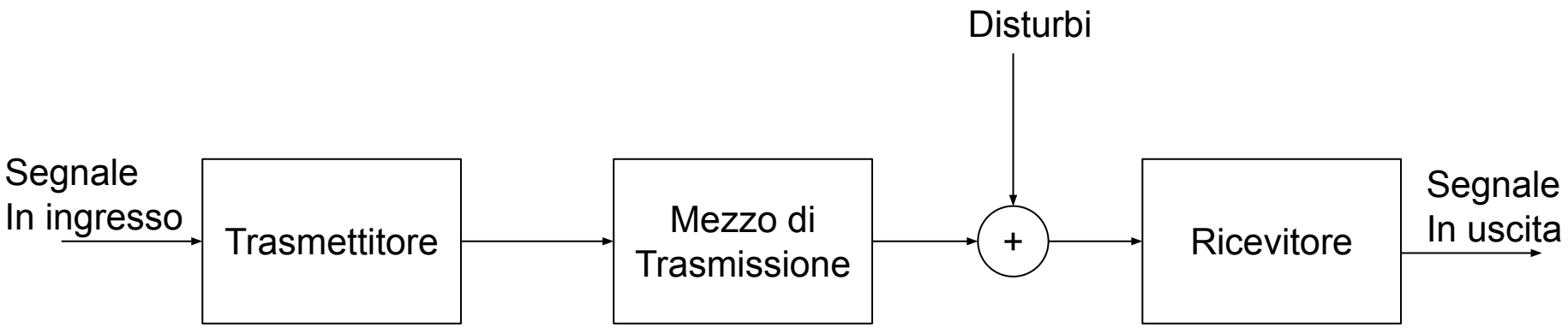
Embedded System

L10

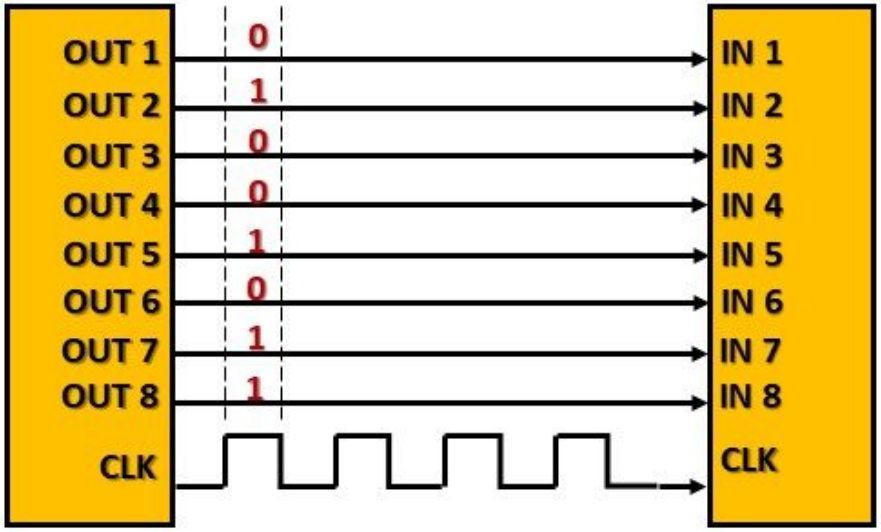
Comunicazioni Seriali

Comunicazioni



Prendiamo in considerazione la trasmissione di un byte: 8 bit.
In una comunicazione di tipo parallelo, per poter inviare un byte abbiamo bisogno di 8 linee dati
(non è sempre così, ma la maggior parte delle volte lo è):

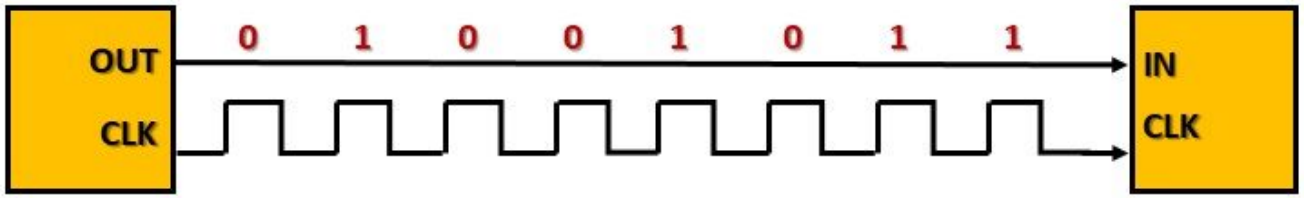
ogni linea invierà un bit, per cui gli 8 bit saranno trasmessi contemporaneamente.



Trasmissione Parallela
Tutti i bit di ogni parola vengono
Trasmessi contemporaneamente
E sincronizzati da un clock

In una comunicazione di tipo seriale si utilizzerà un' unica linea sulla quale saranno inviati in sequenza gli 8 bit.
Le modalità con cui questi bit vengono inviati variano da protocollo a protocollo: ogni protocollo di comunicazione seriale ha le sue regole ed i suoi limiti e quindi le sue applicazioni.

Trasmissione Seriale
Tutti i bit della parola vengono inviati
Uno dopo l'altro e sincronizzati
Da un clock



Come si capisce, a parità di velocità del sistema di trasmissione, una comunicazione di tipo **parallelo** è sicuramente **più veloce**:
gli 8 bit vengono inviati in contemporanea!

Ovviamente questo ha anche i suoi **svantaggi**: c'è bisogno di molte linee dati e le cose si complicano ulteriormente quando su un unico bus (su un'unica linea di trasmissione dati) bisogna collegare più dispositivi: **una marea di cavi**.

Per tale motivo le comunicazioni di tipo parallelo stanno pian piano scomparendo: nei pc, giusto per fare un esempio, si è abbandonato lo standard EIDE a favore del SATA.

I cavi occupano meno spazio e c'è la possibilità di staccare i dispositivi anche mentre sono in funzione senza influenzare il resto del sistema.

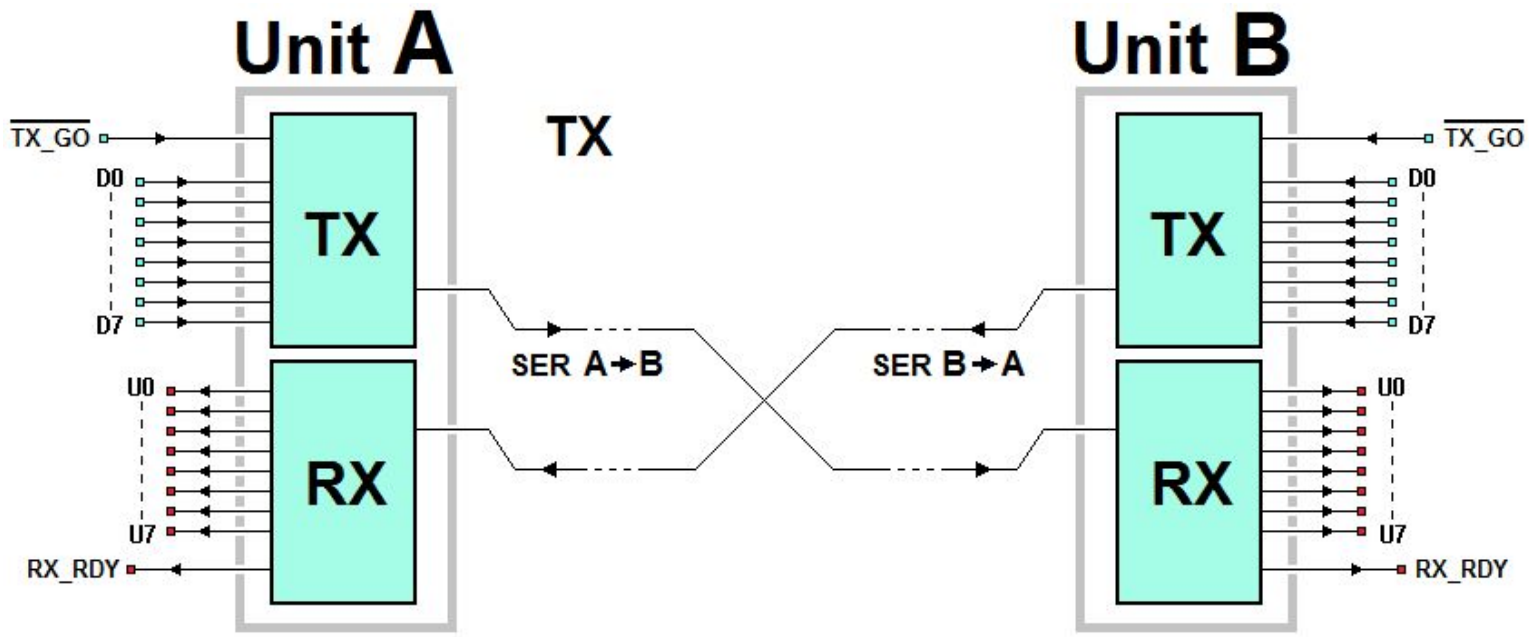
Riguardo alla velocità… ormai le velocità raggiunte hanno sorpassato i vecchi sistemi di comunicazione parallela (questo perché si è deciso di non investire più sulla parallela per i motivi appena detti).

Le modalità di comunicazione (sia di tipo seriale che parallelo) possono essere **Full Duplex** oppure **Half Duplex**:

si parla di comunicazione **full duplex** quando lo scambio di dati avviene in entrambi i sensi e in **contemporanea**.
Nel caso di una trasmissione seriale full duplex possiamo immaginare che vi siano due fili che collegano due dispositivi A e B, il dispositivo A comunicherà verso il B attraverso un filo e il B verso l'A attraverso l'altro filo e questo può avvenire in contemporanea essendo le due linee separate.

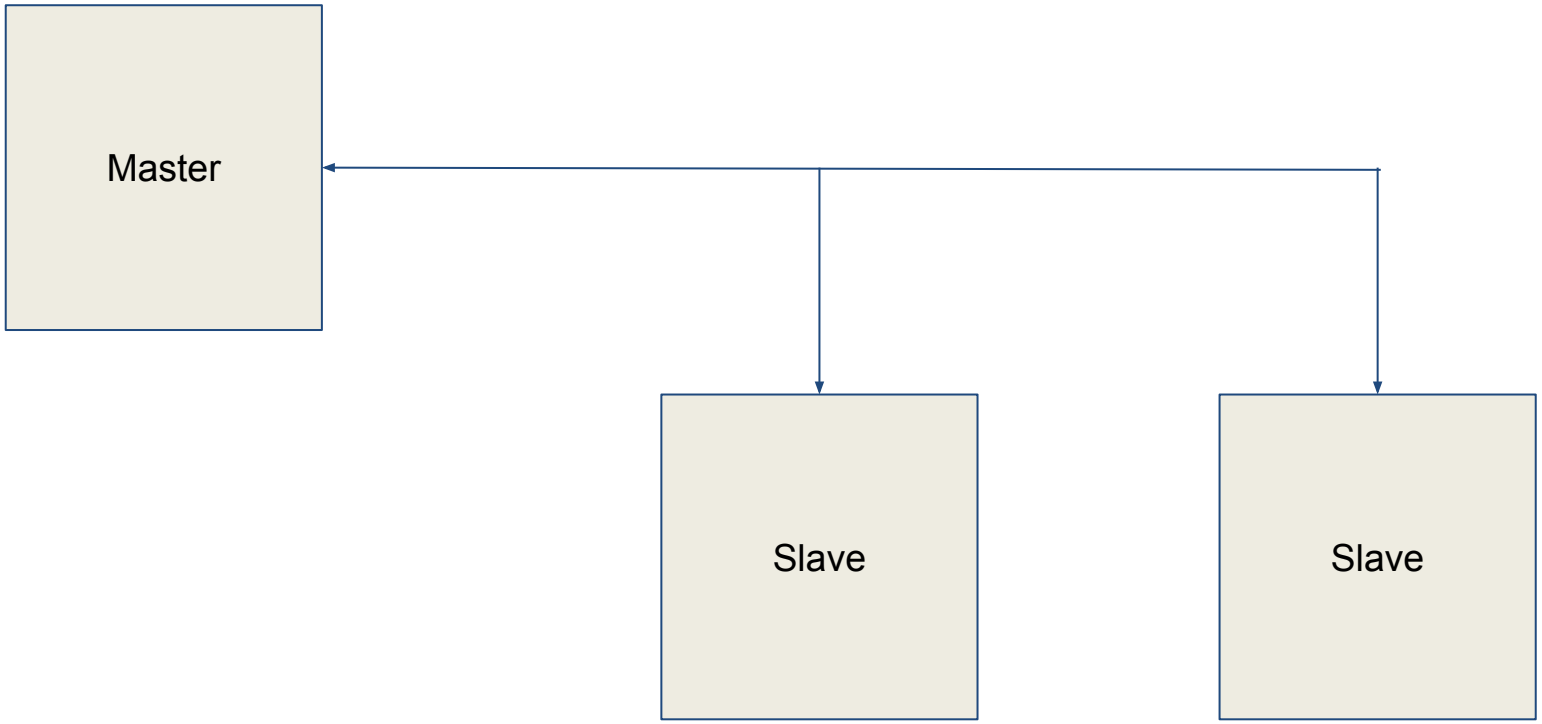
In ogni tipo di comunicazione abbiamo ovviamente bisogno anche della linea di **massa comune**.

Seriale Full Duplex



In una trasmissione **Half Duplex** abbiamo invece a disposizione un'unica linea di comunicazione: potrà trasmettere **un unico dispositivo per volta**, in questo tipo di comunicazione generalmente si individua un dispositivo "**master**" che comanda la comunicazione e uno o più dispositivi "**slave**" che ricevono gli ordini impartiti dal master.

Parlano uno alla volta, e la comunicazione viene diretta dal Master

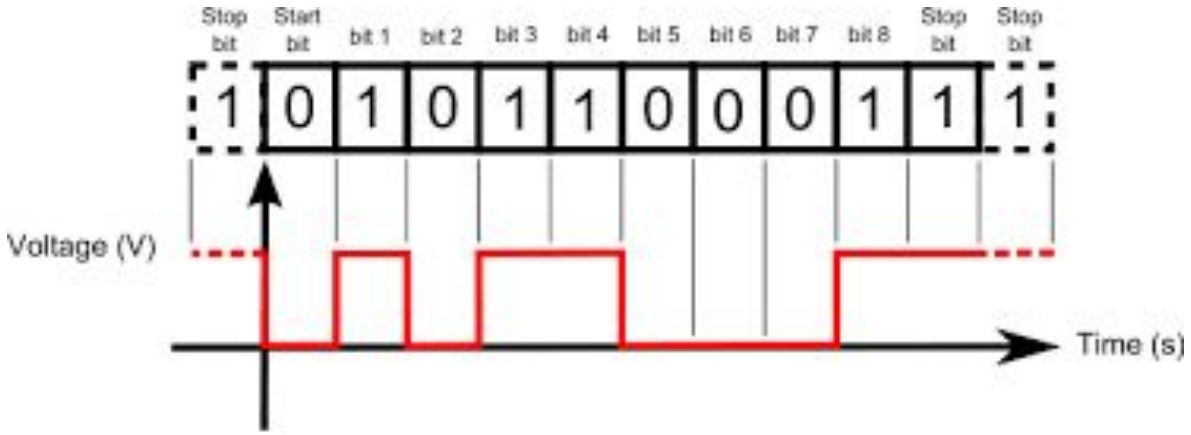


UART

Questo è un esempio di trasmissione seriale “asincrona”. Con questo termine si
Intende una trasmissione nella quale non è presente un segnale di sincronismo,
Un clock, che scandisce quando caricare nel ricevitore i dati.

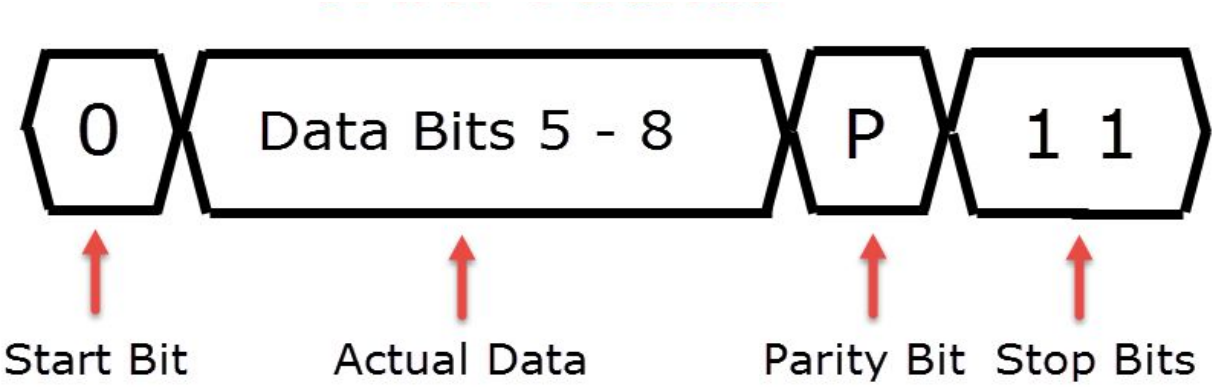
Questa tecnica fa uso di particolari bit di durata predeterminata per avvisare il
Ricevitore su quando inizia e quando finisce una parola.

In queste comunicazioni è fondamentale dire sia al TX che al RX qual’è la velocità
Con cui dovranno essere inviati i dati affinché RX distingua i bit ricevuti



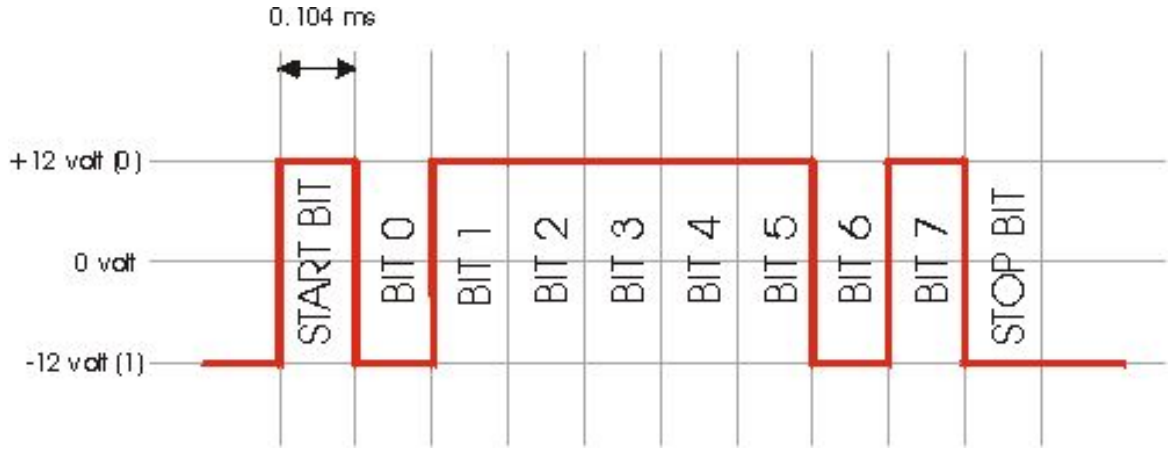
UART

- La linea si trova inizialmente nello stato di riposo, bassa (nessun dato in transito)
- la prima transizione da basso in alto indica l'inizio della trasmissione e dura esattamente 104us (nel caso di una trasmissione a 9600 baud)
- Segue il bit meno significativo (LSB)
- dopo altri 104 us il secondo bit, e così via, per otto volte, fino al bit più significativo (MSB)
- Segue un eventuale bit di parità.
- Segue infine un periodo di riposo della linea di almeno 208 us, cioè due bit di stop
- quindi (eventualmente) inizia un nuovo pacchetto di bit con un nuovo bit di start

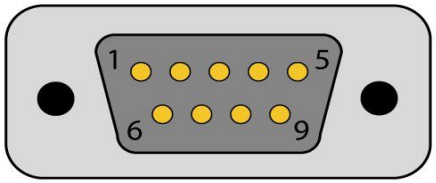


RS-232

- L’ampiezza del segnale è caratterizzata da un valore alto di circa +12V e da un valore basso di circa -12V.
- Il protocollo codifica la tensione positiva come uno ZERO, mentre quella negativa come un UNO.
- Il protocollo è punto-punto, cioè mette in comunicazione due dispositivi.

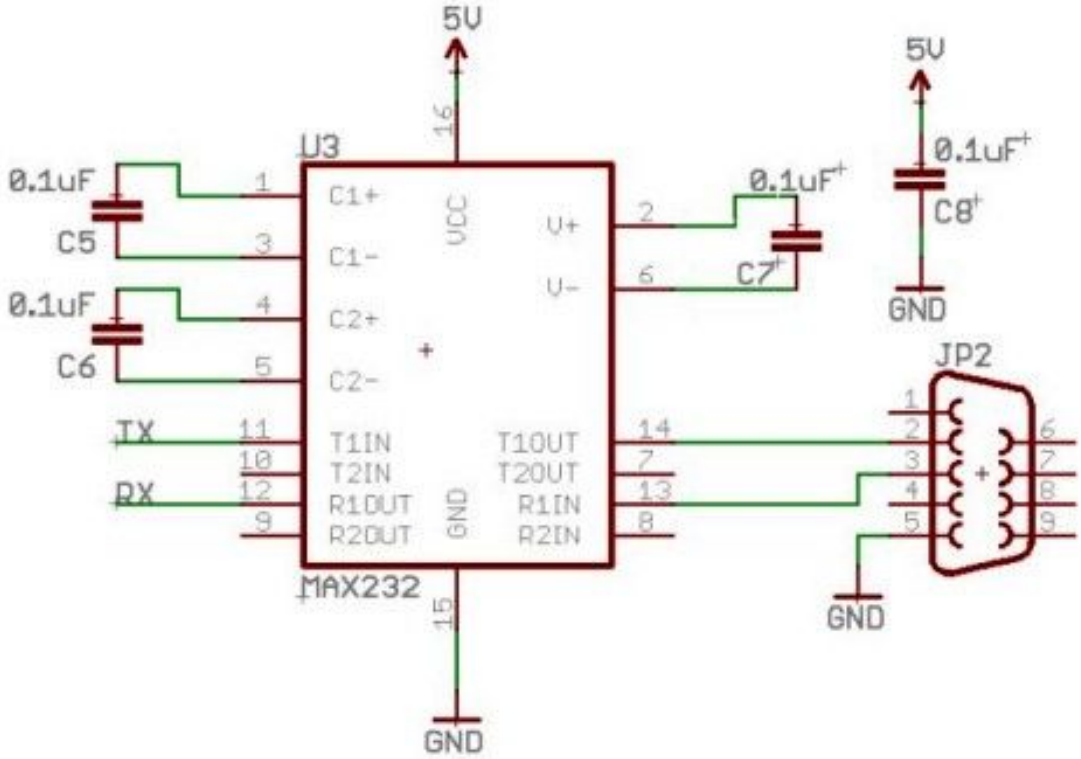


DB9M Connector



RS232 Pin Out

Pin #	Signal
1	DCD
2	RX
3	TX
4	DTR
5	GND
6	DSR
7	RTS
8	CTS
9	RI

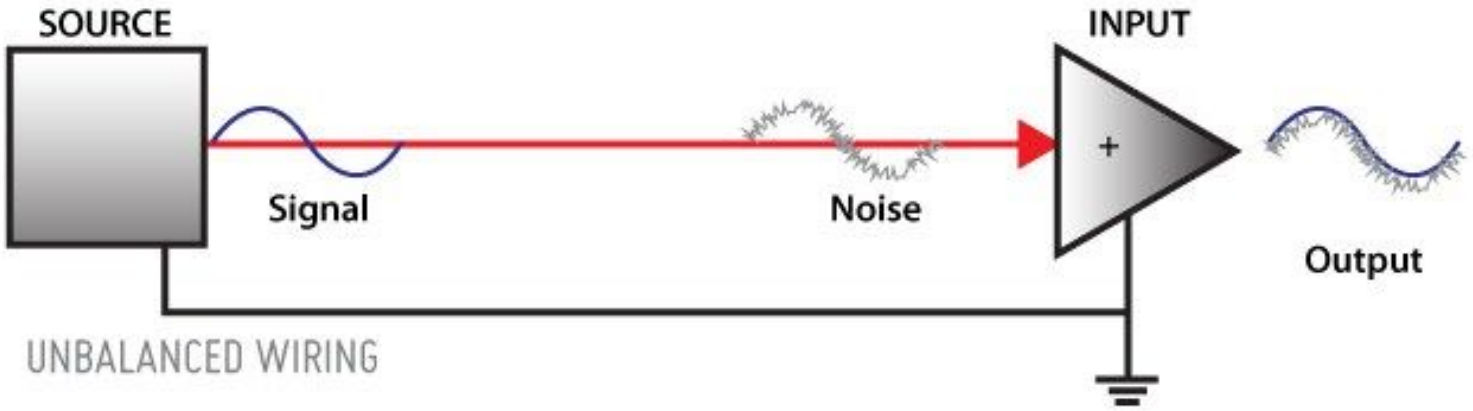



RS232
C6
C5
C8
C7
C11
VCC
GND
RxOUT<--
TxIN -->
C14
CTS
RTS
C13
C9
C12
3V-5.5V Operating
15kV ESD Protect
03271A-03

Caratteristiche RS232

- Massimo un trasmettitore e un ricevitore
- Lunghezza massima del cavo circa 20 mt
- Il protocollo prevede velocità massime di 20kbps, anche se poi troviamo trasmissioni anche fino a 115Kbps
- Tensioni sulla linea: da -12V al +12V
- Half duplex/Full duplex

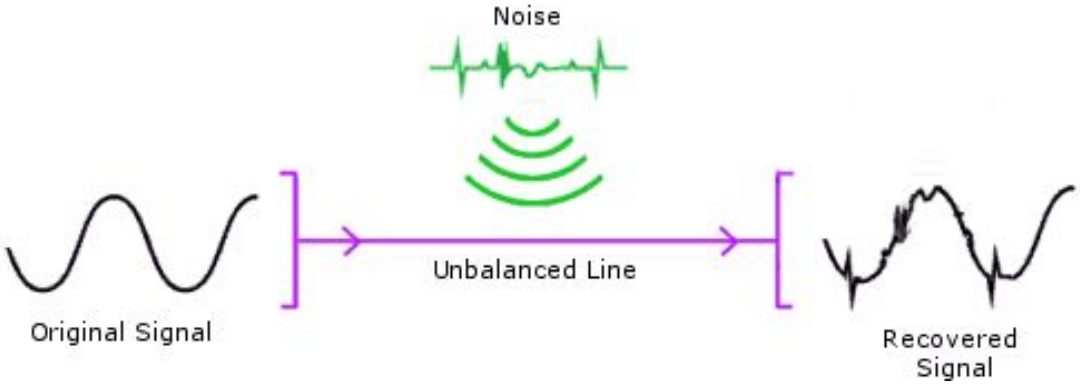
Linea Sbilanciata



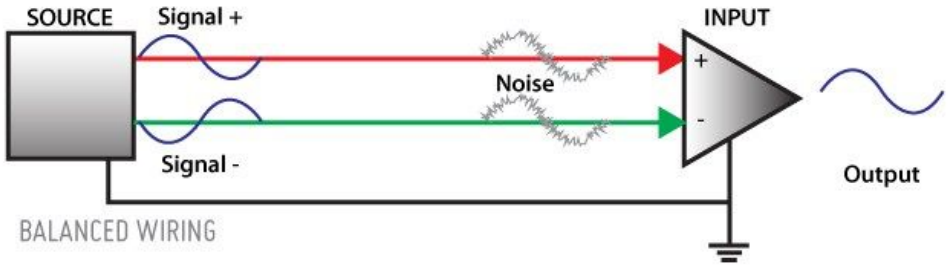
Una linea di trasmissione sbilanciata, trasmette il segnale singolo riferito ad un riferimento comune.

Il rumore (additivo) che si somma lungo il tragitto lo ritroviamo in uscita e rende più sporca l'informazione. A volte tanto da renderla irriconoscibile.

Per questo motivo ci sono dei limiti di lunghezza dei cavi



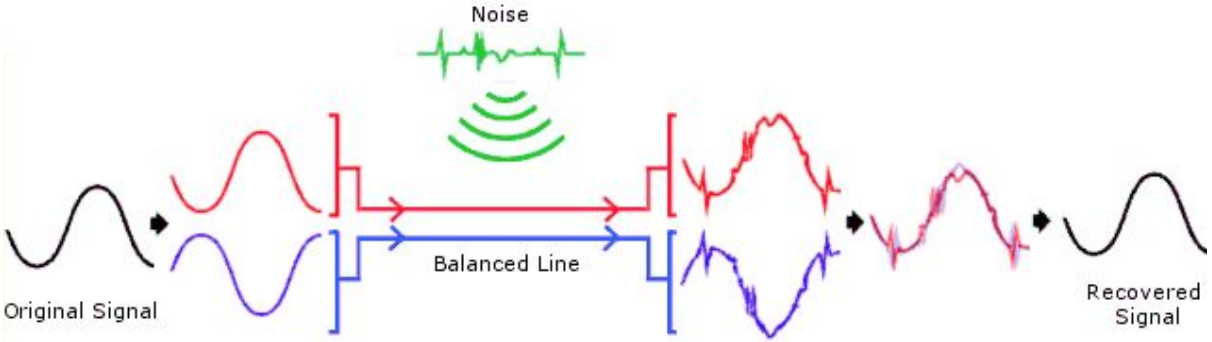
Linea Bilanciata



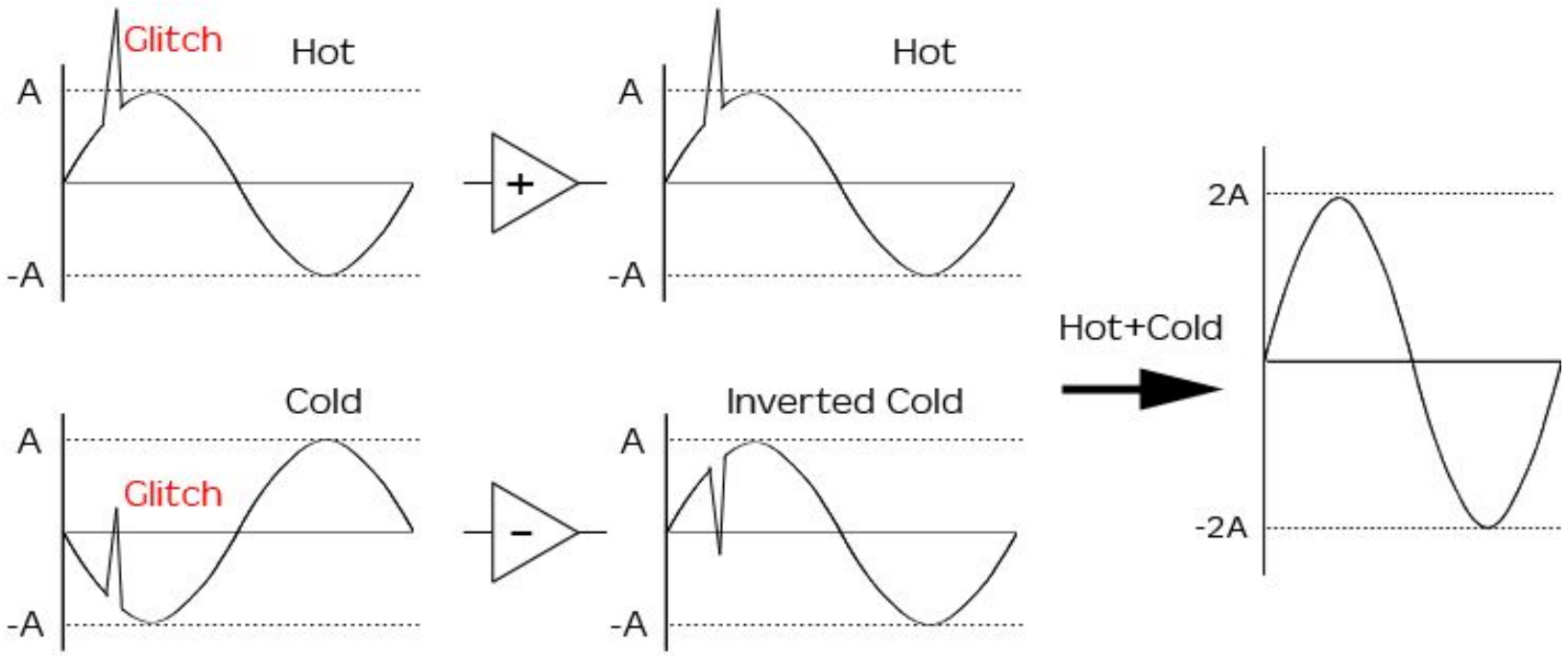
Una linea bilanciata trasmette il segnale in due versioni. Una normale (+) e una esattamente invertita (-)

Il rumore (additivo) si somma ad entrambe le linee

Il ricevitore esegue la differenza dei due segnali. Il rumore viene cancellato, mentre il segnale originale diventa il doppio



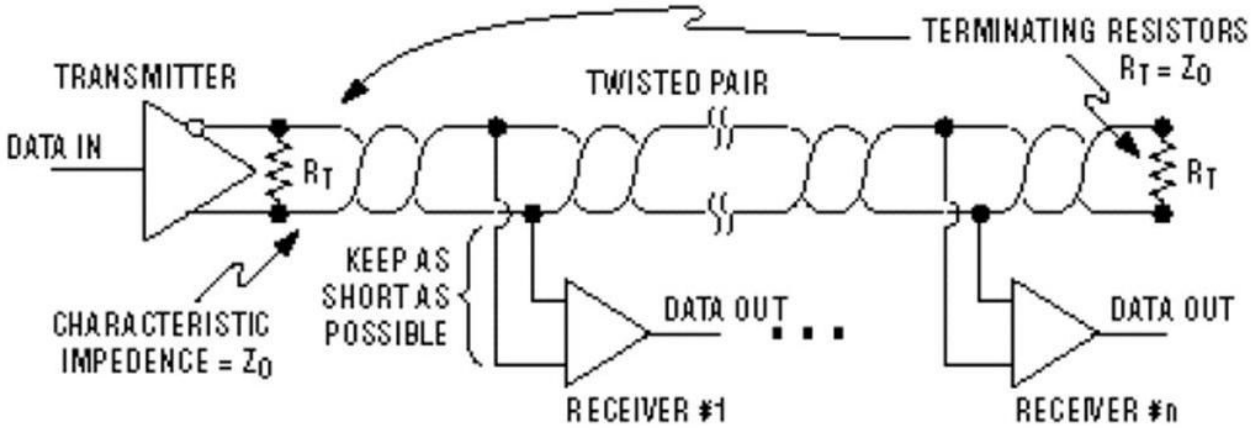
Linea Bilanciata



RS422

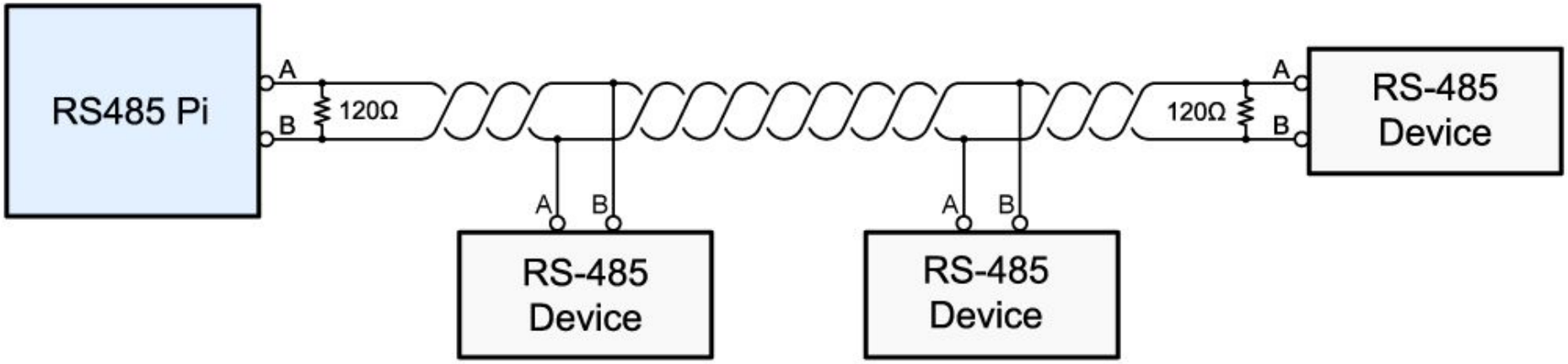
- E’ un protocollo di comunicazione dati seriale su linea differenziale (bilanciata)
- Questa caratteristica gli permette di realizzare connessioni fino ad una distanza massima di circa 1.2Km
- Inoltre prevede un trasmettitore e più ricevitori

RS-422 Connection Diagram



RS485

- E’ un protocollo di comunicazione dati seriale su linea differenziale (bilanciata) half duplex
- Questa caratteristica gli permette di realizzare connessioni fino ad una distanza massima di circa 1.2Km
- Inoltre avendo driver che possono essere impostati “in alta impedenza” il protocollo permette che tutte le periferiche possano lavorare da TX e RX



RS422 - RS485

- Entrambi i protocolli, lavorando su linee bilanciate hanno la necessità che venga rispettata l'impedenza della linea.
- Normalmente la linea richiede un terminatore da 120 ohm, ma non è incredibile trovare dei driver di linea che integrano questa resistenza rendendo superfluo il suo uso.
- Quindi questa necessità va sempre verificata in funzione dei dispositivi che stiamo utilizzando

Configurazione del modulo UART

I registri associati al modulo UART del PIC sono:

TXSTA Transmit Status And Control Register

RCSTA Receive Status And Control Register

SPBRG USART Baud Rate Generator

TXREG USART Transmit Register. Holds the data to to be transmitted on UART

RCREG USART Transmit Register. Holds the data received from UART

Transmit Status And Control Register

TXSTA							
7	6	5	4	3	2	1	0
CSRC	TX9	TXEN	SYNC	-	BRGH	TRMT	TX9D

CSRC: Clock Source Select bit
Asynchronous mode:Don't care.

TX9: 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

TXEN: Transmit Enable bit
1 = Transmit enabled
0 = Transmit disabled

SYNC: USART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

BRGH: High Baud Rate Select bit
1 = High speed
0 = Low speed

TRMT: Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

TX9D: 9th bit of Transmit Data, can be Parity bit

Tale registro si occupa di una parte dei settaggi dell'USART e dello stato della trasmissione. Vediamo che il bit 4 di tale registro, denominato SYNC, se posto a zero ci permette di realizzare la modalità asincrona, realizzando la modalità asincrona avremo che l'I/O RC6 (situato sul pin n°25 del 16F877) sarà il pin utilizzato dal picmicro per trasmettere i dati e RC7 (pin 26) sarà il pin destinato a riceverli (in modalità sincrona, invece, il primo avrebbe avuto la funzione di clock e il secondo di linea dati).

Il bit TX9 ci permette di selezionare la modalità di trasmissione ad 8 o a 9 bit. La modalità ad 8 bit è generalmente quella più diffusa.

Il bit TXEN abilita la trasmissione.

BRGH abilita (1) l'alta velocità, questo bit deve essere tenuto in considerazione quando successivamente andiamo a scegliere la velocità di trasmissione.

Il bit TRMT serve ad indicare lo stato dello shift register di trasmissione: quando vogliamo trasmettere un byte, il byte deve essere caricato nel registro chiamato TXREG, il dato quindi passa in automatico da TXREG allo shift register (TSR) e quindi da qui viene trasmesso, quando il dato viene trasmesso, lo shift register si svuota e viene quindi settato ad 1 il bit TRMT. Tale bit comunque non viene quasi mai utilizzato nei programmi.

Abbiamo infine il bit TX9D nel quale dovremo impostare il nono bit di dati qualora avessimo selezionato la trasmissione a 9 bit. Nel caso stiamo realizzando una trasmissione ad 8 bit, tale bit può essere utilizzato per inviare il bit di parità, ma non ce ne occuperemo.

Registro RCSTA

Tale registro si occupa di un'altra parte del settaggio dell' USART e dello stato relativo alla ricezione. Per non dilungarmi troppo, dirò che di tale registro ci interessano principalmente i bit:

SPEN, che abilita (1) appunto l'utilizzo del modulo USART, configurando i pin RC6 e RC7 come destinati all'utilizzo dell'USART (dovremo comunque impostare queste due porte anche come ingressi agendo sul registro tristato).

RX9, che abilita la ricezione a 9 bit.

CREN, che abilita la ricezione continua dei dati.

RX9D, nono bit di dati ricevuto eventualmente lo volessimo utilizzare.

RCSTA							
7	6	5	4	3	2	1	0
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D

Selezionare il Baud Rate

Il baud rate, la velocità di trasmissione, è in funzione del quarzo utilizzato per il clock di sistema, del bit BRGH e del valore caricato nel registro SPBRG. Il suo valore può essere calcolato con le formule fornite sul datasheet:

TABLE 10-1: BAUD RATE FORMULA

SYNC	BRGH = 0 (Low Speed)	BRGH = 1 (High Speed)
0	(Asynchronous) Baud Rate = FOSC/(64 (X + 1))	Baud Rate = FOSC/(16 (X + 1))
1	(Synchronous) Baud Rate = FOSC/(4 (X + 1))	N/A

Legend: X = value in SPBRG (0 to 255)

```
SPBRG = (_XTAL_FREQ/(long)(64UL*baudRate))-1;
```

Ovviamente sul picmicro non otterremo mai dei valori di baud rate precisi come quelli impostabili sul computer, ci sarà sempre una certa percentuale di errore, sul datasheet si consiglia comunque di selezionare sempre BRGH=1 anche con baudrate bassi in maniera da ottenere un errore inferiore.

Alla pagina successiva del datasheet è riportata un'utile tabella con i valori da impostare nel registro SPBRG (valore x della formula vista poco fa) in funzione del bit BRGH e del quarzo utilizzato. Diamo un occhio alla tabella relativa a BRGH=1 :

TABLE 10-4: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 1)

BAUD RATE (K)	Fosc = 20 MHz			Fosc = 16 MHz			Fosc = 10 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-	-	-	-
1.2	-	-	-	-	-	-	-	-	-
2.4	-	-	-	-	-	-	2.441	1.71	255
9.6	9.615	0.16	129	9.615	0.16	103	9.615	0.16	64
19.2	19.231	0.16	64	19.231	0.16	51	19.531	1.72	31
28.8	29.070	0.94	42	29.412	2.13	33	28.409	1.36	21
33.6	33.784	0.55	36	33.333	0.79	29	32.895	2.10	18
57.6	59.524	3.34	20	58.824	2.13	16	56.818	1.36	10
HIGH	4.883	-	255	3.906	-	255	2.441	-	255
LOW	1250.000	-	0	1000.000		0	625.000	-	0

BAUD RATE (K)	Fosc = 4 MHz			Fosc = 3.6864 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-
1.2	1.202	0.17	207	1.2	0	191
2.4	2.404	0.17	103	2.4	0	95
9.6	9.615	0.16	25	9.6	0	23
19.2	19.231	0.16	12	19.2	0	11
28.8	27.798	3.55	8	28.8	0	7
33.6	35.714	6.29	6	32.9	2.04	6
57.6	62.500	8.51	3	57.6	0	3
HIGH	0.977	-	255	0.9	-	255
LOW	250.000	-	0	230.4	-	0

Una volta settati propriamente i registri per lavorare con la seriale, dobbiamo soltanto sapere che per trasmettere un byte basta semplicemente caricarlo nel registro TXREG (basta fare TXREG=valore del byte).

Finito di trasmettere il byte viene settato il flag di interrupt TXIF.
Volendo si può intercettare l'interrupt di trasmissione, ma normalmente basta controllare TXIF per verificare che la trasmissione precedente sia terminata.

Il bit TXIF viene azzerato in automatico appena si caricano nuovi dati nel registro di trasmissione.

Per ricevere un byte, la cosa migliore da fare è sicuramente intercettare l'interrupt di ricezione seriale, che si abilita portando ad 1 il bit RCIE: nell'ISR intercetteremo quindi il flag di avvenuta ricezione seriale, ovvero il bit RCIF.

Il dato ricevuto da seriale lo avremo disponibile nel registro RCREG.
Il flag di ricezione su seriale, RCIF, viene resettato in automatico appena si effettua la lettura del registro RCREG, per cui non dovremo preoccuparci di resettarlo (anche perchè non si può dal momento che è a sola lettura).

I flag di abilitazione di ricezione e di trasmissione su seriale, quindi, si trovano nel registro PIE1.

PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
EEIE	ADIE	RCIE	C2IE	C1IE	OSFIE	TXIE	TMR1IE
bit 7							bit 0

Considerazioni pratiche

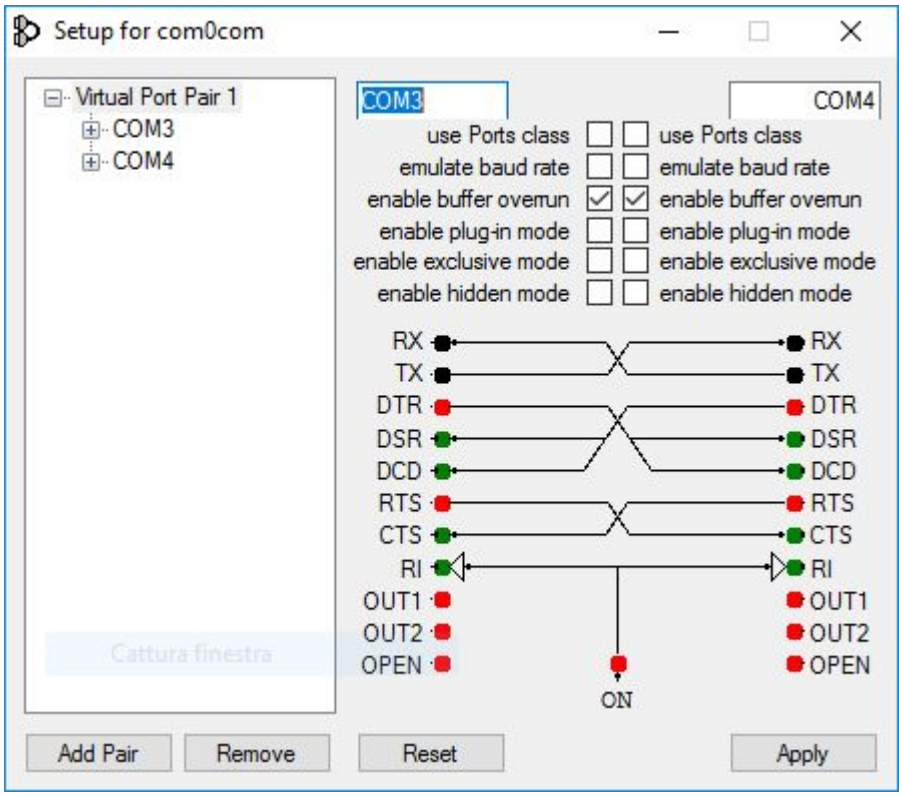
Stiamo lavorando con un simulatore, e per certi versi la situazione si complica.
La comunicazione Seriale prevede l'utilizzo delle porte "COM" che purtroppo non godono della facoltà di essere "condivise" tra i vari applicativi.

Bisogna quindi "virtualizzare" le porte al fine di simulare una connessione fisica tra due porte collegate allo stesso PC.

Considerazioni pratiche

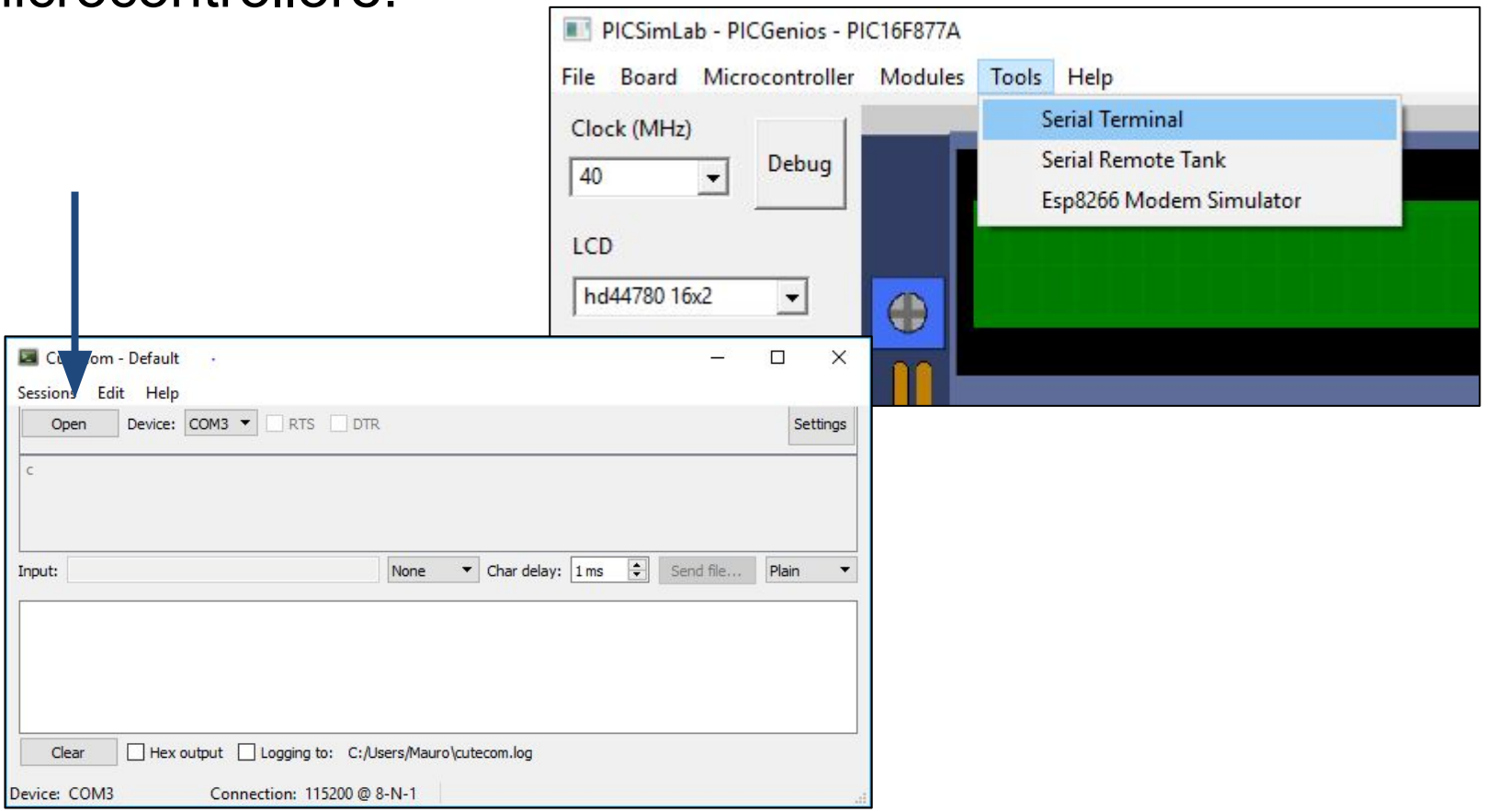
https://sourceforge.net/projects/com0com/

questo software permette di emulare e connettere tra loro porte seriali.



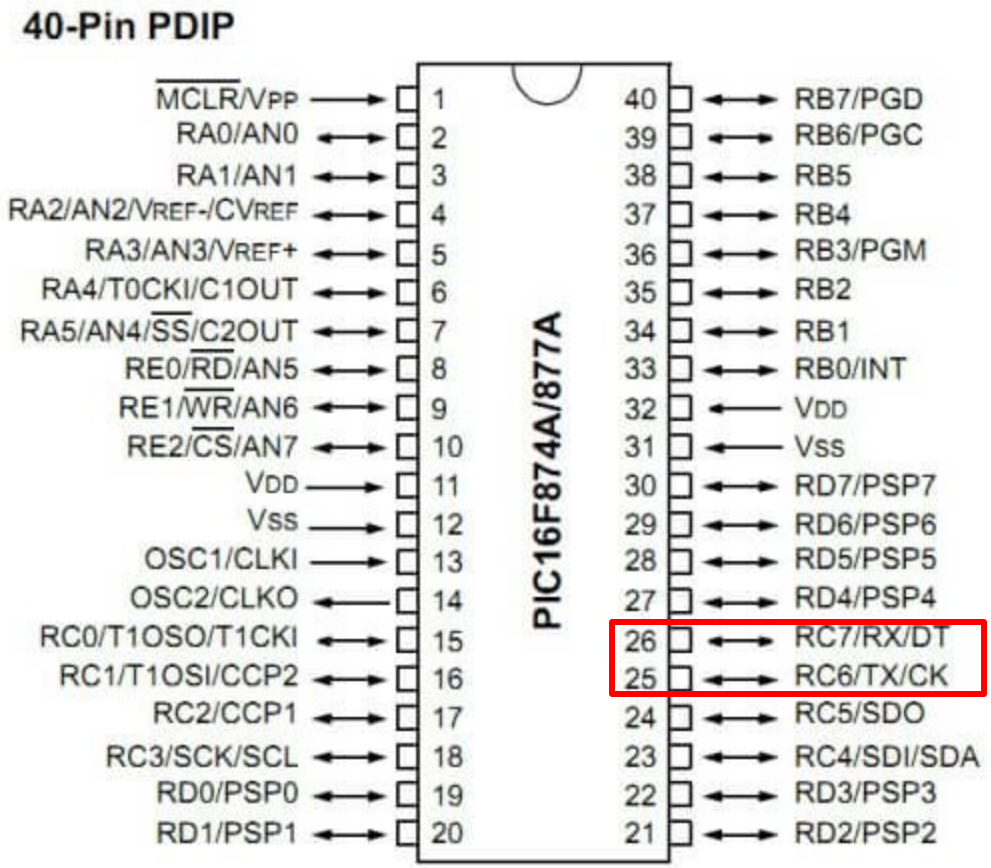
Considerazioni pratiche

Il terminale che ci permette di comunicare con il microcontrollore:



Considerazioni Pratiche

I due terminati che si occupano della trasmissione e ricezione dei dati seriali sono rispettivamente RC6 e RC7



Inizializzazione del modulo UART

Prevediamo la creazione di una funzione dedicata alla configurazione del modulo.

1. Conviene configurare tale funzione affinchè possa accettare un parametro in ingresso che rappresenti il BAUD RATE che vogliamo impostare (int).

2. A questo punto impostiamo il registro di configurazione della porta C al fine di avere RC6 in output e RC7 in input.

3. Impostiamo a 1 nel registro TXSTA il bit TXEN

4. Impostiamo a 1 nel registro RCSTA i bit SPEN e CREN (SPEN abilita la porta seriale, CREN abilita il ricevimento continuo dei dati)

5. Settiamo SPBRG tramite la formula della slide precedente

```
SPBRG = (XTAL_FREQ/(long)(64UL*baudRate))-1;
```

Creiamo una funzione che si occupa della trasmissione di un dato

Anche per questa funzione dobbiamo prevedere il passaggio di un valore, in questo caso di tipo char.

1. Attendiamo che TXIF torni a zero per assicurarci che eventuali precedenti trasmissioni non siano ancora in corso.

2. Copiamo il dato da trasmettere nel registro TXREG

Creiamo l'interrupt che si occupa della ricezione di un dato

Ricordiamoci di impostare a 1 il bit che abilita gli interrupt GIE, quello che abilita gli interrupt di periferica PEIE, e il flag che abilita l'interrupt in ricezione RCIE (nel registro PIE1)

Se rileviamo l'interrupt per la ricezione della seriale:

1. Attendiamo che il flag RCIF vada a 1
2. Resettiamo RCIF
3. Salviamo il registro RCREG nella variabile di ricezione

Esercizio

Si richiede di realizzare un codice che permetta la trasmissione e la ricezione di un singolo carattere.

- Alla pressione di un pulsante della scheda (RB0) il carattere ‘A’ viene inviato tramite Seriale al terminale.

- Contemporaneamente il carattere verrà scritto sul display LCD

- Un carattere qualunque inviato dal terminale viene visualizzato sul display LCD

Esercizio

Si richiede di realizzare un codice che permetta la trasmissione e la ricezione di più comandi.

- Alla pressione di un pulsante del tastierino numerico, il numero o simbolo verrà inviato al terminale
- Contemporaneamente il carattere verrà scritto sul display LCD
- Un carattere qualunque inviato dal terminale viene visualizzato sul display LCD

Esercizio

Si richiede di realizzare un codice che:

- Invii un numero crescente tramite seriale al terminale.
- Tale numero verrà incrementato ogni 1 secondo.
- Il numero verrà visualizzato anche sul display, nella prima riga

- Un carattere qualunque inviato dal terminale viene visualizzato sul display LCD, sulla seconda riga.
- Arrivati alla fine della riga il display si resetta.