Signal Element Timing:** Segnale di clock inviato da DCE a DTE in modo che DTE sia in grado di sincronizzare il proprio circuito di ricezione che pilota la linea Received Data. La frequenza del segnale di clock dipende dal bit-rate della trasmissione sulla linea Received Data. La transizione da 1 a 0 indica il punto centrale del tratto di segnale corrispondente ad un bit sulla Received Data.

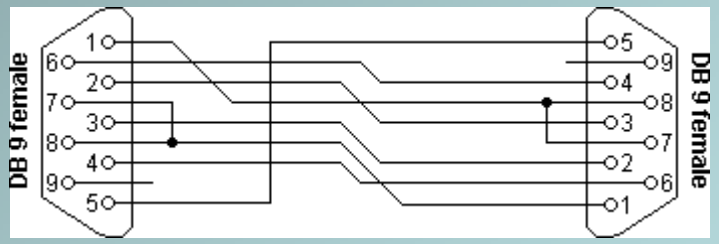
Collegare fra loro due DTE

- Serve un collegamento con trasmissione e ricezione incrociate (null modem).
- Bisogna tener conto anche dei segnali di controllo (handshaking) o collegare mutuamente DTR con DSR e RTS con CTS o tra loro gli RTS e CTS di ciascuna porta ed entrambi al DCD dell'altra
- Alternativamente si possono mettere "a loop" i segnali di handshaking su entrambe le porte, ma a questo punto il controllo di flusso non potrà più essere hardware.

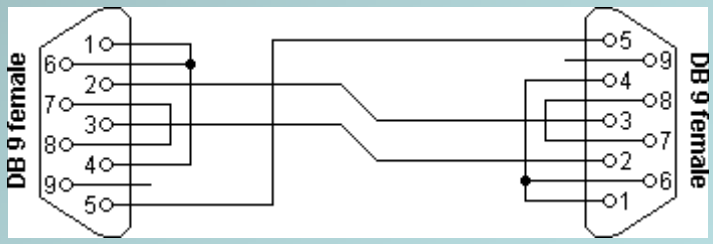
Full Handshaking



partial Handshaking



Loop Handshaking



Controllo di flusso software

- Per la trasmissione asincrona su RS232 è possibile non fare uso dei segnali di controllo per il controllo di flusso.
- A tale scopo vengono trasmessi byte particolari di start e stop. Questi sono definiti dal codice ASCII (American Standard Code for Information Exchange) come XON (binario:10001) e XOFF (10011)
- Questo metodo è utile quando si trasferiscono dati in formato testo (codice ASCII), meno quando si trasferiscono altri formati.
- Il controllo di flusso software è più lento di quello hardware (handshaking).

Linee seriali RS422

- La linea seriale RS232 utilizza un “alto” voltaggio e una singola polarità. Questo la rende sensibile ai disturbi elettromagnetici (EMI) e all’attenuazione.
- L’alternativa è usare una linea *differenziale*, dove, cioè, il segnale si propaga lungo una coppia di cavi con polarità opposte. Questo, sia per i dati, sia per i segnali di controllo.
- Lo standard RS422 prevede che ciascuna linea differenziale sia pilotata da un singolo driver e permette la trasmissione fino a 10 Mbps su distanze fino a 4000ft (1200m) fra un trasmettitore e fino a 10 ricevitori (multipoint).
- All'uscita del trasmettitore la differenza di potenziale tra le linee A e B deve essere di almeno 4 V e la tensione di modo comune deve essere minore di 7 V (normalmente una linea vale 0 V e l'altra circa 5 V). Il ricevitore deve essere in grado di interpretare correttamente lo stato della linea quando la differenza di potenziale è superiore in modulo a 200 mV.

RS422 Pinout

Figure 3 - RS-422 Connector

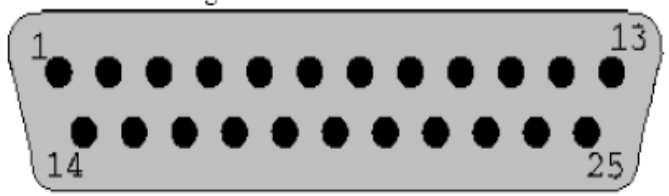


Table 13 - RS-422 Signals

Pin	Description	Pin	Description
1	Earth Ground	14	TXD+
2	**TXD**- - Transmitted Data	15	Transmit Clock-
3	**RXD**- - Received Data	16	RXD+
4	**RTS**- - Request To Send	17	Receiver Clock-
5	**CTS**- - Clear To Send	18	Unassigned
6	**DSR** - Data Set Ready	19	RTS+
7	**GND** - Logic Ground	20	**DTR**- - Data Terminal Ready
8	**DCD**- - Data Carrier Detect	21	Signal Quality Detect
9	Reserved	22	Unassigned
10	Reserved	23	**DTR**+
11	Unassigned	24	Transmit Clock+
12	DCD+	25	Receiver Clock+
13	CTS+		

RS485

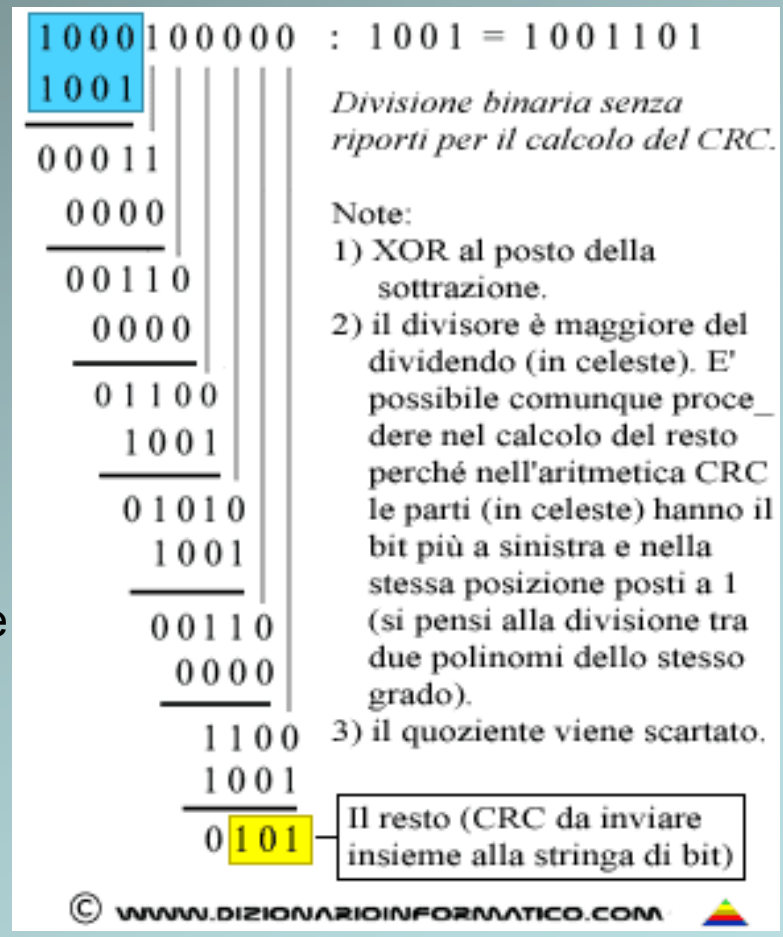
- RS485 è simile a RS422. La differenza sostanziale è il supporto delle linee multi-drop, cioè linee che presentano la coesistenza di più ricevitori e trasmettitori sulla stessa coppia di fili. Al fine di evitare conflitti è ovviamente necessario che un solo trasmettitore alla volta sia attivo. Questo implica l'uso di trasmettitori che, oltre alle uscite corrispondenti allo zero e all'uno, possano gestire anche un "terzo stato" in cui l'elettronica appare come fisicamente non collegata alla linea (stato detto ad alta impedenza o Hi-Z).
- Sia RS422 sia RS485 hanno bisogno di terminare la linea per evitare riflessioni del segnale sul cavo.

Comunicazione sincrona su linea seriale

- La comunicazione sincrona su una linea seriale appare come un flusso (stream) costante di bit.
- Il computer deve fornire o ricevere un segnale di clock per la sincronizzazione del trasmettitore con il ricevitore.
- Anche con questa sincronizzazione l'inizio e la fine dei dati devono essere identificati. A tale scopo si utilizzano protocolli a "pacchetto" di dati, in cui il dato è "impacchettato" ed ogni pacchetto è preceduto e seguito da sequenze note di bit. Esempi: High-speed Data Link Control (HDLC), o Serial Data Link Control (SDLC).
- Per mantenere la sincronizzazione, deve essere stabilita anche una sequenza di bit in assenza di dati.
- Siccome ogni pacchetto può contenere più di un carattere, le sequenze di sincronizzazione non sono ad ogni carattere e di conseguenza si ha un incremento fino al 25% rispetto ala comunicazione asincrona.
- In genere, data la necessità di HW e SW aggiuntivo, non è disponibile su RS232. Lo è invece su RS422/485.

Rivelazione degli errori II - CRC

- In trasmissioni seriali a pacchetto, oltre al bit di parità ci sono altri metodi per controllare gli errori di trasmissione.
- Il numero massimo di bit che un sistema è in grado di individuare con il 100% di probabilità, si chiama *distanza di Hamming*. La parità ha distanza di Hamming = 1.
- CRC (Cyclic Redundancy Check): è una parola (per es. 16 bit) risultante da un'operazione sui bit del pacchetto, il cui risultato è ciclico rispetto a uno shift di n bit. Si tratta in genere di calcolare il resto della divisione modulo 2 per polinomi del tipo g(X)=X16+X12+X5+1 (CCITT), ove XN è la posizione del bit del polinomio il cui valore è 1. g(x) è il polinomio generatore. Nell'esempio a fronte g(x) = 1001 =X3+1 e la parola da testare è 1000100.
- Se la CRC calcolata nel pacchetto ricevuto è uguale a quella ricevuta, non ci sono errori.
- Si raggiungono distanze di Hamming di 4 o 6.



Divisione binaria senza riporti per il calcolo del CRC.

Note:
1) XOR al posto della sottrazione.
2) il divisore è maggiore del dividendo (in celeste). E' possibile comunque proce_ dere nel calcolo del resto perché nell'aritmetica CRC le parti (in celeste) hanno il bit più a sinistra e nella stessa posizione posti a 1 (si pensi alla divisione tra due polinomi dello stesso grado).
3) il quoziente viene scartato.

UART

- UART (**U**niversal **A**synchronous **R**eceiver **T**ransmitter) - ricevitore/trasmettitore asincrono universale - sono chip seriali. Scopo della UART è convertire i byte dal bus parallelo del PC in un flusso seriale di bit e viceversa nel senso inverso.
- Le UART trattano dati divisi in pezzi della dimensione di un byte, che per convenienza è anche la dimensione dei caratteri ASCII
- Ci sono due tipi principali di UART: UART "stupide" (dumb, per es 8250) e UART FIFO (per es. 16550A), . La differenza tra dumb e FIFO (First In, First Out - Il primo che entra è il primo che esce) è che le seconde sono dotate di un buffer (FIFO, per l'appunto) di >16 byte. Nelle prime ad ogni carattere ricevuto veniva generato un interrupt, nelle seconde, l'interrupt viene generato solo a buffer pieno (o quasi pieno – parametro "Trigger level" (TL) aggiustabile), e, a quel punto viene svuotato il buffer.
- Se ci sono meno di TL byte nel buffer, un interrupt viene comunque generato se non arrivano altri caratteri entro un tempo di "timeout".
- Si veda per esempio: http://www.lammertbies.nl/comm/info/serial-uart.html

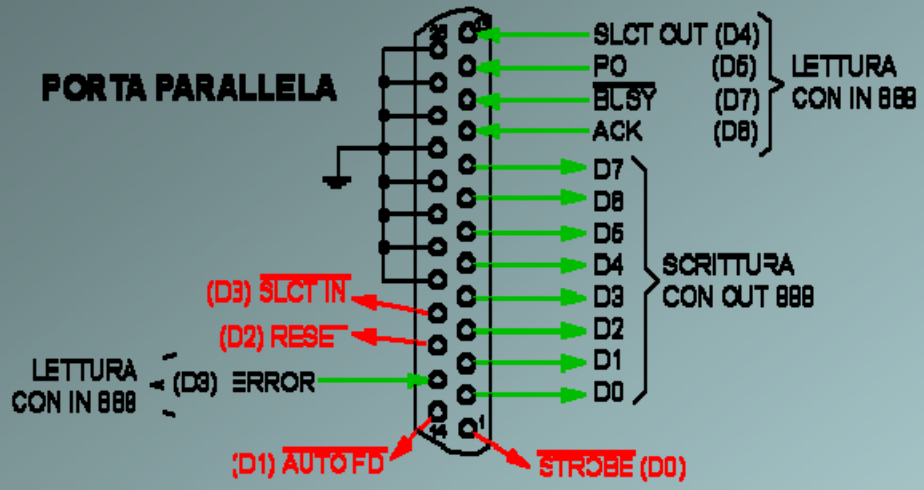
Porta parallela

- Sui PC oltre alla porta seriale è spesso presente la porta parallela.
- È basata su un connettore a 25 poli (DB25) o 36 poli e serve a connettere dispositivi con una maggiore richiesta di banda rispetto ai dispositivi seriali (p.es. Scanner, stampanti)
- È spesso chiamata interfaccia Centronics, dalla compagnia che l'ha ideata.
- Attraverso la porta seriale è possibile inviare dati in parallelo a velocità fino a 150 kbit/s.
- Per ricevere dati bisogna cambiare modalità alla porta e metterla in *nibble mode* che le permette di ricevere 4 bit in parallelo nella direzione inversa.
- http://www.pklab.net/kbase/infind/lpt/

Porta parallela - pinout

Pin No (D-Type 25)	Pin No (Centronics)	SPP Signal	Direction In/out	Register	Hardware Inverted
1	1	nStrobe	In/Out	Control	Yes
2	2	Data 0	Out	Data	
3	3	Data 1	Out	Data	
4	4	Data 2	Out	Data	
5	5	Data 3	Out	Data	
6	6	Data 4	Out	Data	
7	7	Data 5	Out	Data	
8	8	Data 6	Out	Data	
9	9	Data 7	Out	Data	
10	10	nAck	In	Status	
11	11	Busy	In	Status	Yes
12	12	Paper-Out PaperEnd	In	Status	
13	13	Select	In	Status	
14	14	nAuto-Linefeed	In/Out	Control	Yes
15	32	nError / nFault	In	Status	
16	31	nInitialize	In/Out	Control	
17	36	nSelect-Printer nSelect-In	In/Out	Control	Yes
18 - 25	19-30	Ground	Gnd		

Porta parallela – Pinout II

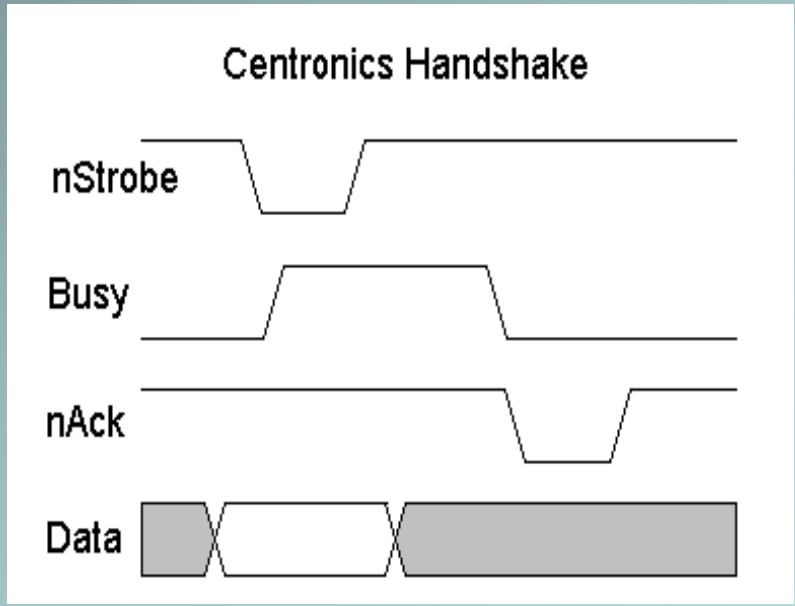


25 poli

36 poli femmina e maschio

Porta parallela - Handshake

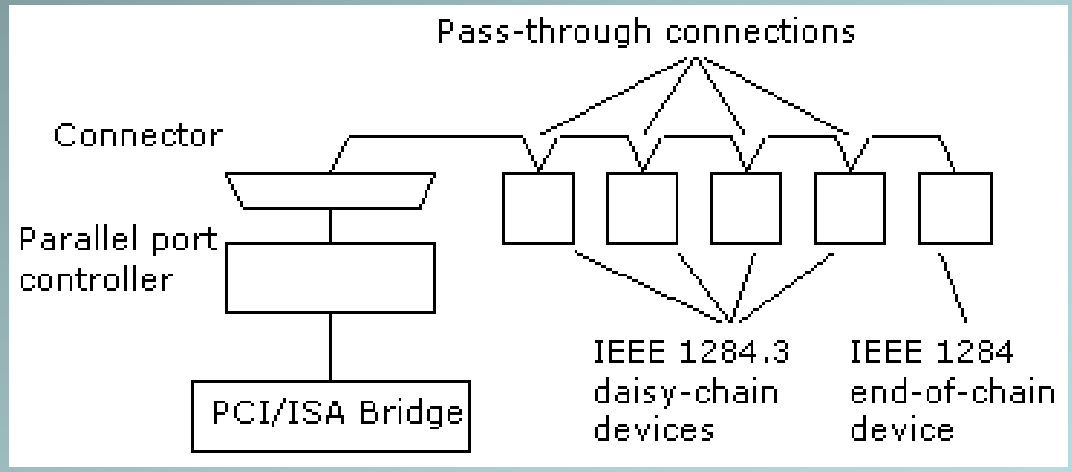
- Per scrivere un byte su una stampante (o qualunque cosa sia) il software deve:
  - Scrivere il dato sulla porta dati
  - Controllare che la stampante non sia occupata, nel qual caso non acetterà dati
  - Mandare basso lo strobe (pin1) per comunicare alla stampante che ci sono dati sulle linee dati (linee 2-9)
  - Rimandare alto lo strobe dopo aver aspettato circa 5 µs con lo strobe basso



Porta parallela – daisy chain

•Vedi: http://people.redhat.com/twaugh/parport/html/parportguide.html

- Con la porta parallela è possibile connettere più dispositivi in serie, secondo uno standard definito dalla IEEE (IEEE1284.3).
- All'inizializzazione del "bus" ai dispositivi viene assegnato un'indirizzo a partire da 0.
- Ci possono essere fino a 4 dispositivi in catena più uno alla fine della catena che non risente del daisy chain e pensa di essere connesso direttamente al computer



Un esempio di programmazione di porta parallela

```
#include <stdio.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <asm/io.h>
char *binprint( unsigned char x, char *buf )
{
  int i;
  for( i=0; i<8; i++ )
     buf[7-i]=(x&(1<<i))?'1':'0'; buf[8]=0;
   return buf;
}
int main( int argc, char *argv[] )
 {
  char c;
  unsigned
  char val;
  char buf[9];
  int x;
  if( argc<2 )
    {
      printf(" example usage: parcon 1l 2l 3h 5h
8l\n");
      return 2;
    }
```

```
if(ioperm(888,1,1) )//permesso di write
    {
      printf("Couldn't get port 888\n");
      return 1;
    }
 val = inb(888);
 printf("old = %s\n",binprint(val,buf));


for( x=1; x<argc; x++ )
  {
   if( argv[x][1]!='h' )
     val &= ~(1<<(argv[x][0]-'1'));
   else
     val |= 1<<(argv[x][0]-'1');
  }
  printf("new = %s\n",binprint(val,buf));
  outb(val,888);
  return 0;
}
```

Specifico Linux…