I Manuali di Giobe2000

PORTA SERIALE

Studio Tecnico ing. Giorgio OBER eurosito@giobe2000.it

La Monografia sulla **PORTA SERIALE** può differire in parte dalla versione on-line
soggetta a costanti aggiornamenti e integrazioni
Verifica le eventuali novità direttamente sul Sito

Porta Seriale — Presentazione

- La ***porta seriale*** è un importante canale di comunicazione del tuo computer, capace tanto di ***ricevere*** dati in ingresso quanto di ***trasmettere*** dati in uscita; in questo assomiglia alle ***porte parallele*** di seconda generazione (EPP/ECP) anche se gli *obiettivi* e il loro *modo di agire* sono effettivamente molto diversi.
- Il paragone tra i 2 tipi di *porta* è inevitabile e utile, per cui in seguito ne confronteremo molto frequentemente le caratteristiche.
- Fisicamente l'oggetto che rappresenta la ***porta seriale*** è il suo **connettore**, posto nella parte posteriore del computer; un tempo il suo aspetto mostrava ***25 spinotti*** (***pin***) ***dorati*** (cioè è di tipo **maschio**) disposti su 2 file, l'una, di ***13***, sfalsata rispetto all'altra, di ***12***, in stretto accordo con le specifiche imposte dallo ***standard RS232 originale***; se non si prestava attenzione, era facilmente confondibile con quello della *porta parallela*, della stessa dimensione ma con ***25 forellini*** (cioè è di tipo **femmina**) al posto dei ***pin***.
- Di fatto ***non tutti*** gli *spinotti* di questo **connettore** erano ***portatori di informazione***; solo ***10*** dei ***25 pin*** erano *utili* alla comunicazione seriale; i rimanenti **15** pin risultavano addirittura inutilizzati, cioè non collegati ai dispositivi dell'interfaccia interna.
- Per questa ragione (ma anche per owi motivi di convenienza) i costruttori hanno pensato di ridurne la dimensione a soli ***9 pin*** (***5*** su una fila e ***4*** sull'altra): non potendo proporre una struttura con numero pari di pin risulta chiaro che una delle ***10*** informazioni è stata ritenuta non indispensabile; maggiori dettagli su questa scelta saranno discussi nella parte dedicata ai connettori, accessibile come le altre con l'aiuto delle icone della *barra interattiva* marrone a sinistra.
- Sta di fatto che il **connettore DB25** a ***25 pin*** (**maschio**) è sparito dalla parte posteriore del tuo computer per lasciare il posto a 2 **connettori** più piccoli, entrambi di tipo **maschio** a ***9 pin*** (**DB9**) destinati ad altrettante porte seriali.
- La capacità della ***porta seriale*** di consentire *intrinsecamente* il *transito dei dati* **in entrambe le direzioni** ne ha fatto il punto di riferimento per collegamenti con periferiche in grado di interagire con il computer, come *Mouse* o altri *dispositivi di puntamento* (*Track Ball*, *Touchpad*, *Digitizer*, ...), *Lettori di Codice a Barre*, *Scanner*, *Macchine fotografiche digitali*, ..., ma soprattutto *Modem telefonici*, *Reti* e, naturalmente, *un altro computer*, per esempio nel trasferimento di files da una macchina all'altra (gestito da applicativi come Laplink e Interlink).
- Ma trova applicazioni anche nei collegamenti sostanzialmente unidirezionali tipici della prima *porta parallela*, **SPP**, dal computer alla periferica, di solito al servizio di *Stampanti* (owiamente con interfaccia seriale) o di *Plotter*.

Come Funziona... — Porta Seriale

- Come Funziona... - La Comunicazione Seriale: Premesse

1.1 Il Futuro delle Comunicazioni Seriali

- Nonostante la sua enorme diffusione e la sua grande popolarità la ***porta seriale*** (e lo standard ***RS232C***) ***sta per sparire*** dai nostri computer: oggi (*2004*) comincia a sentire gli acciacchi dell'età.
- Il suo più grande limite è la **velocità**, obiettivamente **molto bassa** rispetto a quella dei moderni processori; l'inarrestabile frenesia dei nostri giorni e l'effettiva necessità di collegare al computer dispositivi esterni sempre più veloci, come memorie di massa (*Hard Disk esterni* o memoria RAM vista come un disco virtuale, le cosiddette *penne di memoria*) o *sistemi multimediali* chiamati a gestire suoni ed immagini video in tempo reale, ha spalancato le porte all'**USB** (***Universal Serial Bus***).
- Le ***porte USB*** hanno tutto per mantenere a lungo il successo e per diventare il nuovo *standard* per la *comunicazione seriale*:
  - sono versatili: più che semplici porte sono un vero e proprio *bus*, in grado di collegare al PC fino a 127 apparecchiature seriali, contro l'unica gestita dalle porte RS-232C
  - minimizzano i problemi di collegamento (connettori e cavi) e hanno interfacce a basso costo
  - soprattutto, hanno *velocità di trasferimento* fino a 12 megabit per secondo (più di 100 volte quella massima delle porte RS-232C), nella versione USB **1.0**
- L'unico *difetto* è, per il *gran mondo*, *irrilevante*: pochi (noi tra quelli...) si danneranno per la difficoltà di comprendere a fondo i suoi protocolli e per l'attuale assoluta mancanza di punti di riferimento, atti a consentirne la programmazione *a basso livello*; ma verranno tempi migliori!

- Naturalmente è dimostrato che, nel mondo dei computer, non ci sono punti di arrivo definitivi: sono già disponibili standard seriali ancora più affidabili e potenti, come il ***Fire-wire***, noto anche come ***IEEE 1394*** o ***P1394***, in grado di garantire velocità di trasferimento dati fino a 400 megabit per secondo, comparabile con le prestazioni della versione più recente (*2000*) della ***porta USB*** (USB **2.0** (fino a 480 megabit per secondo).
- Nessuno di questi canali di *comunicazione* verrà approfondito, lasciando aperto il discorso per future recensioni ad essi dedicate; da questo momento in poi ci concentreremo esclusivamente sulla ***porta seriale*** alias **RS232C**.

Se lo desideri puoi accedere all'***elenco strutturato degli argomenti*** di questa monografia cliccando in ogni momento sull'*icona* ***indice***, in basso a destra nella barra interattiva marrone.

1 - Come Funziona... - La Comunicazione Seriale: Premesse

1.2 Le Regole dalle Comunicazioni Seriali

- La **comunicazione dei dati** può essere intesa in molti modi (*parallela*, *seriale sincrona*, *seriale asincrona*,..) e supportata praticamente in tanti altri; ciascun produttore di computer e di periferiche potrebbe suggerire un proprio metodo, mettendo nel panico gli utenti finali, ... cioè noi.

Per tentare di dare ordine a questa possibile giungla *numerose* ***organizzazioni internazionali*** si sono fatte carico di studiare e proporre delle **regole**, riconosciute e condivise da tutti come **Standard**, in grado di specificare ogni dettaglio.

- Tra le organizzazioni che si occupano di *comunicazione* sono importanti:
  - Institute of Electrical and Electronics Engineers (**IEEE**): gli standard prodotti da questo Ente sono orientati verso numerosi obiettivi di ingegneria professionale e, di solito, sono numerati, al fine di sintetizzare l'oggetto delle nuove regole; un esempio nell'ambito seriale è lo standard **IEEE-1394**, pubblicato nel settembre del 1986 e messo a punto negli anni successivi: un bus seriale ad alte prestazioni, detto ***Fire-wire***, in grado di coprire distanze di 4 o 5 metri a velocità fino a 400 Mb/s (e, in prospettiva, fino a qualche Gb/s)
  - Consultative Committee on International Telephone and Telegraph (**CCITT**): questo comitato Internazionale si occupa specificatamente di telecomunicazioni e in particolare aiuta a qualificare l'uso dei modem (e della linea telefonica).
- Nel campo della ***comunicazione seriale*** lo standard più popolare è certamente quello noto come **RS232**, sviluppato dalla Electronic Industry Association (**EIA**) e dalla Telecommunications Industry Association (**TIA**), per questo talvolta detto anche **EIA232** o **TIA232**; è significativo sottolineare come proprio una associazione di industriali (statunitensi) si sia accollata questa importante responsabilità. Da notare che il prefisso letterale **RS** sta per ***R**ecommended **S**tandard*.
- Lo standard **RS232** fu presentato nel 1962 per stabilire i ***protocolli*** *software e hardware di ricetrasmissione* tra apparecchiature digitali collegate serialmente tra loro, direttamente o (con l'intercessione di **modem**) tramite linea telefonica.

Va evidenziato che lo ***standard* RS232** è apparso più di vent'anni **prima** della ***porta seriale***, introdotta dall'IBM nel 1984 come **porta di *com*unicazione asincrona dati** (sinteticamente **COM**) dei suoi PC; ciò nonostante esso è diventato il suo ***standard*** *di riferimento*, tanto che la *porta seriale* è spesso identificata come ***porta RS232***.

- Lo standard **RS232** ha subito nel tempo 4 aggiornamenti, volti a migliorare e rendere attuali le necessità delle applicazioni di *comunicazione seriale*. La versione coinvolta è facilmente riconoscibile dalla lettera aggiunta in coda al numero:
  - la sigla RS232**C** indica la *terza revisione* (1969) dello standard **EIA** e corrisponde alle *raccomandazioni* dello standard **CCITT V.28/V.24** e a quelle dello standard **ISO IS2110**
  - la *quarta revisione* (RS232**D** del 1987) dello standard **EIA** è più diffusa con la sigla EIA-232**D**

2 - Come Funziona... - L'Interfaccia (porta) seriale

2.1 Compiti dell'Interfaccia Seriale

- Dalle considerazioni proposte nelle pagine precedenti appare evidente il *compito principale* di una ***porta seriale*** e dei circuiti che la supportano: **mettere in fila** (*in **serie***) **le informazioni**; l'*operazione* è assolutamente *innaturale* per un processore, abituato invece a trattare i suoi dati *in **parallelo***, in quantità comunque multiple di 8, sulle **numerose linee** del suo *bus Dati*.

- Da questo punto di vista la *porta parallela* è certamente più in sintonia, offrendo sul suo connettore la medesima informazione trattata dal processore; la **grande novità** garantita dalla *comunicazione seriale* è dunque la possibilità di trasferire un byte (= 8bit) **su un solo filo**, **invece degli 8** richiesti da *quella parallela*, al costo di una maggiore complessità circuitale.
- Un primo confronto sui 2 tipi di dato ci suggerisce queste considerazioni:
  - nel ***dato parallelo*** ciascun bit ha importanza per la *posizione* (***peso***) che occupa nell'ambito del byte, cioè assume importanza diversa in funzione del filo (degli 8) su cui è trasmesso
  - nel ***dato seriale*** l'importanza di ciascun bit è una *questione di* ***tempo***, cioè di chi è trasmesso (o ricevuto) per primo, di solito il *meno significativo*, e di chi è trasmesso (o ricevuto) per ultimo

Parlare di ***porta seriale*** è dunque riduttivo: è più corretto dire ***interfaccia seriale***, anticipando i gravosi compiti di manipolazione che i suoi circuiti saranno chiamati a compiere sui segnali disponibili sul connettore.

- L'***interfaccia seriale*** deve assolvere a **compiti importanti**:
  - è chiamata ad occuparsi della **conversione** di ciascun dato da trasmettere, fornito in ***parallelo*** dal processore, in un **flusso *seriale*** di bit da porre sulla linea di trasmissione; oppure dell'operazione opposta, **convertendo** in ***parallelo***, per il processore, il **flusso *seriale*** di bit ricevuto dalla linea di ricezione
  - deve occuparsi della **formattazione** del dato seriale (di tipo *asincrono*), aggiungendo *bit di inizio e fine dato* ed eventuali *bit di controllo*, adatti a rilevare eventuali *errori di ricetrasmissione*
  - deve organizzare le temporizzazioni necessarie per sincronizzare la *collocazione* di ciascun bit sulla linea dati, durante la trasmissione, e la loro *lettura*, durante la ricezione
  - in aggiunta alle *linee di dato* deve mettere a disposizione altre *linee*, previste dallo standard **RS232** *per il controllo del flusso* dati (*hardware handshaking*)
- Fin dalla sua prima comparsa sul *PC IBM originale* l'***interfaccia seriale*** ha affidato tutti questi **compiti** ad un componente elettronico appositamente inventato da *National Semiconductor*, l'**U**niversal **A**synchronous **R**eceiver/**T**ransmitter, sinteticamente noto come **UART** e distribuito con la sigla **INS8250 *ACE*** (**A**synchronous **C**ommunications ***E***lement).
- A questo *storico* componente ne sono seguiti numerosi altri, nell'intento di ottimizzarne il lavoro e di renderlo sempre più veloce e affidabile: molte pagine della recensione si occuperanno profondamente di ogni dettaglio, aprendo la strada alla *programmazione a basso livello* della ***porta seriale***.
- In aggiunta l'***interfaccia seriale*** deve garantire le **specifiche** *elettriche*, *meccaniche* e *logiche* previste dallo standard **RS232**; in particolare, per trasformare i livelli di tensione TTL (di norma da 0 a +5 volt) assicurati dall'**UART** in quelli previsti, compresi nell'intervallo da ±3 volt a ±25 volt, si rende necessaria la presenza di dispositivi adatti allo scopo, detti ***Convertitori del Livello RS-232***.
- Essi pure saranno trattati in altra parte della recensione; ora è importante far emergere le caratteristiche di questo importante *canale di comunicazione*.

2 - Come Funziona... - L'Interfaccia (porta) seriale

2.2 Costi e Distanze a confronto con l'Interfaccia Parallela

- Un **primo effetto** della possibilità di scambiare dati in *forma* ***seriale*** si riflette sul **costo** dei *cavi di collegamento*:
  - è facile intuire che bastano 3 fili: uno per la *ricezione*, uno per la *trasmissione* e uno per il riferimento di *massa*, sempre indispensabile nel collegamento tra qualunque apparecchiatura
  - inutile sottolineare che, con la *porta parallela*, solo per gli 8 bit del dato serviranno altrettanti conduttori
- Per onor del vero sia la *comunicazione seriale* che *quella parallela* può essere gestita in modo più articolato, coinvolgendo in entrambi i casi un numero maggiore di fili, necessari per i *segnali di controllo* (detti di *handshaking* e discussi in dettaglio in altra parte); ma questo non intacca l'affermazione precedente: per un collegamento minimo (detto, come vedremo, *Null Modem*) bastano 3 fili.
- Più difficile è capire come sia possibile dare significato ad una sequenza continua di bit e ricostruire da essa le informazioni a 8 bit; ma questa è la *magica valenza* dell'**UART** e, per saperne di più, dovremo pazientare ancora un po'.
- La ***porta seriale*** è nata per garantire lo *scambio* di informazione binaria a **distanze più lunghe** di quelle assicurate dalla *porta parallela*:
  - la cosa è assolutamente fuori discussione se, come spesso succede, le *porte* sono collegate a ***modem*** (o a ***ricetrasmittenti radio***) da entrambe le *terminazioni* del collegamento: poichè in questo caso il supporto dell'informazione è la *linea telefonica* (o l'*etere*) non c'è limite alla distanza raggiungibile
  - ma anche nel normale collegamento ***via cavo*** si possono raggiungere distanze almeno 4 o 5 volte più grandi di quella coperta da una *buona comunicazione* via *porta parallela*; lo standard originale **RS232** assicura ricetrasmissione affidabile fino a **15 metri**, ma in accordo con le più recenti direttive **RS–232D**, con cavi di buona qualità o con velocità di ricetrasmissione lente, si possono raggiungere i **25 metri**.

2 - Come Funziona... - L'Interfaccia (porta) seriale

2.3 Velocità a confronto con l'Interfaccia Parallela

- Per contro la **velocità** di una ***porta*** (*interfaccia*) ***seriale*** è **certamente inferiore** a quella possibile con la *porta parallela*: basti pensare al fatto che quest'ultima mette in gioco 8 bit (un byte) *contemporaneamente* mentre la prima è costretta a porli sulla linea *in 8 tempi consecutivi*, uno dopo un altro.
- In questi termini anche le unità di misura con cui si valuta la velocità saranno diverse nelle 2 tecnologie:
  - la *porta parallela* opera sui ***bytes*** e misura la sua velocità in **B/s** (Bytes al secondo); a seconda del tipo può raggiungere **150 kB/s** (SPP), **2 MB/s** (EPP) o **2,4 MB/s** (ECP)
  - la *porta seriale* opera sui ***bit*** e misura la sua velocità in **bps** (bit per secondo); lo standard RS–232C definisce la sua velocità massima in **19.2 kb/s** (=**19200 bps**) ma a livello Hardware possono tipicamente raggiungere **115.2 kb/s**
- Il confronto, influenzato dalle 2 diverse tipologie, non è immediato: anche volendo ritenere che l'informazione seriale sia affidata ad 8 bit potremo trasformare i **115.2 kb/s** in **14.4 kB/s**, 10 volte più lenta della più lenta *parallela*..
- In realtà la gestione *asincrona* della *comunicazione seriale* rende ancora più sfavorevole il rapporto; come vedremo, per ogni byte di dato saranno necessari 11 (o 12 bit), portando i **115.2 kb/s** a soli **10.5 kB/s**, 10 (o **9.6 kB/s**).
- La **velocità** della *comunicazione seriale* è inoltre legata al conteso in cui si svolge; quella appena discussa è tipica del normale collegamento diretto ***via cavo*** tra 2 computer vicini e deve essere drasticamente ridotta all'aumentare della loro distanza; se il collegamento avviene tramite ***modem*** via *linea telefonica* la velocità più frequente è di **2400** o **1200 bps**.
- Lo standard **RS232** prevede un collegamento cablato, per esempio con ***fili di rame***, ma esiste anche la possibilità di affidare ad altro i segnali:
  - le ***porte IrDA*** (Infrared Device Application) utilizzano i ***segnali infrarossi***, molto simili a quelli usati dal nostro telecomando per controllare il televisore o l'apertura di una serranda; la necessità di supportare lo standard **RS232C** ne limita la *velocità massima* a quella sua tipica di **115200 bps**, sebbene potenzialmente questo supporto assicuri *velocità* fino a **4 Mb/s**, comunque su brevi distanze (2 metri, con unità di ricetrasmissione che si vedono tra loro)
  - tutti i discorsi di competitività della parallela sulla seriale *crollano miseramente* in ambiti seriali affidati alle ***fibre ottiche***, in grado di scambiare dati fino a **2,5 Gb/s**; con particolari tecnologie (come la **DWDM**, Dense Wavelength Division Multiplexer) ogni fibra può portare fino a 1000 canali di trasmissione, ciascuno in grado di trasmettere **10 Gb/s**: se pensiamo che ogni cavo può contenere molte centinaia di fibre ottiche la *capacità di trasmissione* risultante *fa venire il mal di testa* (milioni di miliardi di bit per secondo, **Tb/s**)
  - va detto che questa enorme *capacità di trasmissione* è vanificata dalla *velocità d'acceso* alla rete di distribuzione o delle reti locali come ***Ethernet***, limitata al massimo di **100 Mb/s** (con supporto elettrico) e di **1 Gb/s** (in fibra ottica)

3 - Come Funziona... - Principali Standard Seriali a confronto

3.1 Gli altri Standard delle Comunicazioni Seriali

- In aggiunta, le caratteristiche suggerite dallo standard **RS232** sono comunque *insufficienti* o *poco pratiche* negli ambienti di lavoro più sofisticati; i limiti più rilevanti sono associati a: distanza, velocità, sensibilità ai disturbi, range delle tensioni assunte dai segnali (e quindi necessità di disporre di appositi sistemi di alimentazione), quantità di concorrenti sulla linea. Tutto questo ha suggerito la necessità di definire **altri standard**; la tabella seguente offre un **confronto** superficiale, senza entrare nel merito:

Specifica	Standard			
	RS232C	RS422	RS423A	RS485
Distanza massima	25 m	1200 m [a 100kBs]	1200 m [a 1kBs]	1200 m [a 100kBs]
Velocità massima	20 kBs	10 MBs [max 10m]	100 kBs [max 10m]	10 MBs [max 10m]
linea	single-ended	differenziale	single-ended	differenziale
Tensioni di lavoro	±3V ÷ ±25V	±2V ÷ ±6V	±3,6V ÷ ±6V	±1,5V ÷ ±6V
Sensibilità Ricevitore	±3V	±200mV	±200mV	±200mV
Numero Trasmettitori	1	1	1	32
Numero Ricevitori	1	10	10	32

- Tutti i nuovi standard garantiscono *velocità decisamente interessanti* e consentono di coprire *distanze superiori al chilometro* (sebbene legaTe a diverse velocità); la linea di ricetrasmissione è elettricamente più complessa ma supporta segnali con tensioni massime decisamente inferiori e sensibilità di ricezione di soli ±200mV.

- Come Funziona... - La Comunicazione Seriale: Dettagli

4.1 Comunicazione Seriale: DTE e DCE

- Il collegamento diretto di 2 dispositivi dotati ***interfaccia seriale*** (o, sinteticamente, di ***porta seriale***) può essere impossibile se le distanze sono troppo grandi, a causa dell'attenuazione introdotta dai normali cavi di rame sui *segnali* ***digitali*** in transito lungo la linea.
- come abbiamo visto la *pur costosa* recente tecnologia delle ***fibre ottiche*** minimizza questo problema ma, al tempo (primi anni sessanta) in cui è stato concepito lo standard della *comunicazione seriale* **RS232**, si pensò di ricorrere ad un ***modem***, un dispositivo seriale in grado di acquisire il *segnale* ***digitale*** generato dal *computer* e di trasformarlo in uno adatto ad essere *portato* da una ***linea telefonica***.
- I *segnali* presenti su una linea telefonica sono di tipo ***analogico***, cioè un'onda elettrica proporzionale a quella sonora generata dalla nostra voce; per questo il modem in trasmissione affida i *valori logici* ad onde sinusoidali con la tecnica chiamata ***modulazione***; il modem in ricezione esegue l'operazione opposta rilevando i *valori logici* affidati alle onde sinusoidali con la tecnica opposta, detta ***demodulazione***.
- Questo prezioso dispositivo prende il nome dalla sintesi delle 2 operazioni da esso svolte, ***mod-dem*** o sinteticamente ***modem***.
- A causa dell'*anzianità di servizio* dello standard **RS232** anche la terminologia adottata per descrivere i dispositivi *previsti* alle 2 *estremità* del collegamento seriale è un po' *arcaica*:
  - il punto di partenza è detto **DTE** (**D**ata **T**erminal **E**quipment) a sostegno del fatto che, in passato, il compito era sostenuto da un *terminale* (una tastiera e un schermo) in grado di tradurre in *segnali seriali* la sequenza di caratteri digitati; oggi la parte del *terminale* è di solito assicurata da un *Personal Computer*, inteso talvolta come ***host*** o ***data terminal***
  - il punto d'arrivo è detto **DCE** (**D**ata **C**ommunication **E**quipment o, in modo più raro ma più vicino alla definizione originale, **D**ata **C**ircuit Terminating **E**quipment); oggi come allora può essere inteso come il dispositivo per accedere alla rete telefonica, un ***modem***, talvolta identificato come ***data set***

- Nella pratica comune succede di frequente che anche il punto d'arrivo sia un **DTE**: non di rado un *Personal Computer* è collegato via seriale ad un ***plotter*** o una ***stampante*** o ad un altro *Computer*; in questi casi molti dei segnali previsti dallo standard **RS232** (quelli espressamente pensati per l'interattività con il ***modem***) non sono necessari; si tratta dei cosiddetti collegamenti **Null Modem**, detti anche collegamenti **DTE-DTE**, per i quali sono previsti *cavi* con numero di fili ridotto al minimo (discussi nella sezione ad essi dedicata).
- (Quasi) tutti i nostri futuri progetti si appoggeranno a questa *filosofia*, rendendo marginale la *conoscenza* delle linee *per il controllo del flusso dati* (*hardware handshaking*) tra *Computer* e *modem*, per altro piuttosto *fastidiosa* e *scarsamente mnemonica* a causa della particolare paranoia con la quale sono definiti i segnali coinvolti, nel tentativo di evidenziare il ruolo da essi esercitato nella *comunicazione seriale* **DTE-DCE**.

- Come Funziona... - La Comunicazione Seriale: Dettagli

4.2 Comunicazione Seriale: Modalità simplex e duplex

- Per completare il quadro può essere utile citare le possibili modalità di comunicazione tra i 2 terminali **DTE-DCE** di un sistema di comunicazione seriale:
  - **simplex**: si tratta di una modalità usata raramente; i dati viaggiano in una sola direzione, come le automobili su una strada a senso unico, e non è quindi possibile ricevere dal partner la segnalazione di eventuali errori o un segnale di riconoscimento, alla fine della trasmissione. Un *canale televisivo* rende bene questa idea

- **half-duplex**: i dati viaggiano in entrambe le direzioni ma non nello stesso tempo, come le automobili su una strada a senso unico alternato; entrambi i terminali possono trasmettere e ricevere dati, occupando alternativamente l'*unica linea disponibile*: mentre uno trasmette quello opposto riceve, e viceversa. L'idea è resa bene dalla *ricetrasmissione a radio frequenza*, come *holky-tolky* o *City Band* (*CB*), nella gestione delle quali si parla uno per volta
- **full-duplex**: i dati possono viaggiare nello stesso tempo in entrambe le direzioni, come le automobili su una strada a 2 sensi di marcia; entrambi i terminali possono trasmettere e ricevere dati contemporaneamente, naturalmente *su 2 conduttori diversi*

- Lo standard **RS232** è stato predisposto per la comunicazione *full-duplex*: prevede infatti un conduttore per la trasmissione distinto da quello per la ricezione; l'uso della modalità *full-duplex* invece della *half-duplex* dipende dalla capacità dei modem coinvolti nel trasferimento dati: nell'uno o nell'atro caso essi prowederanno ad attivare i necessari sopracitati segnali di *hardware handshaking*.

- Come Funziona... - La Comunicazione Seriale: Dettagli

4.3 Comunicazione Seriale Asincrona

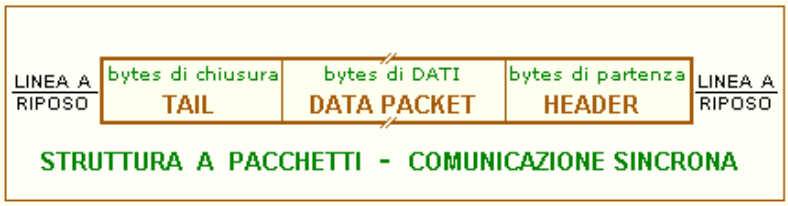
- La *comunicazione seriale* può avvenire in due modi: **sincrona** e **asincrona**:
  - la parola ***sincrono*** richiama l'idea di una cosa che avviene *in un preciso istante*; tecnicamente la cosa è possibile solo in presenza di un *segnale di sincronismo*; di solito si tratta di un **breve impulso** (spesso detto di ***strobe***) ma il *sincronismo* per eccellenza è un segnale di **clock** o più precisamente il suo *fronte di salita* o il suo *fronte di discesa*
  - un fenomeno ***asincrono*** è, per semplice ed owia dualità, un evento ***privo*** *di sincronismo* cioè tale da manifestarsi in istanti di tempo assolutamente imprevedibili
- La *ricetrasmissione seriale* **sincrona** deve dunque prevedere **2** *linee* ***distinte***, una per i *dati* e una per il *clock*: la seconda (generata dal trasmettitore) è usata dal ricevitore per stabilire con assoluta precisione la collocazione dei dati sulla prima.
- La *ricetrasmissione seriale* **asincrona** dispone invece della ***sola*** *linea dati*; per cui sembra *difficile ricostruire* l'informazione seriale; in effetti, pur non disponendo di una specifica linea di clock ***condivisa***, i concorrenti asincroni sfruttano una speciale configurazione dell'informazione scambiata, caratterizzata dalla presenza di ***speciali bit***, davanti e dietro quelli del dato.
- La loro presenza, in aggiunta alla ***precisa durata*** di ciascun bit, consente la *ricostruzione* del byte ricevuto; ma appare evidente sia la necessità di trasmettere (e ricevere) un numero maggiore di bit, rispetto a quello strettamente necessario, sia una certa probabilità di errore.
- La ***porta seriale*** di un PC supporta la *comunicazione seriale* ***asincrona*** e le specifiche **RS232** sono state considerate (fino a qualche anno fa) l'unico *standard* per descriverne a fondo ogni dettaglio, sia elettrico che meccanico.
- La conoscenza dello standard ***RS232C*** ci aiuterà a capire:
  - come sono fatti i ***segnali*** che corrono sui cavi seriali
  - come sono fatti i ***connettori*** e in quanti modi è possibile configurare i rispettivi pin
  - come awiene esercitato il ***controllo*** (*handshake*) per scambio di informazioni seriali tra computer e periferica
  - quali sono i minimi garantiti per le principali caratteristiche.
- Data la sua importanza gli dedicheremo molte delle pagine seguenti...

- Come Funziona... - La Comunicazione Seriale: Dettagli

4.4 Comunicazione Seriale Sincrona

- La *modalità* ***sincrona*** si presta a ***velocità più elevate*** ed è ***più efficiente***, rispetto a quella *asincrona*, sia per minore numero di bit scambiati (i *bit speciali* non sono necessari) ma anche per la particolare *struttura a pacchetti* (*data packet*) che consiste nel raggruppare insieme numerosi caratteri (per esempio *2kBytes*, 2048) in un *blocco dati*, posto poi tra:
  - una *sequenza di partenza* (***header***) con le ***informazioni*** necessarie ai dispositivi in ascolto per riconoscere come proprio il *blocco dati* e sul modo di utilizzarne il contenuto; nell'*header* sono presenti anche identificatori e indirizzi: se non si accoppiano con quelli attesi il seguente *blocco dati* viene semplicemente ignorato dal dispositivo ricevente

- una *sequenza di chiusura* (***tail***), nella quale il trasmettitore include un *codice di controllo* (*check code*) per consentire al ricevitore di determinare se il *blocco dati* è stato ricevuto senza errori.



- La possibilità di ***rilevare gli errori*** è un altro punto di forza della *modalità sincrona* rispetto a quella *asincrona*, che ne è priva; per contro è certamente soggetta a ***costi maggiori*** (2 fili al posto di 1 e circuiti aggiuntivi per assicurare la condivisione del segnale di clock tra trasmettitore e ricevitore).
- Nonostante gli indiscussi pregi la *modalità* ***sincrona*** non viene utilizzata nell'ambito della *comunicazione seriale* di un *Personal Computer*, per questo ogni ulteriore approfondimento è *fuori luogo*.

RS232	Standard RS232C	Porta Seriale

1 - Standard RS232 - Premesse e Generalità

1.1 Premesse e Generalità

- Lo standard *originale* **RS232** affronta tutti gli aspetti della ***comunicazione seriale asincrona***., descritte con dettaglio in più di 40 pagine...
- Lo scopo di uno **Standard** è quello di suggerire una linea comune nell'uso e nella programmazione di un dispositivo; solo in questo modo oggetti prodotti da diversi costruttori potranno essere interconnessi tra loro senza creare problemi di compatibilità.
- Lo **Standard** diventa perciò non solo l'insieme delle *specifiche software ed hardware* di un dispositivo, ma anche lo stimolo a proporlo a prezzi corretti, dettati dalla possibilità della produzione in grandi quantità e dalla sana concorrenza.
- Ciò nonostante è cosa di tutti i giorni riscontrare ***desiderio di visibilità*** e (talvolta) ***trasgressione delle regole***, di norma per trarne vantaggio: entrambe le cose portano a dispositivi non compatibili con quelli standard; di solito le differenze si riscontrano nei *cablaggi* e nei *connettori*, sui quali alcuni segnali sono assenti o in posizione fuori norma.
- Per questa ragione si possono trovare situazioni (molto frequenti nell'ambito seriale) per le quali l'uso dell'apparecchio è subordinato all'acquisto di un *cavo speciale* o di un *adattatore di connettori*; i *furbi* hanno vinto ancora..
- La situazione di incompatibilità maggiore si riscontra, per altro, nell'ambito dei *Sistemi Operativi*; questi particolari *software* hanno l'abitudine di *organizzare* e *far funzionare* ogni cosa presente sul computer o ad esso collegata via *porta seriale* (o *parallela* o *USB*); sebbene la loro capacità di riconoscere dispositivi sia senz'altro rimarchevole è *impossibile* prevederli tutti.
- Per questa ragione ogni costruttore allega, per ogni *Sistema Operativo*, un particolare programma eseguibile, detto ***driver***, in grado di rendere visibile e utilizzabile il suo prodotto; questo importante meccanismo gli consente anche di progettare il dispositivo in modi non strettamente soggetti allo **Standard**: al ***driver*** spetterà il compito di istruire il *Sys-Op*, così come l'*interprete* ci aiuta a comprendere una lingua diversa dalla nostra.
- Abbiamo visto che, fin dal suo apparire sul mercato nel 1982, le **regole** di produzione ed uso dell'***interfaccia seriale*** sono state affidate ad uno **Standard** molto diffuso e noto, caratterizzato dalla sigla **RS232** e presentato vent'anni prima (1962) dalle Associazioni delle Industrie Elettroniche (**EIA**) e delle Telecomunicazioni (**TIA**).
- Lo standard **RS232** utilizza per l'***interfaccia seriale*** un protocollo di tipo ***asincrono*** e codifica con precisione:
  - le *caratteristiche elettriche* dei segnali coinvolti
  - la *struttura* e *temporizzazioni* dei dati seriali adatti alla comunicazione asincrona
  - la *definizione* dei *segnali* e dei *protocolli* hardware e software destinati al controllo del flusso di dati seriali su un canale telefonico a frequenza vocale, cioè richiesti da un ***modem*** chiamato alla modulazione del segnale digitale prodotto dall'**UART**
  - il tipo di *connettore* necessario e la disposizione dei segnali coinvolti sui suoi pin
  - il *tipo* e la *lunghezza* massima dei possibili *cavi di collegamento* e la *velocità massima* possibile, in funzione delle rispettive caratteristiche

- Ciascuna di queste *specifiche* sarà trattata in dettaglio nelle pagine seguenti; di certo discuteremo le *norme* attualmente in uso, dato che la naturale evoluzione tecnologica, a più di quarant'anni dalla sua introduzione, ha introdotto sullo standard **RS232** gli aggiornamenti necessari per migliorare e rendere attuale la *comunicazione seriale*.
- Ciascun aggiornamento è riconoscibile dalla lettera aggiunta in coda al numero; la sigla RS232**C** indica la *terza revisione* dello standard **EIA** e corrisponde alle specifiche dello standard **CCITT** raccomandazione **V.24**; l'ultima modifica è stata presentata nel 1991 con la sigla EIA232**E**, introducendo tra l'altro l'uso di cavo schermato e cambiando il nome da **RS232** a **EIA232**.

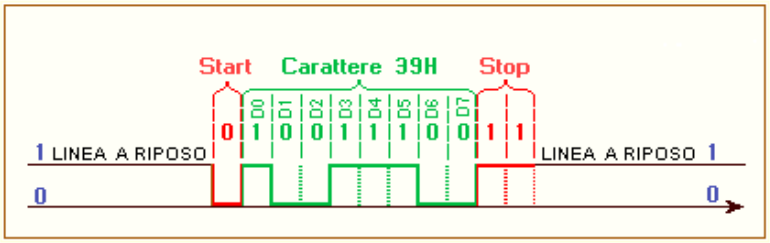
- Standard RS232 - Struttura di un Dato Seriale Asincrono

2.1 Struttura di un dato (Frame) seriale asincrono

- La ***porta seriale*** è uno dei possibili modi per scambiare informazioni con il *computer*, o meglio, con il *processore* che lo governa; nelle pagine precedenti abbiamo sottolineato che, nella *comunicazione dei dati*, è fondamentale disporre di *sincronismi*, indispensabili per stabilire l'esatto momento per leggere o per scrivere.
- Se le 2 postazioni collegate tra loro disponessero di una linea di **clock** in comune (*comunicazione* ***sincrona***) sarebbe facile per l'una *essere avvisata* dall'altra, *quando il dato è pronto*; ma questo non è il caso della ***porta seriale*** di un computer, per la quale lo standard **RS232** prevede una *comunicazione* di tipo ***asincrono***.
- Poichè il problema dei sincronismi di lettura e scrittura **non può** essere trascurato è stato inventato un meccanismo geniale, in grado di consentire la ricezione dati anche senza che il trasmettitore sia costretto a spedire un segnale di clock al ricevitore (e, naturalmente, viceversa...): basta infatti la presenza di **bit speciali** aggiunti **davanti** e **dietro** quelli del **byte** da trasmettere, detti per questo *bit di* ***start*** e *bit di* ***stop***.

Vediamo come funziona: supponiamo che una linea **priva** di informazioni sia **tenuta** ad un livello logico ***costante*** (per esempio *alto*): **come possiamo far capire** al ricevitore **che sta per arrivare un dato**?

- Sono certo che hai già capito... Basta *commutare* per un po' la linea al *livello opposto* (***start*** = *livello basso*, nel nostro esempio); il ricevitore consumerà il tempo del *bit di* ***start*** e si accingerà a leggere i bit del dato, nel numero concordato con il trasmettitore.
- A questo punto non guasta ***dare sicurezza*** al meccanismo, *riportando* per un po' la linea al *livello di riposo* (***stop*** = *livello alto*, nel nostro esempio); se non dovessero seguire altri dati, la linea sarebbe esattamente nelle condizioni di partenza, cioè ***inattiva*** (sempre *a 1 logico*), ***in attesa*** (stato di ***idle***) dell'arrivo del prossimo dato.



- La presenza del *bit di* ***stop*** (di *durata* uguale a quella di tutti i *bit* precedenti) può sembrare un *inutile spreco*: sembrerebbe essere sufficiente riportare la linea in *stato* ***inattivo*** (*livello alto*) senza costringere il ricevitore a consumare un altro tempo/*bit*; in realtà questo periodo (di clock) è molto utile:
  - dà al ricevitore il tempo di ***pulire*** il carattere ricevuto (dai bit non di dato) e di verificarne l'attendibilità (se viene richiesto un controllo di parità)
  - introduce un ***piccolo ritardo***, in attesa del carattere successivo, in grado di assorbire le eventuali leggere differenze tra i 2 clock del ricevitore e del trasmettitore
- Per entrambe le ragioni può esser necessario un tempo maggiore: per questa ragione lo standard **RS232** consente la presenza di **1**, **1 e mezzo**, o **2** *bit di* ***stop***; naturalmente non bisogna esagerare: prevederne più di uno aumenta il numero di bit coinvolti e richiede un tempo maggiore per la ricetrasmissione.

- Standard RS232 - Struttura di un Dato Seriale Asincrono

2.2 Importanza del Clock generato localmente

- Si capisce che tutto il marchingegno è basato sulla **durata** e sulla **quantità** dei bit coinvolti: può funzionare **solo** se queste condizioni sono state **concordate** e **condivise** con certezza tra trasmettitore e ricevitore; prima di iniziare la ricetrasmissione si deciderà, per esempio, di condividere dati a **8 bit** alla velocità di **9600** bit per secondo.

Fissare la ***velocità*** significa dunque *stabilire la* ***durata*** prevista *per ogni bit* coinvolto; in altri termini significa stabilire la *cadenza temporale* (***frequenza***) con cui ciascun bit è posto sulla linea dal trasmettitore e con cui la linea verrà analizzata dal ricevitore, quando sarà il momento di leggerli.

- Le 2 estremità del collegamento devono dunque **disporre di clock** generati localmente, supposti della **stessa** frequenza e owiamente del tutto indipendenti tra loro:
  - il compito del ***clock*** in trasmissione è quello di stabilire la ***durata*** di ogni bit posto, in sequenza, sulla linea; ciascun bit sarà mantenuto per un tempo pari a un ***periodo*** *del* ***clock***, cioè all'*inverso della* ***velocità*** di ricetrasmissione; per esempio con velocità di **9600** bit per secondo ogni bit della sequenza avrà durata di **104** microsecondi
  - il compito del ***clock*** in ricezione è quello di stabilire il *tempo di campionamento* di ciascun bit, cioè l'istante nel quale cercherà di stabilirne il valore; per avere la certezza di "*centrarlo*" il ricevitore ***ritarda*** *di* ***mezzo periodo*** l'analisi del suo *valore*, esercitandola così esattamente nel mezzo della ***durata*** del bit corrente

- In questo *contesto* ***asincrono*** è importante sottolineare il ruolo del *bit di* ***start***:
  - avvisa il ricevitore che sulla linea è disponibile un dato: solo in questo momento il ricevitore fa partire il suo clock, **sincronizzandolo** di fatto con quello del trasmettitore, anche se non *fisicamente* presente
  - il *ritardo di propagazione* del supporto della ricetrasmissione (cavo, aria, ...) è irrilevante, qualunque sia la sua entità: prima o poi la sequenza di bit arriverà a destinazione, **ri-sincronizzando** il ricevitore con i trasmettitore ***ogni volta*** che viene rilevato il *bit di* ***start***, cioè ogni volta che arriva un nuovo dato
  - dunque, nella *fase di* ***attesa*** il ricevitore **non è** sincronizzato con il trasmettitore

- Sebbene la **precisione** delle frequenze *locali* di **clock** sia importante, è facile capire che la *differenza* di *qualche percentuale* non è particolarmente dannosa: bene o male *il valore* del bit sotto test *sarà comunque rilevato*; se l'errore dovesse essere consistente (o se le 2 velocità fossero diverse) si rischia invece di affidare ad un bit il valore destinato ad uno *posto in posizione diversa*, rendendo inutile il significato del dato ricevuto:
  - nell'ipotesi di ricetrasmettere **10 bit** (**1** di *start*, **8** di *dato* e **1** di *stop*) il tempo necessario per rilevarli tutti è di **10 periodi** di clock
  - se la frequenza di ricezione fosse leggermente *più piccola* di quella del trasmettitore il campionamento su ogni bit verrebbe *operato in ritardo*, mettendo a rischio il riconoscimento degli ultimi della sequenza
  - in particolare *l'**ultimo** dei **10*** sarà ancora riconosciuto solo se il ritardo dell'ultimo campionamento *non supera il mezzo periodo*
  - l'errore massimo tollerato è, dunque di **1 su 20** "*mezzi periodi*", cioè la **differenza** tra le 2 frequenze di clock (per altro generate in posti assolutamente diversi e lontani tra loro) potrà essere al massimo del **5%**
  - è facile comprendere che questa previsione è rilevata in *condizioni limite*: in nessuna situazione reale è tollerata una visione così ottimista, consigliando di ***restringere*** la tolleranza di ***almeno 10 volte***.

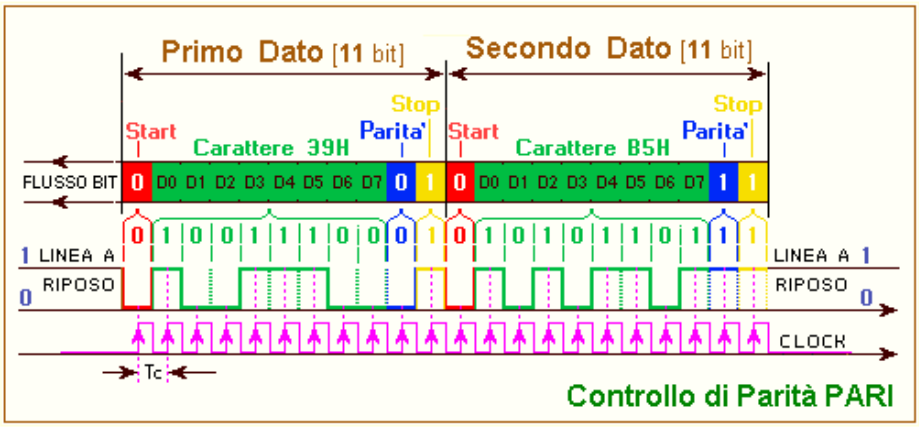
- La *comunicazione seriale* di tipo ***asincrono*** è dunque un sistema basato fortemente sulla **temporizzazione** dei bit coinvolti.

- Standard RS232 - Struttura di un Dato Seriale Asincrono

2.3 Frame seriale completo dei bit di Parità

- La struttura tipica dell'*informazione asincrona* è detta **frame** proprio per la sua forma, che tende a corniciare (*framed*) il byte di **dato**, cioè a *racchiuderlo tra 2 bit di sincronismo* (uno davanti, di *start*, e uno dietro, di *stop*).
- In questo modo il *ricevitore asincrono* è in grado di sapere quando inizia e quando finisce, anche senza disporre della linea di clock condivisa, tipica dei *sistemi sincroni*.
- Il **segnale** digitale seriale **asincrono** viene creato dal **trasmettitore** dell'**UART** collegato al processore del computer che fornisce il byte, ed è posto sulla linea (con i dovuti livelli di tensione) in questa sequenza temporale: prima il *bit di* ***start*** (sempre un bit a **0** logico), poi i *bit del* ***dato***, a partire da quello meno significativo (**LSB**, **L**east **S**ignificant **B**it, **D0**) fino a quello più significativo (**MSB**, **M**ost **S**ignificant **B**it, **D7**) e infine il (o i) *bit di* ***stop*** (sempre un bit a **1** logico).

- La figura seguente mostra i livelli logici di una linea seriale chiamata a **ricevere 2 dati a 8 bit**; in essa si nota una importante novità: il *bit di **parità***.



- Da notare che l'*andamento temporale* si riferisce ai **bit che entrano** in un *UART **ricevitore***; il **clock** generato localmente da quest'ultimo campionerà ciascun bit (nell'esempio sul *fronte di salita*) esattamente *nella parte centrale* della sua durata.

- Standard RS232 - Struttura di un Dato Seriale Asincrono

2.4 Tipologie previste per il controllo di Parità

- In presenza di *scariche o interferenze elettriche* o a causa della possibile *rumorosità* della linea di comunicazione, non è da escludere che uno o più bit del ***frame** seriale **asincrono*** possa subire una commutazione o possa addirittura essere perduto; in entrambi i casi il dato ricevuto risulterà *diverso* da quello trasmesso e, in assenza di provvedimenti mirati a scoprire l'***errore***, non sarà possibile distinguerlo da quelli corretti e neppure farlo eventualmente ritrasmettere.
- Per tentare di risolvere il problema lo standard **RS232** prevede la presenza (facoltativa) del *bit di **parità***; il suo valore dipende dal **tipo** di **parità** (***pari*** o ***dispari***) adottato di comune accordo dal trasmettitore e dal ricevitore.
- In pratica il valore stabilito per esso dall'**UART** sarà quello che **rende *pari* o *dispari*** il numero di *bit a **1*** presenti **prima** del bit di stop (quindi quelli del *dato* ospitato e dello stesso *bit di **parità***).

- Con questa logica, con controllo di **parità *pari*** (***even***):
  - il *bit di **parità*** è lasciato a **0** se il *dato seriale* contiene già un numero pari di *bit a **1*** (esempio: 101100010, B1H +**0**)
  - il *bit di **parità*** è forzato a **1** se il *dato seriale* contiene un numero dispari di *bit a **1*** (esempio: 100100101, 92H +**1**)

- Dualmente, con controllo di **parità *dispari*** (***odd***):
  - il *bit di **parità*** è lasciato a **0** se il *dato seriale* contiene già un numero dispari di *bit a **1*** (esempio: 100100100, 92H +**0**)
  - il *bit di **parità*** è forzato a **1** se il *dato seriale* contiene un numero pari di *bit a **1*** (esempio: 101100011, B1H +**1**)

- Standard RS232 - Struttura di un Dato Seriale Asincrono

2.5 Tipologie previste per i bit di Dato

- Se è in atto un controllo di **parità** (**pari** o **dispari**) il *bit di **parità*** sarà inserito dal trasmettitore prima del *bit di **stop***.
- Non appena ha incamerato tutti i bit del ***frame** seriale* il ricevitore è dunque in grado di rilevare errori operando un conteggio analogo e verificando il risultato con il valore del *bit di **parità***.
- Appare per altro evidente che questo metodo è certamente poco affidabile e, non di rado, ci si rinuncia attivando una *comunicazione seriale* con **nessuna parità**, cioè senza aggiungere alcun bit.

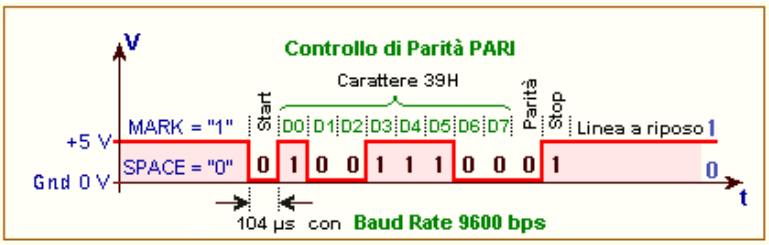
- Lo standard **RS232** prevede, nel ***frame*** *seriale*, la presenza da **5** a **8** *bit di* ***dato***; la dimensione a 5 o 6 bit è senz'altro curiosa, mentre più logica è quella a **7** bit, tipica di un *carattere Ascii standard*.
- Nei progetti comuni è preferibile rinunciare al controllo di parità (comunque precario) e comunicare a **8** bit (= **1** byte), classica dimensione di numero binario, ma adatta anche a rappresentare un *carattere Ascii esteso*.

- E' curioso sottolineare che nell'ambito della comunicazione seriale ***asincrona*** siano significativi i **bit**, piuttosto che i **bytes**; la presenza dei bit addizionali di ***start*** e di ***stop*** la rende **poco efficiente** e **più lenta** di quella ***sincrona***: per ogni 10 bit trasmessi solo gli 8 del byte di dato, sono effettivamente riutilizzabili.
- Ciò significa che i 2 bit su 10, necessari per la sincronizzazione, producono un *aumento del tempo* del **20%**.
- Di certo l'**UART**:
  - scarterà automaticamente i bit di start, parità e stop dal *frame ricevuto*, per dare al processore un byte *pulito*
  - aggiungerà automaticamente i bit di start, parità e stop per dare al modem un byte *formattato* (*frame* adatto alla *trasmissione* RS232)

3 - Standard RS232 - Specifiche elettriche dei Dati Seriali Asincroni

3.1 Specifiche elettriche TTL (UART)

- La sequenza dei bit prevista dallo standard **RS232C** nel ***frame*** *seriale*, deve ora essere fisicamente realizzata dall'***interfaccia seriale***; di questo si occupa l'**UART** (**U**niversal **A**synchronous **R**eceiver/**T**ransmitter), appositamente inventato da *National Semiconductor* su richiesta della *IBM* per il suo *PC originale*.
- Poichè la tecnologia di questo componente è di tipo **TTL** i livelli elettrici dei bit a **1** sono di **+5V** e quelli dei bit a **0** sono di **0V**; la forma d'onda in uscita dalla UART del ***frame*** *seriale* è dunque la seguente; l'esempio organizza il byte **39**H con controllo di ***parità pari*** e un solo bit di ***stop***, oltre al necessario bit di ***start***:

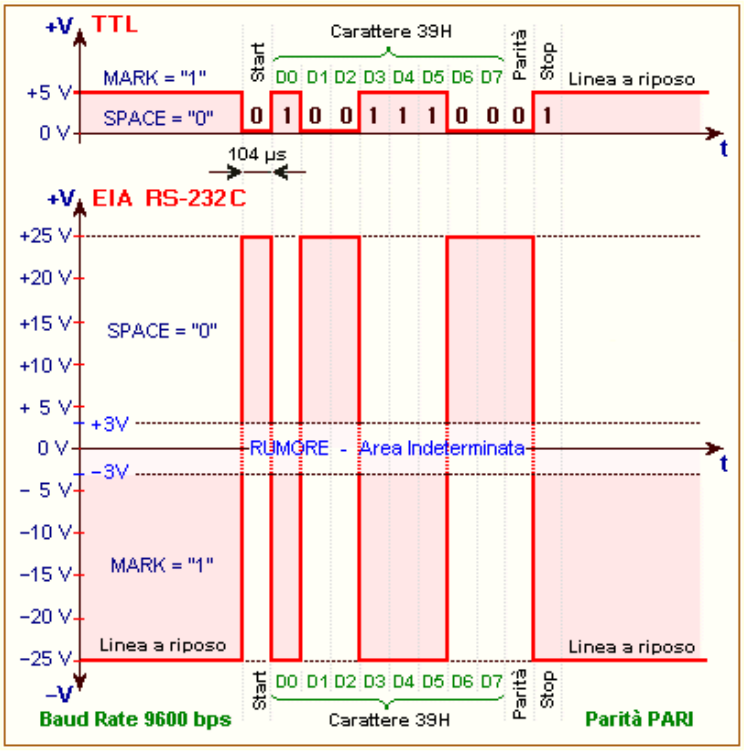


3 - Standard RS232 - Specifiche elettriche dei Dati Seriali Asincroni

3.2 Specifiche elettriche RS232

- Per garantire un **elevato margine di rumore**, lo standard **RS232** prevede valori *di tensione molto diversi* da quelli **TTL** da cui hanno origine; il problema del rumore (*Noise*) e della capacità di esserne immuni (*margine*) è legato al fatto che qualunque segnale, percorrendo la linea (di solito un cavo elettrico) subisce una inevitabile attenuazione, direttamente proporzionale alla sua lunghezza e largamente dovuto alla sua capacità elettrica.
- I *livelli di tensione* **RS232**, introdotti per limitare le *perdite di tensione* attraverso il cavo, sono compresi:
  - tra **-25V** e **-3V** per il livello logico **1** (**+5V** in **TTL**), detto **MARK**
  - tra **+25V** e **+3V** per il livello logico **0** (**0V** in **TTL**), detto **SPACE**
- I valori indicati sono misurati con riferimento alla linea di **massa** e sono quelli massimi (**±25V**) e minimi (**±3V**) tra i quali i segnali di *ciascuna linea* **RS232** sono ritenuti validi; i valori interni, tra **-3V** e **+3V**, sono considerati ***rumore***,

- Poichè l'***alimentatore*** di un Personal Computer offre le tensioni **±5V** e **±12V**, di solito richieste dai dispositivi in esso presenti, anche l'effettiva ampiezza massima dei segnali RS-232 sarà di **±12V**.



3 - Standard RS232 - Specifiche elettriche dei Dati Seriali Asincroni

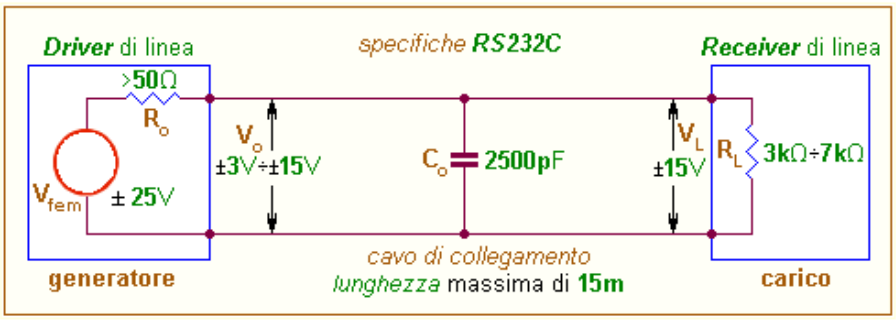
3.3 Traslatori di livello, TTL <> RS232

- E' curioso notare che, contro ogni apparente aspettativa al livello **1** logico (**+5V** TTL sui pin dell'**UART**) è associata una *tensione* ***negativa*** e allo **0** logico (**0V** TTL) è associata una *tensione* ***positiva***; anche i termini **MARK** (**1**) e **SPACE** (**0**) sono più un omaggio alla storia (della telegrafia) che una effettiva esigenza.
- In questi termini:
  - la presenza di dato sulle *linee di ricetrasmissione* sarà annunciata da uno **SPACE=+12V** (*bit di* ***start***) e sarà chiusa da un **MARK=-12V** (*bit di* ***stop***), mantenuto sulla linea anche quando non ci sono dati (*stato di riposo*, **IDLE**)
  - le *linee di controllo di flusso* (*handshaking hardware*) essendo ***attive basse*** (**0** logico) in *ingresso* all'UART (come **CTS**, **DSR**, **RI** e **CD**) o in *uscita* dall'UART (come **DTR** e **RTS**), per lo standard **RS232** sono ***attive*** (**ON**) se in stato **SPACE=+12V** e ***non attive*** (**OFF**) se in stato **MARK=-12V**
- Per questa ragione tra l'**UART** (*porta seriale*, *computer* **DTE**) e il connettore sono presenti dei *dispositivi* ***invertenti*** (***Traslatori di Livello***) in grado di trasformare le *tensioni positive* **RS232C** dei segnali attivi, provenienti o destinati al *modem* **DCE**, in livelli **TTL** (*bassi*) per o dall'UART:
  - **Ricevitori** (***Receiver***) di linea (per esempio i **DS1489**), prima del *ricevitore* e sulle *linee di controllo in ingresso*, per trasformare le *tensioni* ***positive*** dei segnali **CTS**, **DSR**, **RI** e **CD** in arrivo dal *modem* **DCE** in *livelli* ***0*** *TTL* sulle linee *attive basse* **CTS**, **DSR**, **RI** e **CD** in ingresso all'**UART**
  - ***Driver*** di linea (per esempio i **DS1488**), dopo il *trasmettitore* e sulle *linee di controllo in uscita*, per trasformare i *livelli* ***0*** *TTL* delle linee *attive basse* **DTR** e **RTS** in uscita dall'**UART** in *tensioni positive* per i segnali **DTR** e **RTS** diretti al *modem* **DCE**
- Naturalmente entrambi i traslatori di livello agiranno sulle *tensioni* ***negative*** **RS232** (di solito per i *segnali non attivi* o *linee a riposo*) trasformandole in *livelli* ***1*** *TTL* **UART**, e viceversa.
- Oltre alle *tensioni* di lavoro le *specifiche elettriche* ***RS232C*** stabiliscono i valori limite anche di *altre grandezze*, al fine di **ridurre** le **interferenze** e la **perdita d'informazione**; in realtà i valori fissati dalle norme sono ampiamente garantiti dai ***Traslatori di Livello***, nella trasformazione da livelli **TTL** UART a tensioni **RS232C** dei segnali sul connettore seriale, molto più restrittivi, come vedremo nelle prossime pagine.

3 - Standard RS232 - Specifiche elettriche dei Dati Seriali Asincroni

3.4 Circuito equivalente di un sistema seriale

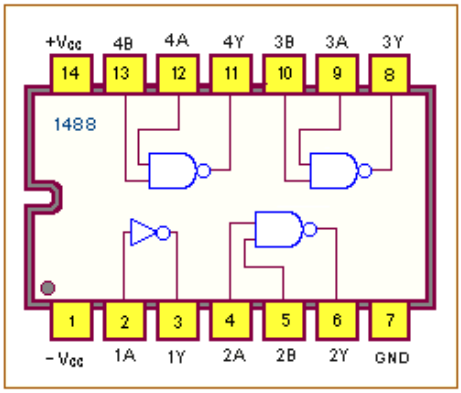
- Dal punto di vista elettrico ciascun **segnale in uscita** dal connettore (garantito da un ***Driver*** di linea) può ritenersi un **generatore**; le *specifiche **RS232C*** prevedono:
  - *forza elettromotrice* V_{fem} da **-25**V (per la logica "**1**", **MARK**) a **+25**V (per la logica "**0**", **SPACE**); owiamente è anche la *tensione d'uscita* V_o *a vuoto* (cioè in assenza di collegamento con il ricevitore)
  - *resistenza interna* R_o non inferiore a **50**Ω, pensata per limitare a **±500**mA la *corrente massima* in caso di *cortocircuito* con qualsiasi altra linea, compresa quella di massa: $I_{oS} = V_{fem}/R_{om}$ = **25**V/**50**Ω = **500**mA; deve garantire inoltre una *tensione in arrivo* V_L da **±3**V a **±15**V
- Analogamente, ciascun **segnale in ingresso** al connettore (accolto dal ***Receiver*** di linea) può ritenersi un **carico**; le *specifiche **RS232C*** prevedono:
  - *resistenza di carico* R_L compresa tra **3kΩ** e **7kΩ**
  - assenza di *componente induttiva*; per evitare danni durante le commutazioni rapide
  - *tensione d'ingresso* V_L da **±3**V a **±15**V
- Per il *cavo di collegamento* la norma originale prevede una *lunghezza* massima di **15m**, sostituita (con la versione ***EIA232D***) dalla una specifica sulla *capacità* massima del cavo stesso, fissata a ***2500pF***.
- In sintesi il sistema di *comunicazione **RS232C*** può essere rappresentato dal seguente ***circuito equivalente***:



4 - Standard RS232 - Dispositivi Traslatori di Livello: dettagli

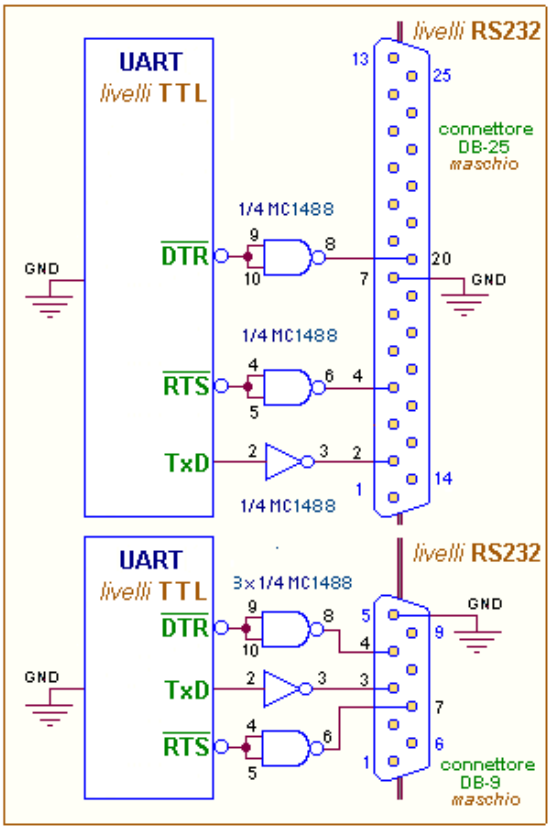
4.1 Driver di linea MC1488

- Un **Driver di Linea** usato frequentemente è il componente **MC1488** della Motorola, ma prodotto anche da altri costruttori; il suo **pin-out** è illustrato dal seguente schema:



- I simboli utilizzati nello schema (**Not** e **Nand**) servono solo per sottolineare la loro azione ***invertente***; non sono gli operatori logici che siamo abituati a trattare ma *traslatori di Livello* in grado di trasformare i *livelli **TTL 0** e **1*** (della *linea* **TxD** e delle *linee di controllo* **in uscita** dall'**UART**, come i segnali **DTR** e **RTS** diretti al *modem* **DCE**) rispettivamente in *tensioni RS232* ***negative*** *e* ***positive*** sul *connettore seriale*.

- Le caratteristiche che un **Driver di Linea** deve avere secondo le *specifiche elettriche* ***RS232C*** sono:
  - le *linee in uscita* devono sopportare un cortocircuito con qualsiasi altra linea, compresa quella di massa; la *corrente di cortocircuito* non deve essere superiore a **±500mA**
  - l'*impedenza d'uscita* deve sempre essere non superiore a **50Ω**
  - la *pendenza dei fronti di salita e di discesa* dei segnali RS232, cioè la *velocità del passaggio da un livello logico all'altro* (detta *slew rate*) deve essere al massimo di **30V/µs**; la probabilità di interferenze (*cross-talk*) tra linee adiacenti aumenta se i segnali cambiano stato in modo più veloce
- I segnali **UART** tipicamente coinvolti in questa trasformazione sono mostrati qui di seguito:



per il Driver 1

INPUT TTL	OUTPUT RS232
A	Y
1	0
0	1

per i Driver 2, 3 e 4

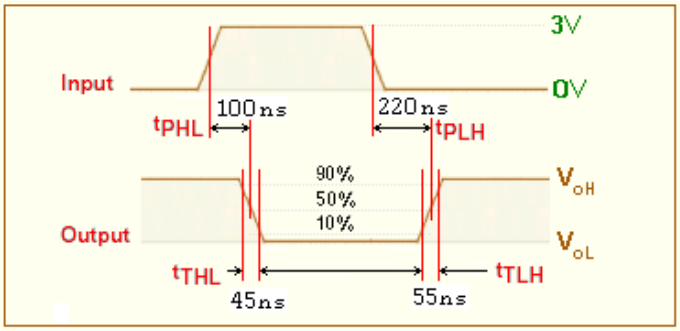
INPUT TTL		OUTPUT RS232
A	B	Y
1	1	0
0	X	1
X	0	1

1 livello logico alto
0 livello logico basso
"X" valore logico indifferente

- La presenza della logica **Nand** è irrilevante per lo scopo finale (trasformare *livelli* ***TTL*** in *tensioni* ***RS232***): basta infatti unire entrambi i pin di ciascun *traslatore di Livello* per garantire comunque la necessaria logica ***invertente***.
- Naturalmente all'occorrenza il *secondo* pin d'ingresso può essere utilizzato per controllare il *segnale RS232*, rendendolo inattivo con la tecnica del *gating* per la quale uno **0** su di esso rende comunque *positivo* (cioè *inattivo*) il *segnale* in uscita, indipendentemente dal livello TTL proposto per esso dall'**UART**.
- I **Driver** di Linea richiedono 2 alimentazioni, da **±7**V a **±15**V: questo fatto può costituire un problema poichè non sempre esse sono disponibili; se si dispone di una sola alimentazione non rimane che affidarsi ad altri componenti, come il **MAX232**.
- La tabella raccoglie le principali **caratteristiche elettriche** del componente (i tempi sono stati rilevati con carico di **3kΩ**/**15pF**):

Caratteristiche Elettriche	Valori
potenza dissipata	333 mW (massima senza carico, assorbe 18,5 mA a ±9V)
	576 mW (massima senza carico, assorbe 24 mA a ±12V)
tensione in uscita a livello alto	7 V (tipica, con VIL=0,8V= "0", Vcc=±9V, RL=3kΩ)
	10,5 V (tipica, con VIL=0,8V= "0", Vcc=±13,2V, RL=3kΩ)
tensione in uscita a livello basso	-7 V (tipica, con VIH=1,9V= "0", Vcc=±9V, RL=3kΩ)
	-10,5 V (tipica, con VIH=1,9V= "0", Vcc=±13,2V, RL=3kΩ)
resistenza d'uscita	300Ω, minima
corrente massima esogata	±12mA, in condizioni di cortocircuito con ingresso irrilevante
tempo di propagazione da ingresso a uscita	350 ns, massimo dal fronte di salita del segnale d'ingresso
	175 ns, massimo dal fronte di discesa del segnale d'ingresso
tempo di transizione in uscita	100 ns massimo durante il fronte di discesa
	75 ns massimo durante il fronte di salita

- La durata dei fronti attivi (tempi di transizione, t_T, Transition time) dei segnali in uscita e i tempi di propagazione (t_P, Propagation time) da ingresso TTL a uscita RS232 sono mostrati nella seguente figura (HL High to Low, LH Low to High):



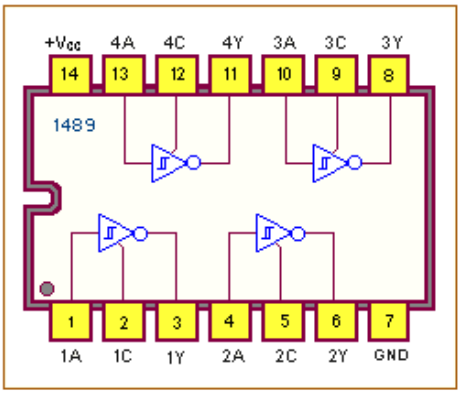
- I seguenti link rendono disponibili i **Data Sheet originali**, in **formato PDF**:

TEXAS INSTRUMENTS	http://focus.ti.com/lit/ds/symlink/mc1488.pdf
MOTOROLA intelligence everywhere	http://www.onsemi.com/pub/Collateral/MC1488-D.PDF
National Semiconductor The Sight & Sound of Information	http://www.national.com/ds/DS/DS1488.pdf
ON Semiconductor	http://www.onsemi.com/pub/Collateral/MC1488-D.PDF

- Standard RS232 - Dispositivi Traslatori di Livello: dettagli

4.2 Receiver di linea MC1489

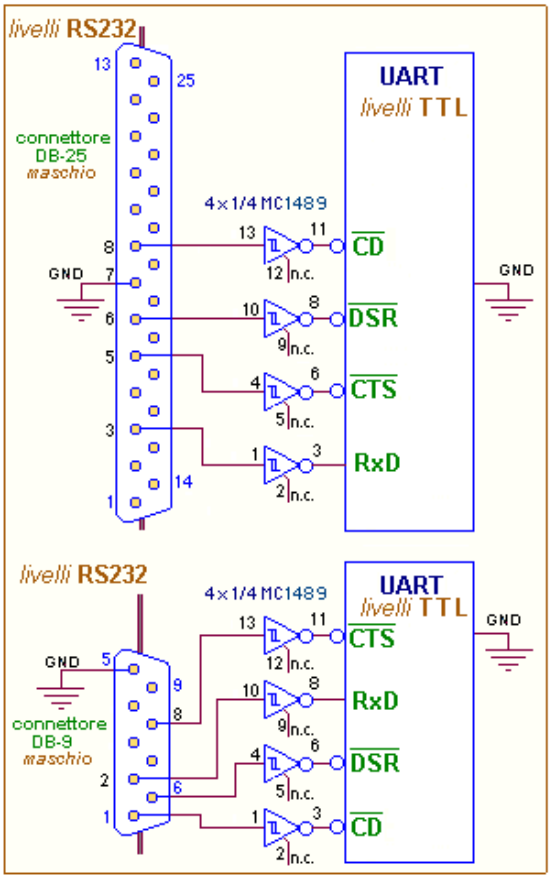
- Un **Ricevitore di Linea** usato frequentemente è il componente **MC1489** della Motorola, ma prodotto anche da altri costruttori; il suo **pin-out** è illustrato dal seguente schema:



- I simboli utilizzati nello schema sottolineano l'azione ***invertente*** esercitata da ciascun operatore *trigger di shmitt*; essi non sono gli operatori logici che siamo abituati a trattare ma *traslatori di Livello* in grado di trasformare le *tensioni RS232* ***negative*** e ***positive*** sul *connettore seriale* rispettivamente in *livelli* ***TTL 0*** e ***1*** per la *linea* **RxD** e per le *linee di controllo* **in ingresso** all'**UART**, come $\overline{\text{CTS}}$, $\overline{\text{DSR}}$ e $\overline{\text{CD}}$ per i segnali in arrivo dal *modem* **DCE**.
- Le caratteristiche che un **Ricevitore di Linea** deve avere secondo le *specifiche elettriche* ***RS232C*** sono:
  - l'*impedenza d'ingresso* deve sempre essere compresa tra **3kΩ** e **7kΩ** e non deve avere *componente induttiva*; durante le commutazioni rapide questa presenza produce *tensioni indotte* che possono portare, per brevi istanti, il livello dei segnali ricevuti a *valori superiori ai previsti ±25-volt*, con possibile danno per entrambe le parti collegate tra loro.

- la *capacità equivalente* non deve superare **2,5nF**; questa specifica (introdotta dalla versione EIA232D) sostituisce quella originale che prevedeva una *lunghezza* massima di **15m** per il *cavo di collegamento*
- in questi termini la *lunghezza* del *cavo* non è più prefissata, ma dipende dalla sua qualità, cioè dalla *capacità per unità di lunghezza* del *cavo* (con cavo schermato da ***30 pF per piede***, in buone condizioni ambientali, la lunghezza del cavo può essere anche di circa **25m**)
- in ambienti rumorosi (cioè in presenza di motori elettrici, rele' o altre apparecchiature che commutano carichi ad alta corrente) è comunque opportuno usare cavo più corti possibile e il più possibile lontani delle sorgenti di interferenza.

I segnali **UART** tipicamente coinvolti in questa trasformazione sono mostrati qui di seguito:



per i Driver 1, 2, 3 e 4

INPUT RS232		OUTPUT TTL
A	Control	Y
1	X	0
0	X	1

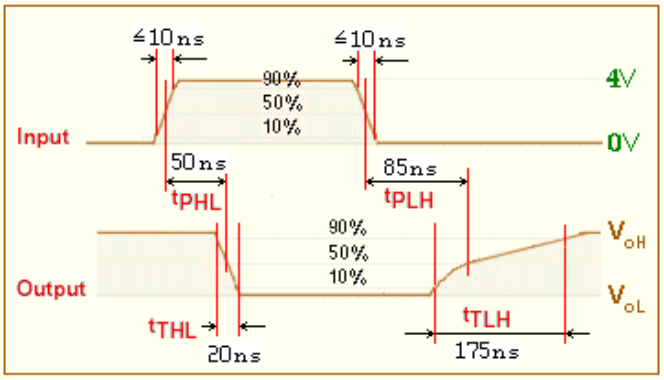
1 livello logico alto
0 livello logico basso
"X" valore logico indifferente

La presenza, nello schema dei *traslatori di Livello*, di un secondo ingresso, detto Response Control, è **logicamente irrilevante**: di solito esso è **non collegato** (*aperto*) ma può essere collegato a massa tramite un condensatore o a una delle 2 alimentazioni tramite un resistore, per *stabilizzare* o *modificare* il tipo di segnale d'ingresso.

Il **Driver** di Linea richiedono una sola alimentazioni, di solito di **5V**: la tabella raccoglie le principali **caratteristiche elettriche** del componente (i tempi sono stati rilevati con carico di **3,9kΩ** o **390Ω/15pF**):

Caratteristiche Elettriche	Valori
potenza dissipata	130 mW (massima , assorbe 26 mA a 5V)
tensione in uscita a livello alto	4 V (tipica, con VI=0,75V)
tensione in uscita a livello basso	0,2 V (tipica, con VI=3V)
corrente erogata tipica in uscita	0,5 mA (tipica TTL con uscita a "1")
corrente assorbita tipica in uscita	10 mA (tipica TTL con uscita a "0")
resistenza d'ingresso	tra 3kΩ e 7kΩ
tempo di propagazione da ingresso a uscita	50 ns, massimo dal fronte di salita dell'ingresso (RL=390Ω)
	85 ns, massimo dal fronte di discesa dell'ingresso (RL=3,9kΩ)
tempo di transizione in uscita	20 ns massimo durante il fronte di discesa con RL=390Ω
	175 ns massimo durante il fronte di salita con RL=3,9kΩ

- La durata dei fronti attivi (tempi di transizione, t_T, Transition time) dei segnali in uscita e i tempi di propagazione (t_P, Propagation time) da ingresso RS232 a uscita TTL sono mostrati nella seguente figura (HL High to Low, LH Low to High):



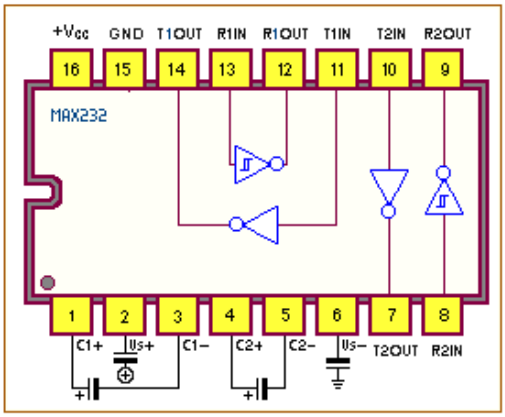
- I seguenti link rendono disponibili i **Data Sheet originali**, in **formato PDF**:

TEXAS INSTRUMENTS	http://focus.ti.com/lit/ds/symlink/mc1489.pdf
MOTOROLA intelligence everywhere	http://www.onsemi.com/pub/Collateral/MC1489-D.PDF
National Semiconductor The Sight & Sound of Information	http://www.national.com/ds/DS/DS1489.pdf
ON Semiconductor	http://www.onsemi.com/pub/Collateral/MC1489-D.PDF

- Standard RS232 - Dispositivi Traslatori di Livello: dettagli

4.3 Driver/Receiver di linea MAX232

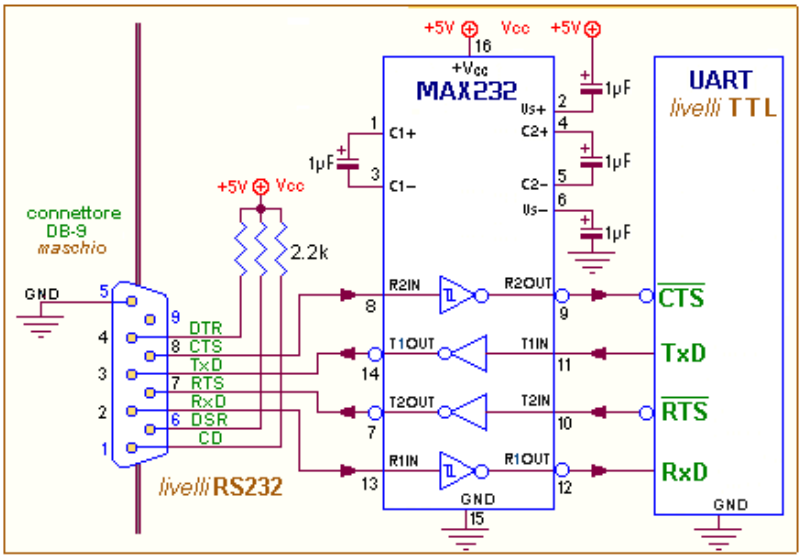
- Il componente **MAX232** della Maxim (prodotto anche da altri costruttori) ha i vantaggi di rendere disponibili sia la parte **Driver** (tipica del **MC1488**) che a quella **Receiver** (assicurata dal **MC1489**) e di richiedere **una** sola alimentazione di **5**V; il suo **pin-out** è illustrato dal seguente schema:



- I simboli utilizzati nello schema sottolineano l'azione ***invertente*** esercitata da ciascun operatore: non sono gli operatori logici che siamo abituati a trattare ma *traslatori di Livello*:
  - i 2 operatori del **Receiver Max232** trasformano le *tensioni RS232* ***negative*** *e* ***positive*** in arrivo dal *connettore seriale* rispettivamente in *livelli* ***TTL 0*** *e* ***1*** per la *linea* **RxD** e per le *linee di controllo* **in ingresso** all'**UART**, come $\overline{\text{CTS}}$, $\overline{\text{DSR}}$ e $\overline{\text{CD}}$ per i segnali in arrivo dal *modem* **DCE**

- i 2 operatori del **Driver Max232** trasformano i *livelli* ***TTL 0 e 1*** (della *linea* **TxD** e delle *linee di controllo* **in uscita** dall'**UART**, come **DTR** e **RTS** per i segnali diretti al *modem* **DCE**) rispettivamente in *tensioni RS232* ***negative e positive*** sul *connettore seriale*; queste *tensioni* (di solito **±10V**) necessarie per il suo servizio sono ottenute con una particolare tecnica (detta a *pompa di carica*) che sfrutta 4 condensatori, di solito posti esternamente, sottoposti a carica con una particolare frequenza di commutazione

- I segnali **UART** tipicamente coinvolti in questa trasformazione sono mostrati qui di seguito:



per i Receiver

INPUT RS232 RIN	OUTPUT TTL ROUT
1	0
0	1

per i Driver

INPUT TTL TIN	OUTPUT RS232 TOUT
1	0
0	1

1 livello logico alto
0 livello logico basso

- La tabella raccoglie le principali **caratteristiche elettriche** del componente:

Caratteristiche Elettriche		Valori
	potenza dissipata	50 mW (massima , assorbe 10 mA a 5V)
Driver	tensione in uscita a livello alto	7 V (tipica, con RL=3kΩ)
	tensione in uscita a livello basso	-7 V (tipica, con RL=3kΩ)
	resistenza d'uscita	300Ω , minima
	slew rate	30V/µs , massimo
	Velocità dati in uscita	120kbs , tipica
Receiver	tensione in uscita a livello alto	3,5 V, minima
	tensione in uscita a livello basso	0,4 V, massima
	corrente erogata tipica in uscita	1 mA (con uscita 3,5 V)
	corrente assorbita tipica in uscita	3,2 mA (con uscita 0,4 V)
	resistenza d'ingresso	tra 3kΩ e 7kΩ
	tempo di propagazione da ingresso a uscita	500 ns, tipico C1..C4 =1µF con Vcc=5V

- I seguenti link rendono disponibili i **Data Sheet originali**, in **formato PDF**:

TEXAS INSTRUMENTS	http://focus.ti.com/lit/ds/symlink/max232.pdf
MAXIM PRODUCTS SOLUTIONS	http://pdfserv.maxim-ic.com/en/ds/MAX220-MAX249.pdf
intersil	http://www.lgagroup.com/supporto/max232.zip

5 - Standard RS232 - Velocità Seriale: Unità di misura e tecniche

5.1 Velocità seriale: differenza tra Baud e bps

- Il *concetto* e la *misura* della **velocità** di ricetrasmissione nelle *comunicazioni seriali* si presta ad **interpretazioni contrastanti** che contribuiscono a generare **confusione** e **incertezza**.
- Già parlare di **velocità**, nello scambio di informazioni *tra 2 computer*, è una scelta infelice: sembrerebbe più logico parlare di **quantità di bytes** (o **di bit**) che possono essere scambiati **nell'unità di tempo**, ma l'assonanza tra *bit al secondo* e *metri al secondo* (la classica definizione di *velocità*) ha vanificato ogni decenza facendo di questa *sovrapposizione concettuale* una prassi normale.
- Come non bastasse, poichè i meccanismi seriali affidano l'informazione a ***segnali*** (***digitali*** o ***analogici***) il concetto di **velocità** può essere sovrapposto, ancora impropriamente, anche a quello di *frequenza* del *segnale seriale*, intendendo in questo caso il numero di *periodi al secondo*.
- Se pure in modo improprio la **velocità** seriale si presta dunque ad essere valutata da **2** diversi ***punti di vista***, ciascuno caratterizzato da una diversa *unità di misura*; di frequente (e *storicamente*) si parla di **baud** precisando però subito (quasi per scusarci..) che si tratta di **bit per secondo** (**bps**):
  - la misura di velocità in **baud** è strettamente legata alla *frequenza* del *segnale* sulla linea seriale, generata localmente e concordata tra ricevitore e trasmettitore; si può cioè definire come il numero di *fronti attivi* (o di *periodi*) di clock rilevabili sulla linea ogni secondo
  - la misura di velocità in **bps** si può definire come *il numero di bit* presenti sulla linea ogni secondo
- Il fatto sconcertante è che, come vedremo tra poco, con l'avvento dei *moderni modem*, nella descrizione del medesimo ambito seriale le 2 definizioni di velocità sono spesso compresenti e di valore diverso.
- La situazione nel momento della definizione dei ***primi*** *standard seriali* ha spinto le persone a ritenere che i 2 *punti di vista* fossero la stessa cosa:
  - lo standard **RS232** prevede che il *segnale seriale* possa assumere **solo 2** *livelli di tensione* durante ***ogni*** *periodo di clock*
  - per codificare il valore del *livello di tensione* presente in linea durante ciascun *periodo di clock* basta un solo **bit**
  - quindi, essendo la ricetrasmissione di tipo **binario**, la misura in **baud** (detta anche ***baud rate***) **coincide** con quella in **bit per secondo** (**bps**, detta anche ***bit rate***)
- Sebbene la seconda definizione (**bps**) fosse più calzante con la struttura dei ***frames*** *seriali asincroni* descritta nelle pagine precedenti, si decise di rendere omaggio all'inventore francese Jean Maurice Emile **Baudot**, eminente studioso dei codici telegrafici, l'inventore del primo *codice telegrafico binario* (1870) e della prima telescrivente.

5 - Standard RS232 - Velocità Seriale: Unità di misura e tecniche

5.2 Velocità seriale: limiti imposti dal canale

Non di rado si sente parlare di **Baud** con il significato di *bit al secondo*, **bps**, la vera unità di misura della *velocità seriale*; in realtà le due unità di misura sono sovrapponibili **solo** a condizione che la *comunicazione seriale* sia *binaria*, cioè sostenuta da segnali a **2 livelli** di tensione, cosa per altro largamente diffusa.

- Questa situazione è tuttora valida per definire la velocità del collegamento **DTE-DCE** ***tra computer e modem***, cioè quella della **porta seriale** e dei segnali RS232 generati dall'**UART**: in questo caso si può propriamente parlare di velocità espressa in **baud**, detta anche ***velocità di terminale***, consapevoli che quella espressa in **bps** è numericamente uguale; i suoi valori tipici sono **2,4K**, **9,6K**, **19,2K**, **38,4K**, **57,6K** e **115,2K** [dove **K** significa chilo**baud** (o chilo**bit**) al secondo].
- Il problema nasce nel collegamento **DCE-DCE** ***tra modem e modem***, dove è corretto esprimere la velocità, detta ***velocità di linea***, in entrambi i modi, ma con significati e valori decisamente diversi; questo fatto aggiunge ulteriore confusione e rende necessario un supplemento d'indagine...
- **In primo luogo** va tenuto ben presente che la **massima velocità** di *comunicazione* dipende dalle *caratteristiche* del *mezzo* attraverso il quale avviene la *trasmissione* (detto *canale*); essa è influenzata da 2 fattori:
  - la presenza di rumore (elettronico) o di interferenze (misurata dal ***rapporto segnale-disturbo***)
  - la differenza tra la frequenza più alta e e quella più bassa che il *canale* può portate (detta ***larghezza di banda***)
- La *larghezza di banda* del canale si può in questo caso ritenere uguale alla *massima frequenza* che il *segnale seriale* può assumere, cioè al numero di *periodi* di clock rilevabili sulla linea ogni secondo, coerentemente misurabile in **baud**.

- **In secondo luogo** va aggiunta la necessità di **ottimizzare il canale**, che ha portato ad associare **più di 2** *livelli di tensione* ad ogni *periodo di clock* del *segnale seriale*, rendendo necessario, per codificarne il valore, più di un **bit** per ogni *periodo*.
- Risulta evidente che la *velocità* (numero di *periodi* al secondo) del *segnale seriale* si potrà ancora misurare in **baud**, mentre la *velocità* (numero di *bit* al secondo) del modem dovrà esser misurata in **bps** e sarà certamente *multipla* della prima; per esempio:
  - supponendo che il segnale possa assumere **4** diversi *livelli* di tensione la sua *conversione digitale* richiede la necessità di **2** bit: **00** per il *livello* V_0, **01** per V_1, **10** per V_2 e **11** per V_3; per questo la velocità sarà di **1000 baud** (1000 periodi al secondo) ma anche di **2000 bps** (**2** bit per periodo, 2000 bit al secondo)
  - analogamente, per *esprimere* **8** diversi *livelli* di tensione sono necessari **3** bit: **000** per V_0, **001** per V_1, **010** per V_2, **011** per V_3, **100** per V_4, **101** per V_5, **110** per V_6 e **111** per V_7; per questo la velocità sarà ancora di **1000 baud** (1000 periodi al secondo) ma anche di **3000 bps** (**3** bit per periodo, 3000 bit al secondo)

5 - Standard RS232 - Velocità Seriale: Unità di misura e tecniche

5.3 Modulazione dei segnali: effetto sulla Velocità

- Rimane da stabilire come sia possibile affidare *contemporaneamente* più di una tensione (o più di un bit) ad un *segnale seriale*; questa opportunità è legata alla *natura* dei **modem**, il cui compito è quello di *affidare* i *livelli digitali* in partenza a *segnali sinusoidali* (*toni*), facilmente trasportabili attraverso il *mezzo della trasmissione* (di solito una linea telefonica *vocale*) tra due modem (**DCE**), con la *tecnica della* ***mod****ulazione*; il modem in ricezione *recupera* poi i *livelli digitali* di partenza con la tecnica opposta, ***dem****odulazione*, dai *segnali modulati* in arrivo.
- Certamente i dettagli sulla *modulazione* non sembrano necessari, in questa trattazione, anche perchè il nostro obiettivo non è quello di descrivere il funzionamento dei modem; di fatto l'informazione presente nei *livelli di tensione RS-232C* sono affidati ad una o più caratteristiche (***ampiezza***, ***fase*** o ***frequenza***) della sinusoide portante.

Nella *comunicazione seriale a distanza* tramite *modem* la velocità in **Baud** è la *frequenza* del segnale mentre quella in **bps** è l'effettiva *quantità d'informazione* (velocità) della ricetrasmissione, di norma molto più grande per effetto delle *tecniche di* ***modulazione*** e (come vedremo tra un po') di *compressione dati*.

- Senza entrare in un merito piuttosto complesso, la *modulazione* consente al modem di *codificare* il segnale portante *anche* ***con molti*** *bit* per ogni periodo di clock; in questo modo la ***bit rate*** può superare anche di molto la ***baud rate***.
- A questo proposito sono stati definiti degli standard di protocollo, legati al tipo di modulazione e caratterizzati da una precisa ***bit rate***; valori tipici per la ***velocità di linea***, quella del **modem** tipica del collegamento **DCE-DCE**, sono **14,4K**, **28,8K**, **33,6K** e **56K** [non dimentichiamo che **K** significa chilo**bit** (non chilo**byte**!) per secondo].

Si capisce che la *quantità di informazione* scambiata sulla linea di comunicazione sia fortemente influenzata sia dal **tipo di modulazione** usato sia dal **supporto** sul quale i segnali modulati sono trasportati

- In concreto supponiamo di collegare i ***modem*** alla *linea telefonica*:
  - poichè si tratta di un *canale* ***vocale***, pensato per supportare segnali analogici di frequenza non superiore a **3,1** kHz (quella dei toni più alti di una voce umana) la loro ***baud rate*** non potrà essere superiore a questo valore, ma per motivi tecnici sarà ristretta a **2400 baud** (**2,4K**baud)
  - per esempio lo standard ***V.32bis*** usa la *Quadrature Amplitude Modulation* ed è in grado di raggiungere una ***bit rate*** di **14400 bps**; questo non significa che il segnale coinvolto abbia frequenza di **14,4** kHz ma che, utilizzando la massima *baud rate* possibile, ad ogni suo periodo sono affidati **6** bit (2400 **baud** x 6 **bit** = 14400**bps** = **14,4K**bps)
  - Il modem che opera sotto lo standard descritto ha quindi una *velocità* 6 volte maggiore [ma sarebbe più corretto dire: *trasferisce* una quantità di *dati* 6 volte maggiore] rispetto ad un modem a **2400 bps**, anche se la *frequenza* del segnale [*portante*] è la stessa in entrambi i casi, limitata a **2400 baud** dalla *larghezza di banda* della linea telefonica

5 - Standard RS232 - Velocità Seriale: Unità di misura e tecniche

5.4 Compressione dei dati: effetto sulla Velocità

- La velocità dati (***bit rate***) può essere ulteriormente aumentata con le tecniche di **compressione** esercitate dai *moderni modem ad alta velocità* sulla sequenza dei dati da trasmettere.

- In pratica *ogni byte* posto sulla linea dal *modem ne rappresenta* ***più di uno*** di quelli in arrivo dal *computer* al modem stesso, rendendo necessario, a parità di informazione, l'impiego di una velocità *computer-modem* superiore alla velocità *modem-modem* in misura proporzionale al rapporto di compressione:
  - in assenza di compressione la ***baud rate*** tra *computer* e *modem* (**DCE-DTE**) può essere uguale a quella del *canale* **DTE-DTE**
  - con ***compressione***, per esempio la ***V.42bis*** (dal nome dello standard che ne descrive il *metodo*, in grado di ridurre i dati in partenza fino al 25%, cioè con *rapporto di compressione* **4:1**) la *velocità di terminale* ***deve*** essere impostata ad un *valore* ***4 volte*** *più grande* rispetto alla *velocità del canale*
- Riprendendo l'esempio di ***modem*** collegato alla *linea telefonica* (***baud rate*** non superiore a **2400 baud**) in grado di **codificare** secondo lo standard ***V.32bis*** (***bit rate*** 14400**bps**=2400**baud** x 6 **bit**) e di **comprimerli** secondo lo standard ***V.42bis*** (*compressione* **4:1**) la ***baud rate*** di terminale dovrà essere impostata a **57600 baud**.

La velocità della porta seriale (cioè della *tratta computer-modem*) potrà essere **enormemente più grande** di quella del modem (*tratta modem-telefono-modem*) per effetto della ***compressione*** dei dati, in aggiunta all'effetto della ***modulazione*** del segnale seriale *portante*.

- In ogni caso la *velocità* ***di linea*** non può superare la *velocità* ***di terminale***, fino a qualche tempo fa (1999) limitata al massimo di **115,2Kbaud**; le più recenti interfacce seriali sono in grado di supportare anche **230,4Kbaud** e oltre velocità assolutamente a disposizione di hardware meno critici di modem su linee telefoniche standard.

5 - Standard RS232 - Velocità Seriale: Unità di misura e tecniche

5.5 Velocità previste dagli Standard per i Modem

- Come anticipato non intendo approfondire ulteriormente concetti legati più al funzionamento del *modem* che a quello della *porta seriale* con cui è collegato; a partire dalle prossime pagine parleremo esclusivamente di queste ultime e dell'**UART** che le realizza sostanzialmente.
- Solo per chiudere il discorso e tirare le somme del discorso può essere interessante vedere quali sono le *velocità di terminale* consigliate in funzione delle caratteristiche di modulazione e compressione attuate sui dati dal modem; la seguente **Tabella** raccoglie gli ***standard*** più comuni:

Standard	Caratteristiche relative
V.21 (Bell 103)	**300** bps
V.22 (Bell 212A)	**1,2k** bps
V.23	**1,2k** bps / **75** bps
V.22 bis	**2,4k** bps
V.27	fax
V.29	fax
V.32	**4,8k** bps, **9,6k** bps
V.32 bis	**7,2k** bps, **12k** bps, **14,4k** bps
V.34	**28,8k** bps
V.34 +	**33,6k** bps
V.42	tecniche di correzione errori
V.42 bis	tecniche di compressione dati
V.90	**31,2k** bps, **56k** bps

5 - Standard RS232 - Velocità Seriale: Unità di misura e tecniche

5.6 Velocità intrinseca della porta seriale (UART)

- La preziosa analisi delle pagine precedenti ci ha fatto capire che la *velocità seriale* è in sostanza il numero di *bit* che attraversano la linea ogni secondo (**bps**, ***bit rate***), anche se fisicamente deve essere intesa come *frequenza* del *segnale seriale* (**baud**, ***baud rate***), cioè come numero di *periodi* di clock rilevabili sulla linea ogni secondo.
- Nell'ambito dello standard **RS232** le due definizioni coincidono, poichè esso prevede di affidare **2 soli** *livelli di tensione* a ciascun *periodo di clock* del *segnale seriale* e di codificarli con **un solo bit**.

La ***velocità*** di una ***porta*** (*interfaccia*) ***seriale***, tipica del collegamento **DTE-DCE** *tra computer e modem*, è misurata in **baud** (anche se può essere espressa con lo stesso valore in **bps**); si tratta della frequenza del *segnale seriale* generato dall'**UART**.

- Lo standard **RS232** originale prevede ***velocità*** di 75 e 110 baud (usate con le vecchie telescriventi meccaniche) e poi 300, 600, 1200, 2400, 4800, 9600, fino a quella massima di **19200 baud** (=**19,2 kb/s**=**19200 bps**), da scegliere in funzione del *mezzo della trasmissione* e della distanza da coprire
- La *velocità* in ***caratteri*** (*bytes*) ***al secondo*** è comunque 10 volte più piccola, per la presenza di almeno 2 bit di sincronismo.
- La velocità di ricetrasmissione è completamente programmabile; tutti i valori indicati sono resi possibili dalla presenza del **quarzo** di cui è dotata la ***porta seriale*** di un PC, in grado di oscillare alla frequenza di **1,84320 MHz**:
  - il valore massimo della scala è garantito da un ***prescaler*** (un ***divisore per 16***) interno all'**UART**: per questa ragione a livello hardware la velocità massima raggiungibile è di **115,2 kbaud** (1843200/16 = 115200 Hz = 115,2 kbaud = 115,2 kbps)
  - i rimanenti valori sono ottenuti dividendo questa *velocità di base* per un numero intero, tipicamente da 2 a 2304, con l'aiuto di un ***divisore programmabile***, sempre messo a disposizione dall'**UART**

- La tabella seguente (generata da un mio piccolo eseguibile, **Prn2Com.COM**) mette in evidenza che, in pratica, non tutti i *divisori* possibili sono realmente usati; i *fattori di divisione* sono espressi in decimale e in esadecimale e sono accostati alla ***baud rate*** generata (*lorda* e *netta*) e ad una stima di ***distanza*** tra porta e modem.

SERVIZI DEDICATI ALLA GESTIONE DELLE PORTE SERIALI COMx PRESENTI NEL SISTEMA

La TABELLA seguente mostra le effettive velocità, per ricetrasmissione con 2 bit di di STOP e nessun bit di PARITA' e 7 bit di DATO (su 10) o 8 bit di DATO (su 11) insieme alla distanza che si può coprire senza perdita di dati

Baud Rate	Effettiva 7 bit	Effettiva 8 bit	Dist. metri	fattore di divisione	Baud Rate	Effettiva 7 bit	Effettiva 8 bit	Dist. metri	fattore di divisione
50	35	36	4800	2304=0900H	1986	1390	1444	121	58=003AH
75	52	54	3200	1536=0600H	2400	1680	1745	100	48=0030H
110	77	80	2182	1047=0417H	3600	2520	2618	67	32=0020H
134	94	98	1791	857=0359H	4800	3360	3491	50	24=0018H
150	105	109	1600	768=0300H	7200	5040	5236	33	16=0010H
300	210	218	800	384=0180H	9600	6720	6982	25	12=000CH
600	420	436	400	192=00C0H	19200	13440	13964	14	6=0006H
1200	840	873	200	96=0060H	38400	26880	27927	6.5	3=0003H
1800	1260	1309	160	64=0040H	57600	40320	41891	4	2=0002H

[PgUp] [PgDn] Scorre le Informazioni [3/3] [ESC] MENU precedente

"Prn2Com" - Versione 1.0 - Copyright febbraio 1996, Giorgio OBER

- La tabella è del tutto indicativa: possiamo osservare che sono possibili anche valori di ***baud rate*** non previsti dallo standard originale, ed altri, nemmeno inclusi nella tabella, potrebbero essere richiesti da qualche tipo di modem.
- Nella parte dedicata ai **Registri** dell'**UART** vedremo come *programmare* la desiderata velocità di terminale.

6 - Standard RS232 - Linee e Segnali dello Standard RS232

6.1 Linee e Segnali dello Standard RS232

- Nelle pagine iniziali ho anticipato che, nella *comunicazione seriale* tra *computer* e *modem* (**DTE-DCE**), sono necessarie numerose linee specializzate nel *controllo del flusso dati* (*hardware* ***handshaking***).
- Il compito di questi segnali è quello di ottimizzare lo scambio dei dati, riducendo il più possibile ogni tempo morto; ciascuno di essi porta un nome piuttosto *bizzarro*, *arcaico* e *scarsamente mnemonico*, pensato molti anni or sono per tentare di evidenziare il ruolo da essi esercitato nell'ambito *seriale* **DTE-DCE**.
- Sebbene questi segnali siano sostanzialmente inutili nell'uso dell'***interfaccia*** (***porta***) ***seriale*** con dispositivi *diversi* dai *modem* (praticamente tutti i nostri futuri progetti) sembra corretto dedicare loro un piccolo spazio.

In molte parti di questa sezione ci siamo riferiti (e ci riferiremo) alle 2 estremità di un *collegamento seriale* con i termini *computer* (**DTE**) e *modem* (**DCE**) dando per scontato che sarebbe più corretto parlare di ***interfaccia*** (o ***porta***) ***seriale*** collegata a *computer* (**DTE**) e *modem* (**DCE**)

- L'*interfaccia* ***seriale*** (o più precisamente l'**UART**), presente ad entrambe le 2 estremità del collegamento, attiva i segnali necessari nei tempi e nei modi previsti dal ***protocollo*** *di handshaking*, per certi versi paragonabile a quello attivato dall'interfaccia parallela ***Centronics*** nella gestione dei caratteri verso una stampante.

6 - Standard RS232 - Linee e Segnali dello Standard RS232

6.2 Linee e Segnali RS232: Tabella sintetica

- Prima di affrontare questo meccanismo è utile descrivere uno per uno i segnali coinvolti; la **Tabella riassuntiva** li raccoglie tutti, associandoli alla parte del collegamento che li genera e specificandone il verso comunque dal punto di vista del computer (**DTE**); il loro nome simbolico è quello previsto dagli standard **EIA** (Electronic Industry Association) **RS232C** e **CCITT** (Consultative Committee on International Telephone and Telegraph) **V.24/V.28**, tra loro equivalenti; quelli in **verde** sono i più importanti:

EIA RS232C	CCITT V.24	NOME	Segnali comuni	NOME	CCITT V.24	EIA RS232C
AA	101	FG	Frame Ground [Protective Ground]	FG	101	AA
AB	102	SG	**Signal Ground**	SG	102	AB

DTE OUT			**dal** [computer] **al** [modem]	**al** [computer] **dal** [modem]	DTE IN		
EIA RS232C	CCITT V.24	NOME	Descrizione	Descrizione	NOME	CCITT V.24	EIA RS232C
BA	103	TD	**Transmitted Data**	**Received Data**	RD	104	BB
CA	105	RTS	**Request To Send**	**Clear To Send**	CTS	106	CB
CD	108.2	DTR	**Data Terminal Ready**	**Data Set Ready**	DSR	107	CC
				Data Carrier Detect	DCD/CD	109	CF
CJ	-	-	Ready for Receiving	**Ring Indicator**	RI	125	CE
CH	111	DSRS	Data Signal Rate Selector	Data Signal Rate Selector	DSRS	112	CI
				Signal Quality Detector	SQ	110	CG
DA	113	ETC	Transmitter Signal Element Timing	Transmission Signal Element Timing	TC	114	DB
-	141	LL	Local Loopback	Receiver Signal Element Timing	RC	115	DD
				Test Mode	TM	142	-
SBA	118	STD	Secondary Transmitted Data	Secondary Received Data	SRD	119	SBB
SCA	120	SRTS	Secondary Request To Send	Secondary Clear To Send	SCTS	121	SCB
				Secondary Data Carrier Detect	SDCD	122	SCF

- La *prima lettera* della definizione dei segnale, secondo lo standard **EIA**, consente una prima distinzione in 5 gruppi funzionali: **A**=massa, **B**=dati, **C**=controllo, **D**=temporizzazione e **S**=canale secondario
- La **Tabella** mostra **tutti i segnali** previsti dagli *standard*; essi possono essere organizzati nelle **6 tipologie** proposte qui dio seguito, una dopo l'altra.

NB: ciascuno dei *segnali* previsti dagli *standard seriali* è presentato con il suo **nome** (e **sigla**), la sua *descrizione sintetica*, l'indicazione di *direzione* (dal punto di vista della *porta seriale* del computer, **DTE**) e la *posizione* (*pin*) nei possibili connettori **DB25** e **DB9**; si sottolinea inoltre che segnale ***attivo*** significa (**ON**, **0** logico, **SPACE**, **+12V**) e segnale ***non attivo*** significa (**OFF**, **1** logico, **MARK**, **-12V**)

6 - Standard RS232 - Linee e Segnali dello Standard RS232

6.3 Linee RS232 di riferimento di massa

Riferimento di massa [2 linee, EIA di tipo **A**]: in ogni rapporto tra dispositivi è necessario condividere un *livello di tensione di riferimento comune* (*massa*) e, nei casi più raffinati, una *massa di protezione*.

Signal Ground (SG)	Massa di segnale	[ ]	[pin**7**/DB25]	[pin**5**/DB9]

ognuno dei segnali descritti nelle altre tipologie prevede di utilizzare *questa* linea come *percorso di ritorno*, detta per questa ragione ***massa** di segnale*; è evidente che, qualora venisse a mancare, non è possibile alcuna comunicazione; può essere connesso alla *massa protettiva*, descritta tra poco, anche se come si può notare, il connettore standard lo prevede *separato*. La presenza di questa *massa comune* rende lo standard **RS232** diverso da altri, con tensione differenziale e bilanciata, in grado di assicurare una maggiore immunità al rumore.

Protective Ground (FG)	Massa di contenitore	[ ]	[pin**1**/DB25]	[- /DB9]

questa linea ***può*** essere collegata all'involucro metallico del *connettore* e quindi al *contenitore* (*chassis*) del dispositivo seriale; in questo modo i contenitori delle 2 apparecchiature collegate tra loro dal cavo seriale hanno lo stesso potenziale, allo scopo di proteggere il collegamento da scariche possibili elettriche e interferenze. In questa operazione bisogna essere ben certi di non creare ***anelli di massa*** per cui è buona norma evitare di collegare la ***massa** di chassis* alla ***massa** di segnale* e, in caso di *cavo seriale schermato*, collegare ***solo*** da una parte la *calza metallica* cui vengono avvolti i conduttori: la presenza di segnali alternati sui conduttori del cavo può indurre nell'***anello di massa*** una differenza di potenziale in grado di produrre una corrente elettrica (anche piuttosto elevata, data la piccolissima impedenza dell'anello) in grado di bruciare i componenti elettrici dei 2 dispositivi. Lo stesso vale per eventuali scariche elettriche, magari dovute a temporali, per cui il collegamento tra questi pin va fatto con molta attenzione.

6 - Standard RS232 - Linee e Segnali dello Standard RS232

6.4 Segnali RS232 del Canale di comunicazione Primario

Canale di comunicazione primario [4 linee]: a questo gruppo appartengono i 2 segnali principali [EIA di tipo **B**] della comunicazione seriale e i 2 segnali di controllo di flusso [EIA di tipo **C**] necessari per coordinare la trasmissione dati tra **DTE** e **DCE**.

Transmitted Data (TD)	Dati in uscita	[OUT=DTE>DCE]	[pin**2**/DB25]	[pin**3**/DB9]

su questa linea (detta anche **TxD**) sono posti i bit del *dato parallelo* forniti dal processore e formattati dall'**UART** nel *frame seriale* previsto dallo standard **RS232**; se, in uscita dal **DTE**, non viene posto alcun dato, la linea rimane ***non attiva***

Received Data (RD)	Dati in ingresso	[IN = DCE>DTE]	[pin**3**/DB25]	[pin**2**/DB9]

su questa linea (detta anche **RxD**) arrivano i bit del *frame seriale* formattati dall'**UART** del trasmettitore, dall'altra parte del cavo seriale, nella sequenza suggerita dallo standard **RS232**; se, in ingresso al **DTE**, non è presente alcun dato, la linea rimane ***non attiva***

Request To Send (RTS)	Richiesta collegamento/DTE pronto a trasmettere	[OUT=DTE>DCE]	[pin**4**/DB25]	[pin**7**/DB9]

questo segnale è ***attivato*** in uscita dal **DTE** per *chiedere* (letteralmente) alla periferica *di collegarsi*, cioè (in sostanza) per avvisare il dispositivo **DCE** che *dispone di dati* ed è *pronto a trasmetterglieli*, così che esso possa predisporre i suoi circuiti a riceverli; quando il **DCE** è in condizioni di *accettare i dati* provvederà ad attivare a sua volta la linea **CTS** (vedi sotto) per segnalare la sua disponibilità a ricevere: solo in questo caso il computer comincerà a trasmettergli dati. In condizioni di riposo questa linea è tenuta a **1** (***non attiva***) consentendo in questo modo una eventuale trasmissione dati dal **DCE** verso se stesso

Clear To Send (CTS)	DCE Pronto a ricevere/DTE può trasmettere	[IN = DCE>DTE]	[pin**5**/DB25]	[pin**8**/DB9]

questo segnale è ***attivato*** in uscita dal **DCE** per avvisare il dispositivo **DTE** che è *pronto a ricevere* i suoi dati, della disponibilità dei quali era stato avvisato dal passaggio da OFF a ON della sua linea **RTS** (vedi sopra); in altre parole il modem **DCE** informa il computer **DTE** che *la trasmissione può cominciare* (o *continuare*). In condizioni di riposo questa linea è tenuta a **1** (***non attiva***) impedendo in questo modo ogni trasmissione dati dal **DTE** verso se stesso.

6 - Standard RS232 - Linee e Segnali dello Standard RS232

6.5 Segnali RS232 del Canale di comunicazione Secondario

Canale di comunicazione secondario [4 linee]: a questo gruppo 4 segnali funzionalmente identici a quelli del gruppo precedente, ma destinati alla *comunicazione seriale* e al *controllo del flusso dati* di un eventuale canale secondario, usato in caso di errori sul canale primario, o per scopi diagnostici, o per controllare il flusso dati nel canale primario; di solito è caratterizzato da baud rate più lente, per aumentare l'affidabilità della ricetrasmissione.

Secondary Transmitted Data (STD)	Dati in uscita	[OUT=DTE>DCE]	[pin14/DB25]	[/DB9]

è la linea sulla quale sono posti i dati seriali in uscita dal **DTE** (canale secondario) verso il **DCE**

Secondary Received Data (SRD)	Dati in ingresso	[IN = DCE>DTE]	[pin16/DB25]	[/DB9]

è la linea sulla quale sono ricevuti i dati seriali in uscita dal **DCE** (canale secondario) verso il **DTE**

Secondary Request To Send (SRTS)	Richiesta collegamento/DTE pronto a trasmettere	[OUT=DTE>DCE]	[pin19/DB25]	[/DB9]

questa linea è usata in uscita dal **DTE** per *chiedere* al **DCE** *di collegarsi* sul canale secondario, poichè *dispone di dati* ed è *pronto a trasmetterglieli*

Secondary Clear To Send (SCTS)	DCE Pronto a ricevere/DTE può trasmettere	[IN = DCE>DTE]	[pin13/DB25]	[/DB9]

questa linea è usata in uscita dal **DCE** per awisare il dispositivo **DTE** che è *pronto a ricevere* i suoi dati, cioè per informarlo che *la trasmissione può cominciare* (o *continuare*) sul canale secondario

6 - Standard RS232 - Linee e Segnali dello Standard RS232

6.6 Segnali RS232 di Stato e del Canale

Segnali di stato del Modem e del Canale [5+1 linee](prima parte): le linee di questo gruppo consentono di monitorare lo *stato* del modem ed assicurano dei punti intermedi per il controllo del suo funzionamento durante la ricetrasmissione sul canale telefonico.

Data Terminal Ready (DTR)	DTE connesso/DTE pronto a comunicare	[OUT=DTE>DCE]	[pin20/DB25]	[pin4/DB9]

quando il **DTE** è *acceso* e nel *corretto modo* di funzionamento ***attiva*** questo segnale per *avvisare* il dispositivo **DCE** che è *regolarmente collegato alla linea di comunicazione* ed è *pronto ad aprire un canale di comunicazione* (cioè a trasmettere o ricevere dati). Questo segnale dichiara la presenza (lo *stato*) attiva del computer **DTE**, cioè la disponibilità per un *collegamento possibile*, mentre **Request To Send** (che può sembrare analogo) dichiara la sua disponibilità di dati da trasmettere, cioè la presenza di un *collegamento già stabilito*; questo segnale è importante soprattutto se il dispositivo **DCE** è un modem: quando è attivato prowede alla sua connessione al circuito telefonico e lo mantiene collegato fino alla sua disattivazione; con altri dispositivi può anche non essere indispensabile. Il computer **DTE** deve *attivare* questo segnale ***prima*** che il modem **DCE** possa attivare la linea **DSR** (vedi sotto) per segnalare a sua volta la sua presenza sulla *linea di comunicazione*.

Data Set Ready (DSR)	DCE connesso/DCE pronto a comunicare	[IN = DCE>DTE]	[pin6/DB25]	[pin6/DB9]

quando il modem **DCE** è *acceso* e nel *corretto modo* di funzionamento ***attiva*** questo segnale per *avvisare* il computer **DTE** che è *regolarmente collegato alla linea di comunicazione* ed è *pronto a concorrere sul canale di comunicazione* (cioè a trasmettere o ricevere dati). Questo segnale dichiara la presenza (lo *stato*) attiva del modem **DCE**, cioè la disponibilità per un *collegamento possibile*, mentre **Clear To Send** (che può sembrare analogo) dichiara la sua disponibilità a ricevere, cioè la presenza di un *collegamento già stabilito*; questo segnale è importante soprattutto se il dispositivo **DCE** è un modem: solo quando è attivato *esso* sarà ritenuto *a disposizione* dal computer che, in precedenza, lo aveva sollecitato attivando la linea **DTR** (vedi sopra). Con altri dispositivi può anche non essere indispensabile, ma è buona norma forzarlo permanentemente ***attivo*** con un collegamento a **+12V** dentro l'apparecchiatura **DCE** o con opportuno ponticello nel cavo di collegamento, dato che il computer **DTE** non trasmetterà alcun dato fino a quando non rileverà un **DSR** attivo in arrivo dal **DCE**.

Ring Indicator (RI)	Ricevuto segnale acustico sul canale	[IN = DCE>DTE]	[pin22/DB25]	[pin9/DB9]

questo segnale è ***attivato*** in uscita dal **DCE** per awisare il computer **DTE** che sul canale di comunicazione è in atto una chiamata entrante, cioè che sulla linea telefonica si sta ricevendo un *segnale acustico* a bassa frequenza e ad alta tensione (che fa suonare il campanello del telefono). Questo segnale è mantenuto attivo per tutta la durata del *segnale sonoro* ed è significativo solo se il dispositivo **DCE** è un modem; in ogni caso la sua considerazione è ininfluente sulla gestione dei dati.

Data Signal Rate Selector (DSRS)	Selettore di velocità	[OUT=DTE>DCE]	[pin23/DB25]	[- /DB9]

questo segnale è usato per selezionare una delle *2 velocità* (se il collegamento è *sincrono*) o una di *2 serie di velocità* (se il collegamento è *asincrono*) predisposte sul dispositivo opposto a quello che lo genera, di solito il computer **DTE** nei confronti del modem **DCE** (ma anche viceversa); se è ***attivato*** risulta selezionata la *baud rate* più alta.

Data Carrier Detect (CD)	Rilevato modem remoto/possibile comunicare	[IN = DCE>DTE]	[pin8/DB25]	[pin1/DB9]

questo segnale (detto anche **Received Line Signal Detector**) è ***attivato*** in uscita dal **DCE** per avvisare il computer **DTE** che ha riconosciuto la presenza del modem (**DCE** remoto) *dall'altra parte* della linea telefonica, o meglio ha rilevato la presenza sulla linea della *portante* (*carrier*) sulla quale il **DCE** remoto ha *modulato* i suoi dati. La presenza di questo *segnale analogico modulato* (e quella conseguente del segnale **CD *attivo***), avvisa il computer **DTE** che si sta ricevendo un *segnale di buona qualità*, che il canale di comunicazione è *pulito, poco rumoroso* e che, quindi, è possibile scambiare dati con profitto. Questo segnale è mantenuto attivo (dal modem **DCE** locale) per tutto il tempo in cui viene rilevato il *tono di risposta* dal modem **DCE** remoto. Quando viene disattivato o la linea ha subito una rottura (*assenza* di portante) o è troppo disturbata (portante *di scarsa qualità*) per consentire la demodulazione dei dati in arrivo; in entrambi i casi non ha senso mantenere il collegamento.

Secondary Data Carrier Detect (SCD)	Rilevato modem remoto/possibile comunicare	[IN = DCE>DTE]	[pin12/DB25]	[- /DB9]

questo segnale (detto anche **Secondary Received Line Signal Detector**) esercita, per il canale di comunicazione secondario, la stessa funzione esercitata dal segnale **Data Carrier Detect** (vedi sopra), essendo ***attivo*** quando il modem **DCE** locale ha rilevato la presenza sulla linea secondaria di un *segnale portante* (*carrier*) *di buona qualità*, sintomo della presenza fattiva di un modem **DCE** remoto.

5 - Standard RS232 - Linee e Segnali dello Standard RS232

6.7 Segnali RS232 per la diagnostica del Canale

Segnali per la diagnostica del canale [3+3 linee]: la presenza di questi segnali non è normale: in certe condizioni può essere utile sottoporre il canale di comunicazione ad alcuni test d'integrità, per esempio per verificare la velocità massima che il canale può sostenere. In questi casi potremo coinvolgerli per una migliore messa a punto, prima di dar luogo allo scambio di dati.

Local Loopback (LL)	Ricetrasmissione locale su se stessa	[OUT=DTE>DCE]	[pin18/DB25]	[/DB9]

questo segnale è ***attivato*** in uscita dal **DTE** per avvisare il modem **DCE** che dovrà predisporsi per lo *stato di prova*, cioè collegare per motivi diagnostici la sua linea d'uscita alla sua linea d'ingresso; in questo modo il suo segnale modulato d'uscita, di solito diretto verso la linea telefonica, viene ridiretto verso il suo ricevitore, consentendo al computer **DTE** di verificare se i circuiti del modem sono in buono stato, cioè di verificare se il dato ricevuto è quello appena trasmesso. Da notare che il modem ***attiva*** il suo segnale **Test Mode** (vedi sotto) per far capire al computer **DTE** che sta lavorando in condizione di loopback.

Signal Quality Detector (SQ)	Controllo di qualità sui dati ricevuti	[IN = DCE>DTE]	[pin21/DB25]	[/DB9]

questo segnale è ***attivato*** in uscita dal modem **DCE** per confermare al computer **DTE** che i dati ricevuti sono attendibili; se il **DCE** ritiene alta la *probabilità d'errore* lascia la linea ***non attiva***, per esempio per richiedere automaticamente ana ritrasmissione dei dati.

Test Mode (TM)	Riconoscimento comando Local Loopback	[IN = DCE>DTE]	[pin25/DB25]	[/DB9]

questo segnale è ***attivato*** in uscita dal modem **DCE** per avvisare il computer **DTE** che il suo comando **Local Loopback** (vedi sopra) è stato ricevuto e messo in atto, cioè che si è predisposto per lo *stato di prova* ed ha collegato per motivi diagnostici la sua linea d'uscita alla sua linea d'ingresso.

Reserved (+P)	Tensione positiva di prova	[ ]	[pin9/DB25]	[/DB9]

Se, per scopi diagnostici, è necessaria una tensione positiva di prova, questo piedino è tenuto a **+ 12V**

Reserved (-P)	Tensione negativa di prova	[ ]	[pin10/DB25]	[/DB9]

Se, per scopi diagnostici, è necessaria una tensione negativa di prova, questo piedino è tenuto a **- 12V**

Unassigned (-)	non usato	[ ]	[pin11/DB25]	[/DB9]

di norma non utilizzato, può essere coinvolto per scopi diagnostici

6 - Standard RS232 - Linee e Segnali dello Standard RS232

6.8 Segnali RS232 per la temporizzazione del Canale

Segnali per la temporizzazione del canale [3 linee]: questi segnali sono usati ***solamente*** con *protocollo* di comunicazione *sincrono* e sono necessari per assicurare i *segnali di temporizzazione* necessari al trasmettitore e al ricevitore; il *protocollo asincrono* standard non ne ha bisogno.

Transmission Signal Element Timing (TC)	Clock DCE di trasmissione dati	[IN = DCE>DTE]	[pin**15**/DB25]	[/DB9]

questo segnale (detto anche **Transmission Clock**) è significativo **solo se** il dispositivo **DCE** è un modem operante con protocollo **sincrono**; si tratta di un'*onda quadra* (*clock*) generata dal modem **DCE** per *imporre* e *controllare con precisione* (*sincronizzare*) la velocità di trasmissione del computer **DTE**, obbligandolo a mettere ciascun *bit* del dato in uscita (sulla sua linea **Transmitted Data**) esattamente quando riceve il *fronte di discesa* del segnale, cioè durante la transizione da **1** a **0** logico (da OFFa ON, da **SPACE** a **MARK**, da **-12V** a **+12V**). In questo modo ciascun *bit* del dato in uscita è mantenuto sulla linea per un *intero periodo di clock* e la presenza del *fronte di salita* (transizione **0>1** logico, ON>OFF, **MARK>SPACE**, **+12V>-12V**) sta ad indicare indicare la sua posizione centrale.

Transmitter Signal Element Timing (ETC)	Clock DTE di trasmissione dati	[OUT=DTE>DCE]	[pin**24**/DB25]	[/DB9]

questo segnale (detto anche **External Transmitter Clock**) è utilizzato quando il modem **DCE** non è in grado di assicurare la presenza degli altri due, **TC** e **RC** (vedi sopra e sotto), oppure essi non sono in uso; in questo caso spetta al computer **DTE** il compito di *imporre* (*sincronizzare*) la velocità di trasmissione dati del modem **DCE**.

Receiver Signal Element Timing (RC)	Clock DCE di ricezione dati	[IN = DCE>DTE]	[pin**17**/DB25]	[/DB9]

questo segnale (detto anche **Receiver Clock**) è significativo **solo se** il dispositivo **DCE** è un modem operante con protocollo **sincrono**; si tratta di un'*onda quadra* (*clock*) generata dal modem **DCE** per *imporre* e *controllare con precisione* (*sincronizzare*) la velocità di ricezione del computer **DTE**, obbligandolo a rilevare ciascun *bit* del dato in ingresso (sulla sua linea **Received Data**) esattamente quando riceve il *fronte di discesa* del segnale, cioè durante la transizione da **1** a **0** logico (da OFFa ON, da **SPACE** a **MARK**, da **-12V** a **+12V**). In questo modo ciascun *bit* del dato in ingresso è mantenuto sulla linea per un *intero periodo di clock* e potrà essere testato in presenza del *fronte di salita* (transizione **0>1** logico, ON>OFF, **MARK>SPACE**, **+12V>-12V**), cioè esattamente nella sua posizione centrale.

7 - Standard RS232 - Protocolli di Controllo del Flusso dei Dati Seriali

7.1 Protocolli di Controllo del Flusso Dati: Premessa

- Per il *controllo completo* del ***flusso dati*** tra *computer* **DTE** e *modem* **DCE** sono effettivamente necessari **solo 8 segnali** (più la massa), **riducibili a 2** (più la massa) se si desidera utilizzare la linea seriale per collegare 2 *dispositivi* **DTE** (come faremo noi, nella parte applicativa).
- Per spiegare cosa succede quando 2 *stazioni seriali*, locale e remota, sono collegate tra loro *via telefonica* ricordiamo brevemente i **9 segnali necessari** (si noti come siano elencati tutti e soli quelli presenti sul connettore **DB9**):

Data Carrier Detect (CD)	Rilevato modem remoto/possibile comunicare	[IN = DCE>DTE]	[pin**8**/DB25]	[pin**1**/DB9]
Received Data (RD)	Dati in ingresso	[IN = DCE>DTE]	[pin**3**/DB25]	[pin**2**/DB9]
Transmitted Data (TD)	Dati in uscita	[OUT=DTE>DCE]	[pin**2**/DB25]	[pin**3**/DB9]
Data Terminal Ready (DTR)	DTE connesso/DTE pronto a comunicare	[OUT=DTE>DCE]	[pin**20**/DB25]	[pin**4**/DB9]
Signal Ground (SG)	Massa di segnale	[ ]	[pin**7**/DB25]	[pin**5**/DB9]
Data Set Ready (DSR)	DCE connesso/DCE pronto a comunicare	[IN = DCE>DTE]	[pin**6**/DB25]	[pin**6**/DB9]
Request To Send (RTS)	Richiesta collegamento/DTE pronto a trasmettere	[OUT=DTE>DCE]	[pin**4**/DB25]	[pin**7**/DB9]
Clear To Send (CTS)	Richiesta collegamento/DTE pronto a trasmettere	[IN = DCE>DTE]	[pin**5**/DB25]	[pin**8**/DB9]
Ring Indicator (RI)	Ricevuto segnale acustico sul canale	[IN = DCE>DTE]	[pin**22**/DB25]	[pin**9**/DB9]

- Alcuni di essi (**RI** e **CD**) stabiliscono *la presenza* sulla *linea telefonica* del *modem* remoto; altri (**DTR** e **DSR**) stabiliscono *la disponibilità* del *modem* e del *computer* locale a concorrere alla comunicazione seriale sulle linee **RD** e **TD**; se il collegamento seriale non coinvolge la *linea telefonica* i segnali **DTR**, **DSR**, **RI** e **CD** non sono necessari.

- La necessità dei 2 segnali rimanenti (**CTS** e **RTS**) è legata al fatto che la *linea telefonica* (tra *modem* locale e *modem* remoto) è generalmente **più lenta** della *linea seriale* (tra *computer* locale e *modem* locale); abbiamo già evidenziato che, mentre i modem possono comunicare sulla *linea telefonica* a bassa *velocità* [pur ottimizzata con le tecniche di modulazione e compressione a valori come **33,6K**bps o **28,8K**bps (**V.34**) o **56K**bps (**V.90**)], la *linea seriale* può essere spinta a quella massima (**115200** bps).

- Standard RS232 - Protocolli di Controllo del Flusso dei Dati Seriali

7.2 Protocolli Hardware per il Controllo del Flusso Dati

- E' chiaro che, dal *collo di bottiglia* della *linea telefonica*, possono uscire molti meno dati di quelli forniti dalla *linea seriale*; per prevenire la *perdita di dati* è dunque necessaria la presenza di **buffer interni** e di un **meccanismo** (***protocollo***) **di *controllo*** del ***flusso dati***:
  - il buffer è un'***area di memoria*** in cui è possibile raccogliere una discreta quantità di dati in transito; quello più importante (qualche Kbyte) è disponibile nel *modem* (inteso come dispositivo), sebbene uno di piccole dimensioni (qualche byte) sia presente anche nell'**UART** della *porta seriale* ad esso associata
  - il *computer* prowede *velocemente* ad *erogare dati* verso il buffer e, quando è pieno, sarà momentaneamente bloccato dal *modem*, per consentirgli di vuotarlo in linea, *in tempi telefonici*
  - solo quando i dati del buffer stanno per essere smaltiti completamente il *modem* autorizzerà il *computer* a riempire di nuovo il buffer, alla *massima velocità*
- La descrizione si riferisce al *controllo* del ***flusso dati*** in **uscita** dal *computer* locale verso il *modem* locale; se esso è affidato ai segnali **CTS** e **RTS** (in *half–duplex*) si parla di ***hardware handshake***, per sottolineare che può essere realizzato anche sotto il controllo di *programmi seriali* (***software handshake***), con l'aiuto di speciali caratteri con compiti analoghi a quelli delle linee fisiche.
- Naturalmente può essere operata anche per il *controllo* del ***flusso dati*** in senso opposto, in **ingresso** al *computer* proveniente dal *modem*: data la maggiore velocità della *linea seriale* (*computer* locale) rispetto a quella della *linea telefonica* (*modem* locale), la presenza dei buffer *sembra* non essere necessaria.
- In realtà è ancora utile per ridurre le perdite di dati in arrivo quando il *computer è* ***occupato*** in altre mansioni (per esempio spostare dati sull'hard disk): in questo caso il *modem* locale deve bloccare il flusso dati in uscita dal *modem* remoto.
- Vediamo in dettaglio come si sviluppa il ***protocollo seriale*** previsto dallo standard **RS232** per il ***flusso dati*** in **uscita** dal *computer*.
  - il *modem* locale è in attesa di rilevare la presenza di un *segnale acustico* (*chiamata entrante*) sulla linea telefonica, generato dal *modem* remoto per chiedere di accedere al canale di comunicazione
  - non appena ciò awiene il *modem* locale spedisce il segnale **RI** al *computer* locale
  - in risposta alla ricezione del segnale **RI** il *computer* locale *attiva* il segnale **DTR**, per *avvisare* il *modem* locale che è *pronto a comunicare*
  - quando il *modem* locale riceve il segnale **DTR**, consapevole che il *computer* locale è pronto, si mette in attesa che il *modem* remoto si faccia riconoscere attivando la sua *portante*, il *segnale analogico modulato* sul quale il *modem* remoto ha *modulato* i suoi dati
  - non appena ha rilevato sul canale di comunicazione la presenza della *portante* del *modem* remoto, purchè *di buona qualità*, il *modem* locale spedisce il segnale **CD** al *computer* locale, per awisarlo che il *segnale* è *pulito* e il canale è *poco rumoroso* e che, quindi, è possibile scambiare dati con profitto
  - il *modem* locale mette a punto il collegamento e, quando è in condizioni di trasmettere dati lungo la *linea telefonica*, *attiva* il segnale **DSR**, per *avvisare* il *computer* locale che, a sua volta, è *pronto a comunicare*
  - in queste condizioni la comunicazione può avere inizio nei due sensi: il *modem* locale trasferisce i bytes dalla *linea telefonica* alla linea **RD**, verso il *computer* locale, oppure quest'ultimo mette i propri bytes sulla sua linea **TD**, incaricando il *modem* locale di spedirli sulla *linea telefonica* verso il *modem* remoto
  - se il *computer* locale *dispone di dati* dichiara al *modem* locale di essere *pronto a trasmetterglieli*, *attivando* il suo segnale **RTS**
  - se il *buffer* è vuoto il *modem* locale *attiva* **CTS** per awisare il *computer* locale che è *pronto ad accettarli*; non appena il *buffer* è pieno il *modem* locale interrompe erogazione dei dati in uscita dal *computer*, *disattivando* **CTS**; solo quando ha smaltito la sua dote sulla *linea telefonica* (buffer di nuovo vuoto) *riattiva* **CTS** per awisare il *computer* locale che è *pronto ad accettarne altri*
  - quando il *modem* locale non rilevava più la *portante* del *modem* remoto (fine comunicazione o canale disturbato) *disattiva* i segnali **CD**, **DSR** e **CTS**, confermando rispettivamente al *computer* locale la *mancanza di segnale*, l'*indisponibilità a comunicare* e l'*indisponibilità ad accettare dati*
  - non appena il *computer* locale perde il segnale **CD** torna nello stato d'attesa, *disattivando* i segnali **DTR** e **RTS**, per confermare al *modem* locale l'*indisponibilità a comunicare* e l'*indisponibilità a trasmettergli dati*

- Standard RS232 - Protocolli di Controllo del Flusso dei Dati Seriali

7.3 Protocolli Software per il Controllo del Flusso Dati

- Il *controllo* del ***flusso dati*** (*handshake*) tra dispositivi seriali può essere esercitato anche da ***software***, includendo particolari *bytes* tra quelli *di dato*, direttamente sulle linee di comunicazione **TD** e **RD**.

- Il metodo software si rende necessario quando le linee **CTS** e **RTS** sono presenti solo da una parte del collegamento (di solito il *computer* **DTE**), rendendo impossibile l'***hardware handshake***, sebbene esso sia più naturale e più diffuso; spesso il *dispositivo* **DCE**, invece di un *modem* locale, è una stampante o un plotter.

- La scelta è spesso suggerita in fase di installazione dei programmi (driver) di gestione della periferica seriale.

- Il protocollo di *controllo* ***software*** del ***flusso dati*** è detto anche **Xon/Xoff** dal nome dei 2 caratteri usati:

  - non appena il *buffer* è pieno la *periferica seriale* obbliga il *computer* ad interrompe l'erogazione dei dati in uscita verso se stessa, *spedendo* il carattere **Xoff**, di valore **13**H (=**19** decimale), corrispondente al **carattere Ascii di controllo** **DC3** (o **Control-S**, dal **nome dei tasti** che si possono premere per ottenere lo stesso effetto)

  - solo quando ha smaltito i dati a disposizione (*buffer* di nuovo vuoto) la *periferica seriale* avvisa il *computer* locale che è *pronta ad accettarne altri*, *spedendo* il carattere **Xon**, di valore **11**H (=**17** decimale), corrispondente al carattere ascii di controllo **DC1** (o **Control-Q**, dal nome dei tasti che si possono premere per ottenere lo stesso effetto)

- La mancata necessità delle linee addizionali **CTS** e **RTS** rende il *controllo* ***software*** certamente più economico rispetto a quello hardware ma anche soggetto a svantaggi piuttosto evidenti:

  - in caso di ***overrun*** del buffer del ricevitore, cioè se qualche dato non viene scaricato prima dell'arrivo dei successivi, tutta la comunicazione seriale è a rischio se tra i bytes perduti ci sono quelli di *controllo* ***software***: il trasmettitore smetterà per sempre di spedire nuovi dati (nel caso di perdita di **Xon**) oppure continuerà a riversarli in linea (nel caso di perdita di **Xoff**) con effetti disastrosi..

  - la comunicazione seriale sarà più lenta, data l'inevitabile necessità di un grande numero di caratteri (**Xon/Xoff**) in più, tra l'altro da almeno 10 bit ciascuno, per la presenza (come per gli altri) dei 2 bit di start/stop

  - il software si dovrà occupare della ***verifica*** e della ***gestione*** dei caratteri di *controllo* ***software***: in particolare dovrà riservare *particolare attenzione* nell'interpretazione dei caratteri ricevuti, per poter *distinguere con certezza* i 2 *bytes di controllo* da *quelli di dato*, essendo i primi normalmente inseriti tra i secondi ed essendo probabile che tra i dati ci siano bytes uguali a **13**H o a **11**H, assolutamente da non confondere con quelli operativi, **DC1** e **DC3**

In definitiva il *controllo* ***software*** del ***flusso dati*** è applicabile solo la **probabilità di overrun** è **molto bassa**, cioè se la velocità di comunicazione è *bassa* e se la linea è *di buona qualità* (*poco rumorosa*).

Indice Analitico	Prima Parte	Porta Seriale