APPUNTI SULLA PORTA SERIALE
“RS232C”

Ripropongo, aggiornato e notevolmente ampliato, un piccolo lavoro scolastico sulla porta seriale sviluppato teoricamente in classe e praticamente in laboratorio nel lontano 1996.

GENERALITÀ SULLA COMUNICAZIONE

La trasmissione dell’informazione tra utenze diverse può essere di tipo Parallelo o di tipo Seriale. Fondamentalmente l'elaboratore dialoga con le periferiche in modo parallelo a causa del suo modo particolare di muovere i dati. Infatti al suo interno la C.P.U. dialoga con le memorie e con le periferiche tramite i BUS, siano essi adatti per trasmettere gli indirizzi o per trasmettere i dati o per trasmettere i controlli. Un *"bus"* è praticamente un fascio di linee dove i segnali transitano in parallelo. Questo tipo di comunicazione è molto veloce ed è sempre accettabile se le distanze tra i componenti sono relativamente corte, in modo tale da non risentire sensibilmente dei fenomeni di accoppiamento induttivo e capacitivo tra traccia e traccia del bus. Fenomeno avvertito specialmente quando le velocita delle C.P.U. sono molto elevate. Se è necessario un collegamento con l'esterno, ad esempio con una stampante, la lunghezza del cavo difficilmente può raggiungere i due metri senza il pericolo di modifica o perdita dei dati. Quando vi è la necessità di collegare periferiche (chiamate di solito **D.C.E.** = Data Circuit Equipment) molto distanti dall'elaboratore (chiamato anche **D.T.E.** = Data Terminal Equipment) l'unica soluzione è la **trasmissione seriale**.

LA TRASMISSIONE SERIALE

La trasmissione seriale avviene su un singolo filo e, quindi, non può essere influenzata da segnali adiacenti. Su un singolo filo però, un dato non può essere trasmesso in blocco, ad es. ad otto bit alla volta contemporaneamente, caratteristica peculiare della trasmissione parallela, ma un bit alla volta, riducendo così notevolmente la velocità di scambio delle informazioni.

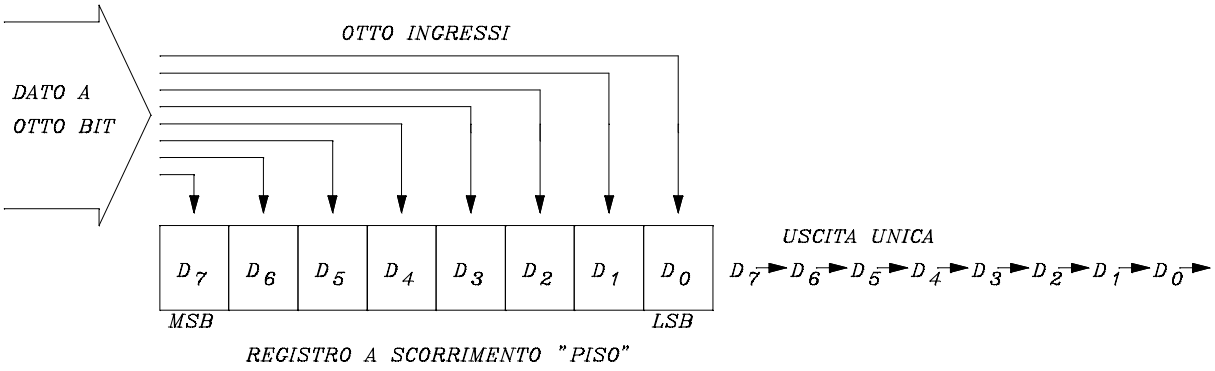

Fig.01

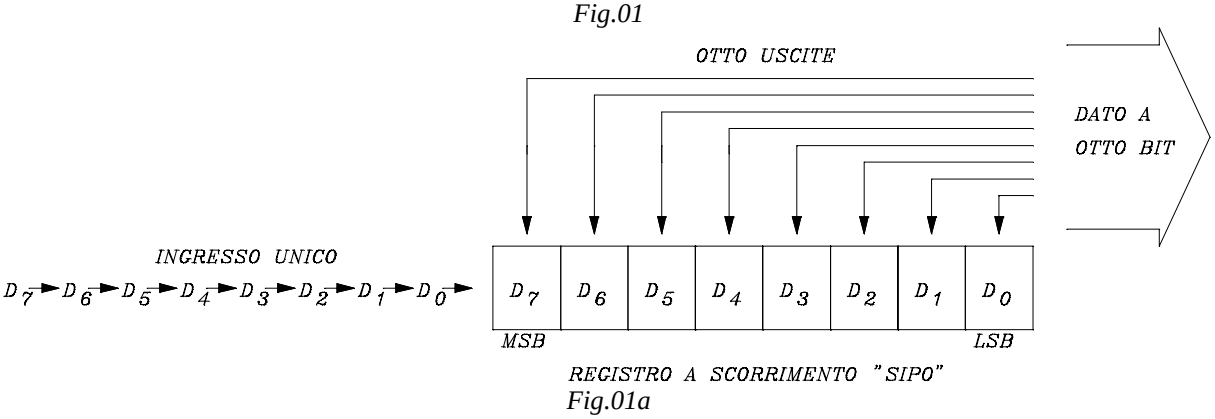

Fig.01a

Poiché il dato nell'elaboratore nasce "parallelo" vi è la necessità di trasformarlo in seriale mediante un'adatta conversione, decidendo anche, quale degli otto bit entrerà prima nella trasmissione, il primo D_0 o l'ultimo D_7. Su questa decisione, comunque, è unanime l'accordo di considerare il primo bit trasmesso quello D_0 a peso minore (Fig.01).
La conversione di un dato espresso in parallelo in una serie di bit in successione è di solito realizzata mediante un particolare registro chiamato “Registro a Scorrimento P.I.S.O.” (P.I.S.O. = Parallel In/Serial Out), dove gli otto bit entrano contemporaneamente, ed escono poi in successione, con un particolare software di supporto. Essi si ricomporranno attraverso un altro tipo di registro chiamato S.I.P.O. (Serial In/Parallel Out ossia Registro ad ingresso seriale ed uscita parallela) come in Fig.01a.

La trasmissione di questa serie di bit può essere trasmessa in modo **Seriale Sincrono** o in modo **Seriale Asincrono.** La trasmissione Sincrona è più veloce di quella Asincrona però ha bisogno di adatti segnali di sincronismo per far corrispondere istante per istante il trasmettitore con il ricevitore. Nel caso di perdita accidentale di sincronismo tutta la comunicazione viene persa.

LA TRASMISSIONE SERIALE ASINCRONA

La parola stessa ci fa capire che tra ricevitore e trasmettitore **non è necessario uno stretto sincronismo** per tutta la durata della comunicazione: l'essenziale è che esso si mantenga solo per il tempo della trasmissione del Byte di dato. Per questo motivo si lancia prima un bit di partenza (Start) poi vengono trasmessi i bit di dato seguiti da un eventuale bit di rilevazione di errore, infine si lancia un bit di fine dato (Stop). Dopo aver ricevuto questa serie di bit il ricevitore rimarrà in attesa di un altro dato finché non rileva la presenza di un nuovo bit di Start. Si vede perciò che nella Trasmissione Asincrona è sufficiente che il sincronismo si mantenga per i soli bit di simbolo. La Fig.02 mostra, tra le tante, una classica formattazione del segnale trasmesso, con le specifiche appena accennate. Si vede come esso sia composto da un bit di START, da otto bit di DATO, da un bit di PARITA' (rilevamento d’errore) e da un bit di STOP.

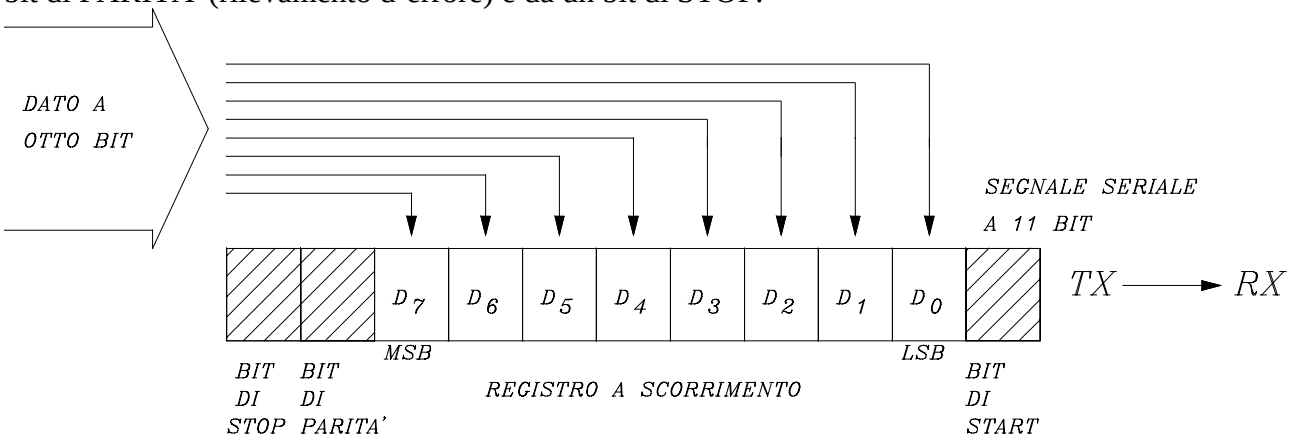

Fig.02

Tenendo conto del verso di trasmissione si nota come il primo bit di dato trasmesso è il bit a minor peso (L.S.B.=Last Significant Bit) mentre l'ultimo è quello a maggior peso (M.S.B.=Most Significant Bit).

La trasmissione seriale asincrona si è diffusa rapidamente perché è semplice da gestire. I circuiti integrati progettati e costruiti apposta per implementarla sono chiamati **U.A.R.T.** (Universal Asyncronus Receiver Trasmitter) e forniscono facilmente la soluzione dei vari problemi di protocollo di comunicazione mediante una serie di registri interni di cui sono provvisti. Ovviamente è necessario un ***programma di inizializzazione e gestione*** di questi registri che può essere scritto con un **linguaggio di programmazione ad alto o a basso livello**, ad esempio con il Pascal o con l’Assembler. Poiché la comunicazione avviene mediante una successione di bit è importante conoscere la velocità di trasmissione degli stessi perché la cadenza dei bit deve essere ovviamente la stessa sia per il trasmettitore che per il ricevitore. Questa velocità è chiamata **BAUD RATE** ed in pratica esprime il numero di bit trasmessi al secondo.

EIA RS232C

L'**EIA RS232** è stato il primo sistema di comunicazione seriale veramente valido e sicuro ed ha gestito per molti anni tutte le connessioni tra periferiche ed elaboratori, fin quando non è diventato teoricamente obsoleto con la realizzazione del nuovo protocollo seriale chiamato U.S.B. Ma ancora è presente su elaboratori mediamente datati e molti collegamenti sono ancora in uso con questo tipo di porta, specialmente nel campo industriale tra le periferiche e i PLC o i microcontroller. La necessità di gestire con sicurezza una comunicazione seriale ha portato alla creazione di uno standard applicativo creato inizialmente dalle industrie americane (E.I.A. R.S.= Electronic Industry Alliance - Recommended Standard) denominato EIA-RS232C che in seguito è diventato lo standard accettato a livello internazionale. L'Interfaccia RS232C è basata su questi quattro parametri generali:

1) Piedinatura dei connettori.
2) Significato dei vari segnali
3) Livelli logici dei segnali
4) Velocità di trasmissione

1) Piedinatura dei connettori

In Fig.03 sono riportati i due connettori standard che vengono usati per le porte seriali. Essi possono essere indifferentemente a 25 o a 9 piedini. Sul connettore a 25 piedini (derivato dal connettore Cannon della porta parallela), molti di essi sono inutilizzati. Sul connettore a 9 piedini, essi sono invece tutti utilizzati. Nella Fig.03 sono indicati in nero i piedini a cui fanno capo i segnali più importanti.

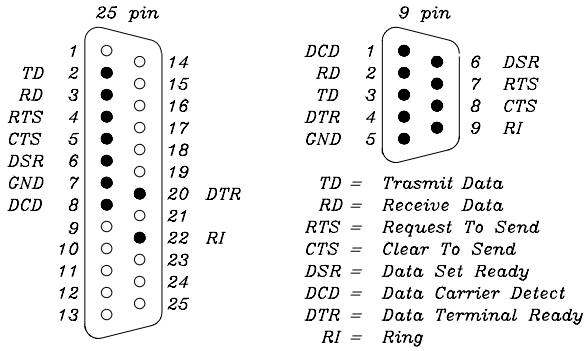

Fig.03

I connettori (a 9 o a 25 piedini) posti sul DTE (Data Terminal Equipment, praticamente l'elaboratore) sono prese di tipo maschio. Quindi il cavo di connessione al DTE dovrà terminare con un connettore a spina femmina. ***I connettori (a 9 o a 25 piedini) posti sul DCE (Data Communication Equipment ossia la periferica) sono prese di tipo femmina.*** Quindi il cavo di connessione al DCE dovrà terminare con un connettore a spina maschio.

2) Significato dei vari segnali

In Fig.03 sono anche riportate le sigle dei vari segnali associati ai piedini, con il significato mnemonico per ciascuno di essi:

TD	**(Trasmit Data)**	**Dato trasmesso**	**DTE→DCE**
RD	**(Receive Data)**	**Dato ricevuto**	**DCE→DTE**
RTS	**(Request To Send)**	**Richiesta di trasmissione**	**DTE→DCE**
CTS	**(Clear To Send)**	**Pronto a trasmettere**	**DCE→DTE**
DSR	**(Data Set Ready)**	**Modem pronto**	**DCE→DTE**
DTR	**(Data Terminal Ready)**	**Terminale dati pronto**	**DTE→DCE**
RI	**(Ring)**	**Indicatore di chiamata**	**DCE→DTE**
GND	**(Ground)**	**Massa comune**	
DCD	**(Data Carrier Detect)**	**Rivelatore di portante dati**	

Spieghiamo ora brevemente il significato dei vari segnali:

TD: Questa linea trasmette dati in formato adeguato (ASCII per esempio). Se non ci sono dati da trasmettere essa è nello stato logico "1" (MARK).

RD: Questa linea riceve i dati nello stesso formato. Vale quanto detto per la linea TD.

DTR: Il terminale è pronto a comunicare.

DSR: Il modem segnala che ha stabilito il collegamento. Se questo segnale è disattivato il DTE (elaboratore) ignora tutti i segnali tranne RI.

RTS: Richiesta invio dati. E' attiva quando il terminale DTE segnala al DCE che deve o può inviare dati.

CTS: E' attivo se il modem è pronto a ricevere i dati.

GND: Massa comune.

DCD: Rivelatore di portante. Il modem segnala che ha ricevuto il segnale di richiesta ed è pronto a ricevere dati.

RI: Indicatore di chiamata.

Non tutti questi segnali sono necessari contemporaneamente, per una comune trasmissione.
Addirittura la trasmissione potrebbe avvenire con i soli segnali TD e RD, comunque, nei casi più semplici, vengono richiesti anche i segnali DTR e DSR, che avvertono che i terminali sono pronti. Ma, per rendere più sicuro il collegamento è bene che vi siano anche RTS e CTS per fermare la comunicazione in caso di necessità**.**

In conclusione, una linea seriale ben gestita dovrà avere almeno sette fili di collegamento, compresa la massa.

3) Livelli logici dei segnali

I valori di tensione che vengono applicati ai dati sono in ***logica negativa*** e sono nominalmente:

- 12V per il livello logico "1" chiamato **MARK**
+12V per il livello logico "0" chiamato **SPACE**

Il bit di START è a livello logico "0" quindi è uno SPACE (+12V), mentre il bit di STOP è a livello logico "1" quindi è un MARK (-12V).
Il valore minimo accettabile delle tensioni sulla linea non deve essere inferiore a $\pm 3V$.

Nel caso di attesa di comunicazione la linea è sempre in stato di MARK, perciò è facile notare se è in atto un collegamento pronto a trasmettere o a ricevere dati: basta controllare se vi è potenziale negativo sul piedino n°3.

4) Velocità di trasmissione

Si definisce ***"Velocità di trasmissione"*** il numero di bit trasmessi al secondo e si misura in BAUD. Vi è una normalizzazione internazionale sulle velocità adoperate.
Tipiche sono le velocità: 110, 150, 600, 1200, 2400, 4800, 9600 b/s. Comunque sono spesso utilizzate velocità più basse (*45, 50, 57, 75, 100)* e più alte di queste (*19200, 38400, 57600, 115200*). La scelta della velocità è in funzione delle caratteristiche del mezzo trasmissivo e dalla più o meno lentezza di risposta delle periferiche.

Facciamo un esempio pratico di collegamento seriale: vogliamo collegare a colloquio **(handshaking= stretta di mano)** un elaboratore **(DTE)** e un plotter **(DCE)** e vediamo quali segnali sono necessari per attuarlo.
Per semplificare al massimo il dialogo (handshake) ammettiamo che l'elaboratore sia sempre pronto ad inviare dati al plotter e che il plotter risponda solo con un *"terminale non pronto"* (DTR) se è pieno di dati.
Bastano quindi solo tre fili poiché il plotter non trasmette dati per sua costituzione:

1) Il filo di trasmissione dati tra il piedino 2 (TD) del DTE al piedino 3 (RD) del DCE
2) Il filo di controllo tra il piedino 20 (DTR) del DCE e il piedino 6 (DSR) del DTE.
3) Il filo di massa tra i piedini 7 del DTE (elaboratore) e 7 del DCE (plotter).

Il piedino 20 (DTR) sul plotter (DCE), se attivo, informa che esso è pronto a ricevere i dati e il collegamento col piedino 6 (DSR) fornisce all'elaboratore l'informazione che la linea è attiva e si possono trasmettere i dati.

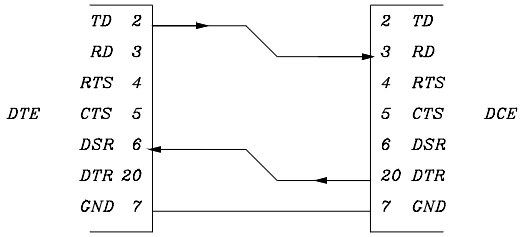

Fig.04

In Fig.04 è disegnato il semplice cavo di collegamento tra elaboratore e plotter.

----*----

UART INS8250A

Vi è un'interrupt del BIOS, l'**INT14h,** che dà la possibilità di realizzare in modo molto semplice una comunicazione seriale sulle due porte, COM1 e COM2, di cui sono dotati quasi tutti gli elaboratori mediamente datati. Però, essendo questa interrupt dipendente dal particolare BIOS impostato sulla macchina, spesso i programmi non possono essere esportati se non tra macchine perfettamente compatibili. Inoltre, la INT14h non supporta velocità al di sotto di 110 baud perciò non può essere utilizzata per la RTTY. Se ne deduce che un perfetto controllo sulla comunicazione si può ottenere solo programmando direttamente un particolare integrato (U.A.R.T.), nato apposta per gestire la comunicazione seriale. L'**INS8250A,** inserito nella piastra dell'elaboratore, è l'integrato più diffuso per la realizzazione di una comunicazione seriale.
Gli indirizzi prestabiliti per le due porte COM1 e COM2 sono uguali su tutte le macchine elaboratici e sono **03F8h** per la COM1 e **02F8h** per la COM2, quindi su questi due indirizzi noi dobbiamo lavorare (sono comunque disponibili a volte altri due indirizzi: 3E8h per la COM3 e 2E8h per la COM4).

A tale scopo l'INS8250A ha a disposizione ***dieci registri*** raggiungibili tramite particolari offset rispetto ai valori di indirizzo riportati sopra. Prendendo in considerazione la porta COM1 il cui indirizzo I/O (Input/Output) è **03F8h** scriviamo gli offset che portano ai vari registri:

3F8+0	**3F8**	**[THR]**	**Trasmit Holding Reg.**	**Registro Buffer Trasmettitore**
3F8+0	**3F8**	**[RBR]**	**Receive Buffer Reg.**	**Registro Buffer Ricevitore**
3F8+0	**3F8**	**[DLL]**	**Divisor Latch LSB**	**Registro Divisore di Baud-Rate basso**
3F8+1	**3F9**	**[DLM]**	**Divisor Latch MSB**	**Registro Divisore di Baud-Rate alto**
3F8+1	**3F9**	**[IER]**	**Interrupt Enable Reg.**	**Registro Abilitazione Interruzione**
3F8+2	**3FA**	**[IIR]**	**Interrupt Identification Reg.**	**Registro Identificazione Interruzione**
3F8+3	**3FB**	**[LCR]**	**Line Control Reg.**	**Registro Controllo Linea**
3F8+4	**3FC**	**[MCR]**	**Modem Control Reg.**	**Registro Controllo Modem**
3F8+5	**3FD**	**[LSR]**	**Line Status Reg.**	**Registro Stato Linea**
3F8+6	**3FE**	**[MSR]**	**Modem Status Reg.**	**Registro Stato Modem**

Per la COM2 gli offset sono gli stessi partendo dall'indirizzo **2F8h**.

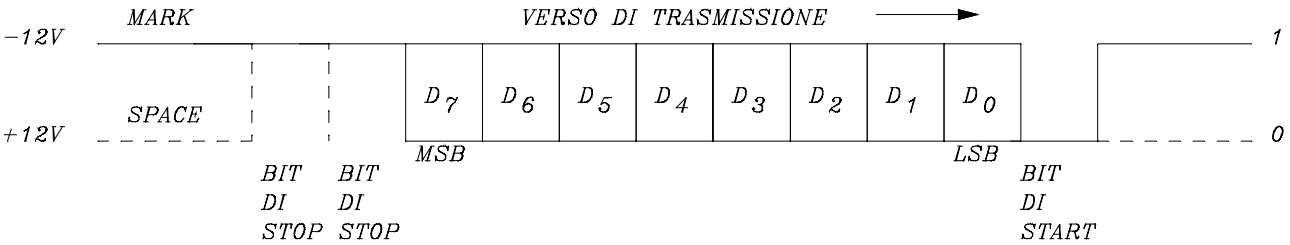

Fig.05

Ad esempio, in Fig.05 è stato impostato un formato di comunicazione seriale asincrona, in cui abbiamo, oltre al bit di start sempre presente, un dato a otto bit, nessun bit di parità e due bit di stop. L'inizializzazione dei registri dell'UART INS8250 deve essere tale da poter fornire questo formato ed anche la velocità di trasmissione in BAUD.
In fondo l'UART non è altro che un integrato che trasforma un dato parallelo in un dato seriale con determinate caratteristiche.Vediamo con **un esempio molto semplice** come si impostano i registri per una trasmissione ad interrogazione "polling" ossia ciclica (quindi senza l'intervento degli Interrupt o interruzioni) e senza nessun controllo di errori (N parità) sulla comunicazione.
Le specifiche imposte nell'esempio sono:

COM1, 1200baud, 8bit, 2stop, N parità

Innanzi tutto la linea e i connettori devono essere costruiti (Fig.06) in modo che sia il trasmettitore che il ricevitore si trovino sempre nelle condizioni normali di trasmissione e ricezione con DTR e RTS attivi con la conseguenza che anche DSR e CTS siano sempre attivi. ***Noi però possiamo ingannare i DTE collegando con ponticelli i piedini dei connettori come in Fig.06a in modo che un DTE sia sempre in condizione di comunicazione anche se l'altro è distaccato***. In questo caso parliamo di sistema **"autoavviante"**

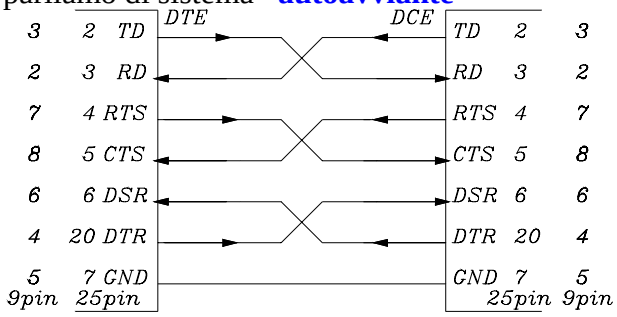

Fig 06

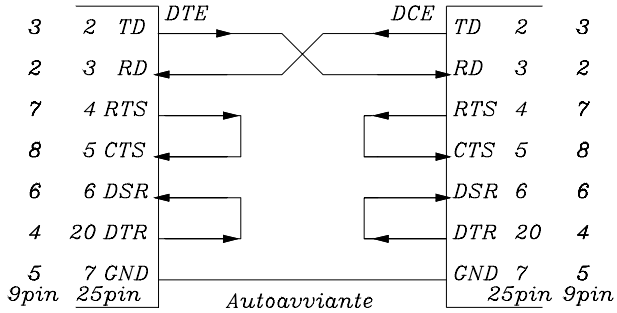

Fig.06a

Vediamo come si determina il Baud-Rate. Il ricevitore campiona il bit in arrivo al centro del suo tempo di presentazione, dividendo questa durata in sedici parti a cui corrispondono sedici cicli sinusoidali e testando il segnale all'ottavo ciclo (Fig.07).

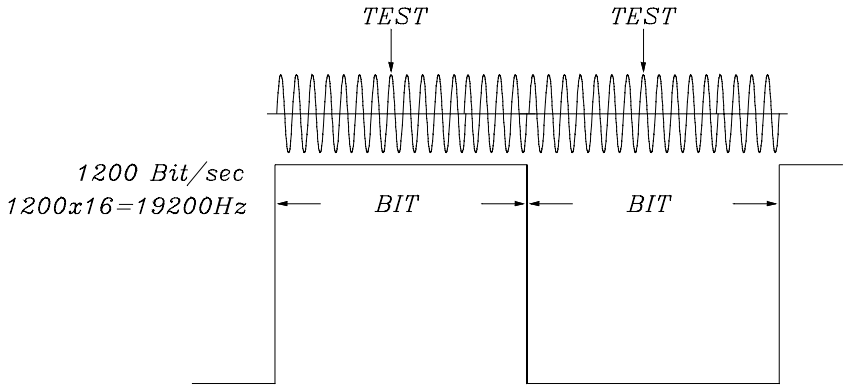

Fig.07

Normalmente la frequenza di clock dell'INS8250A (generata all'interno dell'elaboratore) viene posta al valore di **1,843200MHz.** Questa frequenza viene divisa tramite un divisore interno per 16, quindi in uscita dall'INS8250A abbiamo:

$$1843200/16=115200\text{Hz}$$

per cui il valore di divisione, per avere 1200 Bit/sec, è:

$$115200/1200 = 1843200/(16\cdot 1200) = \mathbf{96.}$$

La frequenza di campionamento, per quanto detto sopra, è sedici volte più grande di 1200 ossia:

$$16\cdot 1200 = 19200\text{Hz}$$

Il valore decimale **96d** corrisponde alla word di valore esadecimale **0060h.**
Ebbene, quando il **bit 7** del registro controllo linea **[LCR],** all'indirizzo 3FBh (3F8+3), è portato al valore "1", significa che si può impostare il Baud/Rate: nel Registro Divisore Alto **[DLM],**

all'indirizzo 3F9h, metteremo il byte alto **00h,** nel Registro Divisore Basso **[DLL],** all'indirizzo 3F8h metteremo il byte basso **60h.** Un frammento di programma in Borland Pascal 7 per determinare la velocità di 1200 baud può essere il seguente:

```
Port[LCR] := $80;              {[LCR]; 1000.0000 ---> b7=1:Baud Rate}
If Port[LCR] = $80 then
  Begin                    { $0060 --->96 ---> 1200: DLM=$00;DLL=$60}
    Port[IER]:=$00;                          {[IER]=$3F8+1:[DLM]=$00}
    Port[Com]:=$60;                            {[Com]=$3F8:[DLL]=$60}
  End;
```

Finita questa impostazione, il bit 7 di [LCR] deve essere riportato a **"0",** cioè imponiamo che:

```
Port[LCR] := $00;
```

Si impostano ora i restanti parametri sempre tramite il Registro Controllo Linea **[LCR],** ponendo i suoi bit al valore 0000.0111 che significano: Nparità, 2stop, 8bit:

```
Port[LCR]:=$07;       {[LCR]:b7=0 ---> 0000.0111=Nparita',2stop,8bit}
```

È utile osservare la Fig.07b dove vengono riportati i significati dei vari bit del registro **[LCR].**

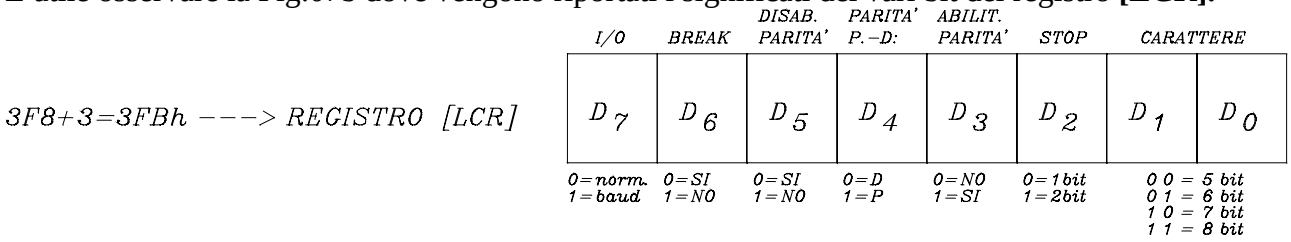

Fig.07b

Si può intuire quindi come la programmazione del registro **[LCR]** sia molto importante e decisiva per il buon funzionamento della RS232, perciò in questa fase è necessaria molta attenzione. In ogni caso conviene avere a disposizione e consultare il manuale dell'INS8250.

Si passa poi all'impostazione del protocollo di trasmissione tramite il Registro Controllo Modem **[MCR]** all'indirizzo 3FCh (Fig.07c), ponendo i suoi bit al valore 0000.0011, con il significato di RTS, DTR attivi, senza Interruzioni, non in circuito chiuso (osservare attentamente la Fig.07c).

In Pascal possiamo scrivere:

```
Port[MCR]:=$03;    {[MCR]:0000.0011--->UART non Loop,RTS,DTR attivi}
```

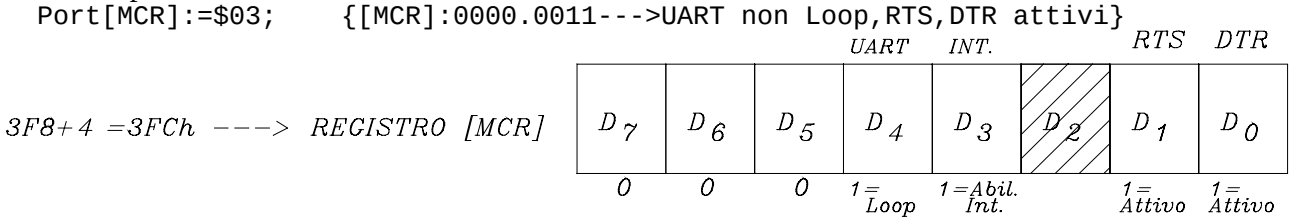

Fig.07c

Poiché la comunicazione è "polling" si deve azzerare il Registro di Abilitazione delle Interruzioni **[IER]** all'indirizzo 3F9h in modo da escludere l'apposito integrato (8259) che le gestisce:

```
Port[IER]:=$00;       {[IER]:interruzioni non abilitate = 8259 escluso}
```

Si controlla finalmente il Registro di Stato della Linea **[LSR]** all'indirizzo 3FDh (3F8+5), per vedere se è stato ricevuto un dato: il bit 0 deve essere "1" se il dato è arrivato, altrimenti la linea è libera e si può trasmettere ponendo il dato in [THR] all'indirizzo 3F8h.

In Fig.07d è riportato il Registro [LSR]. L'unico bit che ci interssa è il bit 0.

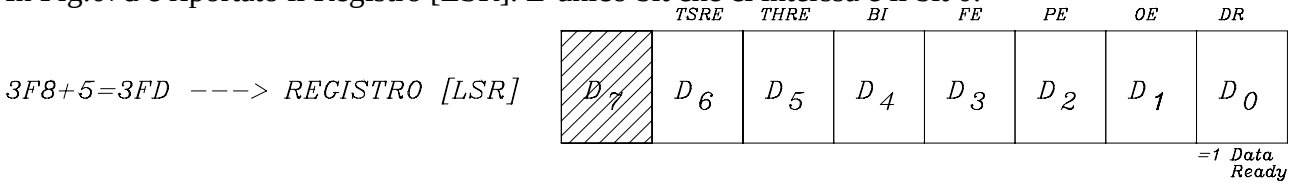

Fig.07d

In Pascal possiamo scrivere:

```
If (Port[LSR]) and $01 = 1 then     {[LSR]:se b0=1--->dato in linea}
Begin
  Rx                                        {procedura di ricezione}
End;
```

A questo punto l'INS8250A è inizializzato per comunicare con le specifiche prestabilite.

1843200:16=115200 115200:Baud=Div.

Baud	Div.	Esa.	Baud	Div.	Esa.
45	2560	0A00h	1200	96	0060h
50	2304	0900h	2400	48	0030h
57	2021	07E5h	4800	24	0018h
75	1536	0600h	9600	12	000Ch
100	1152	0480h	19200	6	0006h
110	1047	0417h	38400	3	0003h
300	384	0180h	57600	2	0002h
600	192	00C0h	115200	1	0001h

Fig.08

Nella tabella di Fig.08 sono riportati alcuni valori di Baud/Rate con i rispettivi divisori in decimale ed esadecimale, che saranno utili nella preparazione della Porta per le diverse velocità.

----*----

ESEMPIO DI PROGRAMMA DI GESTIONE DI UNA PORTA SERIALE

Qui di seguito è riportato un semplice programma didattico scritto per il ***Turbo-Assembly*** della Borland che realizza ciò che abbiamo detto finora.

Il collegamento seriale è tra due elaboratori: ciò che viene battuto sulla tastiera di uno di essi è letto sul video dell'altro. Il cavo di collegamento deve avere due terminali a spina femmina (**DTE/DTE**). Il tutto è disegnato in Fig.09 e realizzato secondo la configurazione della Fig.06a.

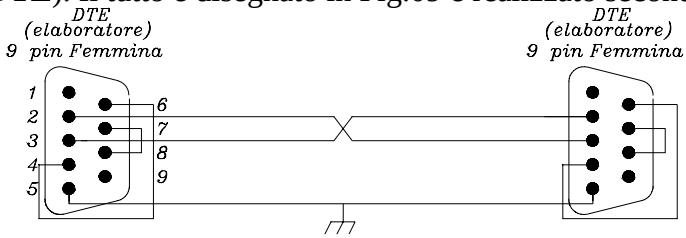

Fig.09: Cavo di collegamento a tre fili

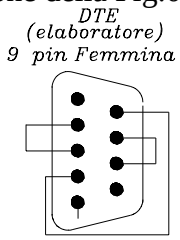

Fig.09a

Ovviamente lo stesso programma deve essere avviato su ambedue gli elaboratori. Ma si può adoperare un solo elaboratore utilizzando un solo connettore con il piedino 2 collegato al 3 come in Fig.09a. Si attua così un ritorno di segnale con l'effetto di stampare due volte sul video il simbolo del tasto premuto. In fondo è un metodo molto semplice per vedere se la porta funziona.

Poiché può verificarsi che la porta Com1 sia già impegnata da qualche altra periferica, abbiamo fatto richiesta di gestione della porta seriale ***Com2*** (2F8h) funzionante con il seguente protocollo:

Porta COM2 **02F8h**
Funzionamento: **Ciclico**
Baud Rate: **1200;**
Bit: **8**
Parità: **N**
Stop: **2**

Per inizializzare la Com1 basta modificare il valore di porta (da 2F8h a 3F8h).

Questo è il semplice programma in TurboAssembly* S232TR2.ASM *che realizza la nostra richiesta: Produce il file eseguibile S232TR2.EXE.

```
;************************************************
; Turbo Assembler - N.d.C.                      *
; S232TR2.ASM - Seriale COM2 : 02F8h            *
; Esperimento di Comunicazione "polling":       *
; --->Tx,Rx  tra due computer.                  *
; fxtal=1.8432MHz ---> 96d = 0060h              *
; freq.di campionamento per 1200 Baud:          *
; Fq=1843200/96=19200Hz                         *
;-----------------------------------------------*
; 5ªA - IPSIA "G.Marconi"-A.S.1996/97-Ortona(CH) *
; Ins.: prof.ing.Nicola del Ciotto              *
;************************************************
```

```
   DOSSEG
   .MODEL small
   .STACK 200h
   .DATA
   .CODE
;-------------------------------------------------------------------------
;**** inizializzazione dell'INS8250: COM2, 1200baud, 8bit, 2stop, Nparita'
;-------------------------------------------------------------------------
InitComSer  PROC NEAR
   mov dx,2FBh          ;[LCR]: Registro Controllo Linea
   mov al,80h           ;se bit 7=1: Baud rate 1200 (---> DLM, DLL: 0060h)
   out dx,al            ;esegui
   mov dx,2F8h          ;[DLL]: Reg.Divisore Baud basso
   mov al,60h           ;60h nel byte meno significativo (LBR)
   out dx,al            ;esegui
   mov dx,2F9h          ;[DLM]: Reg.Divisore Baud alto
   mov al,00h           ;00h nel byte più significativo (HBR)
   out dx,al            ;esegui
   mov dx,2FBh          ;[LCR]: Reg.Controllo Linea
   mov al,07h           ;00000111b: Nparita', 2stop, 8bit
   out dx,al            ;esegui
   mov dx,2FCh          ;impostazione [MCR]: Reg.Controllo Modem
   mov al,03h           ;00000011b: UART non in Loop, DTR,RTS attivi
   out dx,al            ;esegui
   mov dx,2F9h          ;[IER]: Reg.Identificazione interruzione
   mov al,00h           ;azzerato per polling (8259 escluso)
   out dx,al            ;esegui
   ret
InitComSer  ENDP
;=========================================================================
;.............Tx e Rx ad Interrogazione Ciclica (polling)................
;=========================================================================
;subroutine di ricezione
;-----------------------
Ric         PROC
   mov dx,2F8h          ;[RBR]:Buffer ricevitore (Rec.Buffer Reg.)
   in al,dx             ;ricezione carattere in AL da Port dx(=02F8h)
   mov ah,0Eh           ;visualizzazione "telescrivente"
   int 10h              ;tramite INT10h (BIOS)
   ret
Ric         ENDP
;--------------------------
;subroutine di trasmissione
;--------------------------
Trasm       PROC
   mov ah,01h           ;acquisizione carattere da tastiera (in AL)
   int 21h              ;tramite INT21h (DOS)
   mov dx,2F8h          ;[THR]: Buffer trasmettitore (Trasm. Holding Reg.)
   out dx,al            ;trasmissione carattere
   ret
Trasm       ENDP
;---------------------------------
;Controllo Trasmissione e Ricezione
;---------------------------------
ContrTxRx   PROC
P1:
   mov dx,2FDh          ;[LSR]: Registro Stato Linea
   in al,dx             ;controlla se è stato ricevuto un dato (b0=1)
   test al,1            ;controlla se b0 = 1 in [LSR]
   jz P2                ;altrimenti trasmissione
   call Ric             ;subroutine di ricezione
P2:
   mov ah,01h           ;#01:legge lo stato tastiera (premuto un tasto?)
   int 16h              ;INT16h del BIOS: in AL il carattere
   jz P1                ;se non è stato premuto un tasto, continua
   cmp al,1Bh           ;"ESC" (1Bh) termina la comunicazione
   je Fine
```

```
   call Trasm           ;subroutine di trasmissione
   jmp P1               ;continua il ciclo
Fine:
   ret
ContrTxRx   ENDP
;------------------------
;...programma di gestione
;------------------------
Inizio:
   mov  ax,@data
   mov  ds,ax
   call InitComSer
   call ContrTxRx
   mov  ah,4Ch          ;Funz. 4Ch di int21h: termina il programma
   int  21h
      END Inizio        ;fine
```

Il programma è solamente dimostrativo però funziona perfettamente.
Ha una particolarità di cui bisogna tener conto: la battuta del tasto [Invio] porta il cursore ad inizio riga con la sgradita conseguenza di distruggere il messaggio precedente quando si continua a battere sulla tastiera. Per evitare ciò vi è la necessità del salto riga che si ottiene con [Ctrl+Invio]. Evidentemente il tasto [Invio] produce il codice 0Dh (13d) che significa C.R. (Carriage Retutn ossia Ritorno Carrello della telescrivente) mentre [Ctrl]+[Invio] produce il codice 0Ah (10d) che significa L.F. (Line Feed ossia Salto Riga).

----*----

Qui di seguito è riportato invece un frammento di programma in Pascal in cui una Procedura, che raggruppa tutte le istruzioni già scritte, inizializza con molta semplicità e stringatezza la Porta seriale scelta sui valori più sopra assegnati. Questa Procedura la useremo nel Programma RTTY.

```
Procedure InitPort;
Begin
  Port[LCR] := $80;              {[LCR]; 1000.0000 ---> b7=1:Baud Rate}
  If Port[LCR] = $80 then
    Begin                     { $0060--->96 ---> 1200: DLH=$00;DLL=$60}
      Port[IER]:=$00;                                      {[DLM]=$00}
      Port[Com]:=$60;                                      {[DLL]=$60}
    End;
  Port[LCR]:=$07;       {[LCR]:b7=0 ---> 0000.0111=8bit,2stop,Nparita'}
  Port[MCR]:=$03;     {[MCR]:0000.0011--->UART non Loop,DTR,RTS attivi}
  Port[IER]:=$00;       {[IER]:interruzioni non abilitate=8259 escluso}
End;
```

Come si vede, non vi è nulla di veramente complicato.

----*----

Un semplice programma per la ricezione "RTTY" su video

Come applicazione utile di quanto detto finora sul collegamento seriale RS232, presento un semplice programma (RTTY.EXE) in Borland Pascal funzionante sotto DOS, adatto per la ricezione in Radiotelescrivente delle trasmissioni commerciali, d'informazione e amatoriali (RTTY = RadioTeleTYpe). È stato realizzato molti anni fa (1996) in ambiente scolastico, recuperato poi e rielaborato nel 2004 in occasione della riparazione di un ricevitore professionale (Plessey PR1553A), rivisto ancora e migliorato recentemente (2015) in occasione della scrittura di queste righe. Il programma trasforma il codice Baudot della RTTY nel Codice ASCII in modo da poter rappresentare la comunicazione su un comune video di elaboratore. Riesce a girare in ambiente WindowsXP ma non oltre.

Il programma è anche in grado di poter selezionare sia la Porta utile (Com1 o Com2) sia la velocità di trasmissione in Baud (45, 50, 57, 75, 100 Baud).

CODICE BAUDOT CCITT n°2

n (esa)	n (dec)	start	EMISSIONE I	II	III	IV	V	stop	lettere	cifre
00	0	○	○	○	○	○	○	●		
01	1	○	●	○	○	○	○	●	E	3
02	2	○	○	●	○	○	○	●	LF	LF
03	3	○	●	●	○	○	○	●	A	–
04	4	○	○	○	●	○	○	●	SPAZIO	SPAZIO
05	5	○	●	○	●	○	○	●	S	'
06	6	○	○	●	●	○	○	●	I	8
07	7	○	●	●	●	○	○	●	U	7
08	8	○	○	○	○	●	○	●	CR	CR
09	9	○	●	○	○	●	○	●	D	chi è?
0A	10	○	○	●	○	●	○	●	R	4
0B	11	○	●	●	○	●	○	●	J	camp.
0C	12	○	○	○	●	●	○	●	N	,
0D	13	○	●	○	●	●	○	●	F	°
0E	14	○	○	●	●	●	○	●	C	:
0F	15	○	●	●	●	●	○	●	K	(
10	16	○	○	○	○	○	●	●	T	5
11	17	○	●	○	○	○	●	●	Z	+
12	18	○	○	●	○	○	●	●	L	)
13	19	○	●	●	○	○	●	●	W	2
14	20	○	○	○	●	○	●	●	H	;
15	21	○	●	○	●	○	●	●	Y	6
16	22	○	○	●	●	○	●	●	P	0
17	23	○	●	●	●	○	●	●	Q	1
18	24	○	○	○	○	●	●	●	O	9
19	25	○	●	○	○	●	●	●	B	?
1A	26	○	○	●	○	●	●	●	G	%
1B	27	○	●	●	○	●	●	●	CIFRE	CIFRE
1C	28	○	○	○	●	●	●	●	M	.
1D	29	○	●	○	●	●	●	●	X	/
1E	30	○	○	●	●	●	●	●	V	=
1F	31	○	●	●	●	●	●	●	LETTERE	LETTERE

Fig.10

In Fig.10 è riportata la tabella del codice Baudot utilizzato per le radiocomunicazioni in Telescrivente. La tabella può essere utile per capire meglio il listato del programma.

È ovvio che per la ricezione è necessario avere a disposizione un adatto demodulatore FSK per telescrivente[1].

Questo è il listato sorgente del programma:

```
{*****************************************************************
 *       RTTY.PAS - Ricezione RTTY su RS232 – A.S.1996/97 - NdC     *
 *                     Com1: $3F8,  Com2: $2F8.                     *
 *            Per 50 Baud e': 1843200/16*50 = 2304(=$0900)          *
 *   Freq. di ripetizione: 25Hz; Freq. di rivelazione: 50*16=800Hz  *
 *          Codice BAUDOT: 5bit, 1stop, Nparita' (polling)          *
 *           IPSIA "G.Marconi" –ORTONA, 5ªT.I.E.E. Sez.A:           *
 *           Ins. Elettronica: prof.ing.Nicola del Ciotto           *
 *****************************************************************}
Program RTTYRic;   {Ricezione RTTY su video-Rev.:2004-Modif.:Set.2015}
Uses
  Crtn;                                        {Versione di Crt a 32 bit}
Var
  Com,IER,LCR,MCR,LSR :Integer;
  i,h,l,Q,P : Integer;
  a,bb : Char;
  PP : String;
  dato : byte;
  lettere : boolean;
{------------------Settaggio Registri UART ISN8250}----------------}
Procedure Porta;                {scelta della porta di comunicazione}
Begin
```

[1] È utile leggere l'articolo "Un demodulatore per telescrivente" sul sito "Le radio di Sophie", Pagine tecniche, Strumentazione da laboratorio.

```
 Write('Se [Com1] battere 1; se [Com2] battere 2:   '); ReadLn(P);
 If P = 1 then
 Begin
  Com := $3F8;
  PP := 'Com1';
 End Else
 If P =2 Then
 Begin
  Com := $2F8;
  PP := 'Com2';
 End;
 IER := Com+1;                                {Registro Interruzioni}
 LCR := Com+3;                               {Registro Controllo Linea}
 MCR := Com+4;                               {Registro Controllo Modem}
 LSR := Com+5;                                {Registro Stato Linea}
End;
{---------------------------------------------------------------------}
Procedure Velocita;      {determina la velocita' di ricezione in BAUD}
Begin
  Write('Quanti Baud? '); ReadLn(Q);
 If Q = 45 then                          {1843200/16*45 = 2560(=$0A00)}
  Begin
  h := $0A;
  l := $00;
  End else
 If Q = 50 then                          {1843200/16*50 = 2304(=$0900)}
  Begin
  h := $09;
  l := $00;
  End else
 If Q = 57 then                          {1843200/16*57 = 2021(=$07E5)}
  Begin
  h := $07;
  l := $E5;
  End else
 If Q = 75 then                          {1843200/16*75 = 1536(=$0600)}
  Begin
  h := $06;
  l := $00;
  End else
 If Q = 100 then                         {1843200/16*100= 1152(=$0480)}
  Begin
  h := 04;
  l := 80;
  End;
End;
{---------Inizializzazione Com: 50 baud,5bit,1stop,Nparita'----------}
Procedure InitPort(h,l:Integer);        {h,l definiscono la velocita'}
Begin
  Port[LCR] := $80;             {[LCR]; 1000.0000 ---> b7=1:Baud Rate}
  If Port[LCR] = $80 then
    Begin
      Port[IER]:=h;                           {per 50 baud:[DLH]=$09}
      Port[Com]:=l;                           {per 50 baud:[DLL]=$00}
    End;
  Port[LCR]:=$00;      {[LCR]:b7=0 ---> 0000.0000=5bit,1stop,Nparita'}
  Port[MCR]:=$03;    {[MCR]:0000.0011--->UART non Loop,DTR,RTS attivi}
  Port[IER]:=$00;      {[IER]:interruzioni non abilitate=8259 escluso}
End;
{---------------------------------------------------------------------}
Procedure LettereRx; {Codice Baudot trasformato in cod. ASCII-lettere}
Begin
   If dato = $00 then
     Begin
       dato := $00;Write(Char(dato));                  {00000    NUL}
     End Else
   If dato = $01 then write('E')                       {00001    E  }
```

```
    Else
   If dato = $02 then
     Begin
       dato := $0A;Write(Char(dato));                  {00010    LF }
     End Else
   If dato = $03 then write('A')                       {00011    A  }
    Else
   If dato = $04 then
     Begin
       dato := $20;Write(Char(dato));                  {00100    SP }
     End Else
   If dato = $05 then write('S')                       {00101    S  }
    Else
   If dato = $06 then write('I')                       {00110    I  }
    Else
   If dato = $07 then write('U')                       {00111    U  }
    Else
   If dato = $08 then
     Begin
       dato := $0D;Write(Char(dato));                  {01000    CR }
     End Else
   If dato = $09 then write('D')                       {01001    D  }
    Else
   If dato = $0A then write('R')                       {01010    R  }
    else
   If dato = $0B then write('J')                       {01011    J  }
    Else
   If dato = $0C then write('N')                       {01100    N  }
    Else
   If dato = $0D then write('F')                       {01101    F  }
    Else
   If dato = $0E then write('C')                       {01110    C  }
    Else
   If dato = $0F then write('K')                       {01111    K  }
    Else
   If dato = $10 then write('T')                       {10000    T  }
    Else
   If dato = $11 then write('Z')                       {10001    Z  }
    Else
   If dato = $12 then write('L')                       {10010    L  }
    Else
   If dato = $13 then write('W')                       {10011    W  }
    Else
   If dato = $14 then write('H')                       {10100    H  }
    Else
   If dato = $15 then write('Y')                       {10101    Y  }
    Else
   If dato = $16 then write('P')                       {10110    P  }
    Else
   If dato = $17 then write('Q')                       {10111    Q  }
    Else
   If dato = $18 then write('O')                       {11000    O  }
    Else
   If dato = $19 then write('B')                       {11001    B  }
    Else
   If dato = $1A then write('G')                       {11010    G  }
    Else
   If dato = $1B then                   {passaggio da lettere a cifre}
     Begin
       lettere := FALSE;                               {11011  cifre}
     End Else
   If dato = $1C then write('M')                       {11100    M  }
    Else
   If dato = $1D then write('X')                       {11101    X  }
    Else
   If dato = $1E then write('V');                      {11110    V  }
End;
```

```
{-------------------------------------------------------------------}
Procedure CifreRx;     {Codice Baudot trasformato in cod. ASCII-cifre}
  Begin
     If dato = $00 then
      Begin
       dato := $00;Write(Char(dato));                {00000      NUL}
      End Else
     If dato = $01 then Write('3')                  {00001        3}
      Else
     If dato = $02 then
     Begin
       dato := $0A;Write(Char(dato));                {00010       LF}
     End Else
     If dato = $03 then
     Begin
       dato := $2D;Write(Char(dato));                {00011        -}
     End Else
     If dato = $04 then
     Begin
       dato := $20;Write(Char(dato));                {00100       SP}
     End Else
     If dato = $05 then
     Begin
       dato := $60;Write(Char(dato));                {00101        '}
     End Else
     If dato = $06 then Write('8')                  {00110        8}
      Else
     If dato = $07 then Write('7')                  {00111        7}
      Else
     If dato = $08 then
     Begin
       dato := $0D;Write(Char(dato));                {01000       CR}
     End Else
     If dato = $09 then
     Begin
       dato := $05;                                 {01001      ENQ}
     End Else
     If dato = $0A then Write('4')                  {01010        4}
      Else
     If dato = $0B then
     Begin
       dato := $07;                                 {01011     BELL}
     End Else
     If dato = $0C then
     Begin
       dato := $2C;Write(Char(dato));                {01100        ,}
     End Else
     If dato = $0D then
     Begin
       dato := $21;Write(Char(dato));                {01101        !}
     End Else
     If dato = $0E then
     Begin
       dato := $3A;Write(Char(dato));                {01110        :}
     End Else
     If dato = $0F then
     Begin
       dato := $28;Write(Char(dato));                {01111        (}
     End Else
     If dato = $10 then Write('5')                  {10000        5}
      Else
     If dato = $11 then
     Begin
       dato := $2B;Write(Char(dato));                {10001        +}
     End Else
     If dato = $12 then
     Begin
```

```
       dato := $29;Write(Char(dato));                {10010        )}
      End Else
     If dato = $13 then Write('2')                  {10011        2}
      Else
     If dato = $14 then
     Begin
       dato := $3B;Write(Char(dato));               {10100        ;}
     End Else
     If dato = $15 then Write('6')                  {10101        6}
      Else
     If dato = $16 then Write('0')                  {10110        0}
      Else
     If dato = $17 then Write('1')                  {10111        1}
      Else
     If dato = $18 then Write('9')                  {11000        9}
      Else
     If dato = $19 then
     Begin
       dato := $3F;Write(Char(dato));               {11001        ?}
     End Else
     If dato = $1A then
     Begin
       dato := $25;Write(Char(dato));               {11010        %}
     End Else
     If dato = $1C then
     Begin
       dato := $2E;Write(Char(dato));               {11100        .}
     End Else
     If dato = $1D then
     Begin
       dato := $2F;Write(Char(dato));               {11101        /}
     End Else
     If dato = $1E then
     Begin
       dato := $3D;Write(Char(dato));               {11110        =}
     End Else
     If dato = $1F then                {passaggio da cifre a lettere}
     Begin
       lettere := TRUE;                             {11111  Lettere}
     End;
  End;
{-------------------------------------------------------------------}
Procedure Selezione;         {verifica se i dati sono lettere o cifre}
Begin
  Case lettere of
   TRUE:           {se e' presente $1F i dati successivi sono lettere}
    Begin
     LettereRx;
    End;
   FALSE:          {se e' presente $1B i dati successivi sono simboli}
    Begin
     CifreRx;
    End;
  End;
End;
{-----------Ricezione dato seriale dal demodulatore RTTY-------------}
Procedure Rx;
Begin
  dato := Port[Com];            {legge il dato sul buffer di ingresso}
  Selezione;                        {seleziona le lettere dai simboli}
End;
{-------------------------------------------------------------------}
Procedure Controllo;                                   {dato in linea}
Begin
 lettere := TRUE;                {si impone di partire con le lettere}
 Repeat
  If (Port[LSR]) and $01 = 1 then     {[LSR]:se b0=1--->dato in linea}
```

```
  Begin
    Rx
  End;
 Until KeyPressed;
End;
{-------------------------------------------------------------------}
Begin                                          {programma principale}
 ClrScr;
  h := 0; l := 0;                    {azzeramento parametri velocita'}
  WriteLn;
  Writeln(' STAMPA SU VIDEO DI MESSAGGI RTTY  -  * NdC-1996/1997 * ');
  WriteLn;
  WriteLn('-----------------------------------------------------------');
  WriteLn('Quale Porta?');
  Porta;
  WriteLn('Ricezione RTTY su Porta seriale ['+PP+'] ');
  WriteLn('-------------------------------------');
  WriteLn('Velocità di ricezione (45,50,57,75,100 Baud):');
   Velocita;
   WriteLn('-------------------------------------------');
   WriteLn;
   WriteLn('Inizio ricezione (Battere [INVIO] per interrompere)');
   WriteLn;
   InitPort(h,l);
   Port[LSR] := $00;      {controllo iniziale dello stato della linea}
   Controllo;                                {controllo dati in arrivo}
  Writeln;
  Write('Ricezione interrotta... Battere [INVIO] per uscire');
 ReadLn(bb);
End.
```

----*----

Il listato del programma può sembrare difficile, invece è solo lungo e concettualmente molto semplice. È necessario comunque dare un avvertimento sul suo buon funzionamento, in quanto all'inizio del listato non compare il consueto richiamo (Uses) alla **Unit Crt** originale della Borland ma ad una nuova unit chiamata **Unit Crtn**. Spiego perché. La Unit Crt della Borland è a sedici bit e funziona con CPU non più veloci di 250MHz (diciamo: fino al Pentium S e non oltre). Per CPU più veloci, purtroppo, quando si vuol far partire un programma, compare un riquadro con la scritta **"Division by zero error"** che blocca inesorabilmente la sua prosecuzione. Questo avviene per tutti i programmi che utilizzano in qualche modo lo schermo. Il problema sorge perché all'interno della Crt vi è la funzione "Delay" che scandisce il millisecondo e che deve poter dare sempre lo stesso valore per qualsiasi velocità della CPU presente. Tanto più è breve il periodo di clock della CPU tanto più deve essere grande il parametro "DelayCnt" per ottenere sempre lo stesso valore di "millisecondo". Ma, essendo DelayCnt a 16 bit, il suo valore non può essere superiore a 65535. Da qui la comparsa della scritta "Division by zero error", non per denominatore zero ma per numeratore superiore a 65535, irrealizzabile e quindi visto dalla macchina come valore infinito.
Perciò ho modificato la Unit Crt facendola funzionare in alcune parti a **32 bit**, aumentando così enormemente il valore limite di DelayCnt (4294836225) e dando luogo alla nuova versione **Crtn**. **Questa versione di Unit funziona con qualsiasi CPU** (dalla .386 in poi). Ovviamente la Crt originale non è stata modificata. Essa deve essere usata nei programmi con vecchie CPU che non supportano il linguaggio a 32 bit (.286, .186, 8086, 8088).

----*----

Appendice: Come modificare la Unit CRT

Il file **CRT.PAS**, che dà luogo alla Unit CRT.TPU, ha tutte le sue Procedure "external" ossia esterne. Esse provengono dal file oggetto CRT.OBJ formato dalla compilazione di CRT.ASM in codice Assembly. Perciò bisogna lavorare essenzialmente su questo file.

Ebbene, per i più esigenti o i più curiosi o i più interessati descrivo tutte le modifiche che ho apportato al listato sorgente della **CRT.ASM** per farla diventare **CRTN.ASM.** È sottinteso che bisogna avere il pacchetto del Borland Pascal 7 che contiene anche il Turbo Assembler TASM (ancora meglio se si ha il pacchetto "Professional", in cui si trovano i listati sorgenti delle unit).
Le modifiche e le aggiunte fatte sono evidenziate in neretto, con i commenti preceduti da due cancelletti (**##**). Per una migliore comprensione sono stati stilati brevi commenti a fianco delle istruzioni più importanti. Alcuni commenti in inglese sono stati tradotti in italiano. Come editor di testo è bene utilizzare lo stesso Editor del Pascal.

Aprire la Crt.asm e in successione:

Modificare:

```
      TITLE CRT
      INCLUDE     SE.ASM
```

in:

```
      TITLE CRTN
      INCLUDE     SE.ASM
```

.....

Modificare:

```
  ; Local workspace
  DelayCnt        DW    ?
  CurCrtSize      DW    ?
  NormAttr        DB    ?
  ScanCode        DB    ?
  BreakFlag       DB    ?
```

in:

```
  ; Spazio di lavoro locale
  DelayCnt        DD    ?     ;## deve essere una DoubleWord
  CurCrtSize      DW    ?
  NormAttr        DB    ?
  ScanCode        DB    ?
  BreakFlag       DB    ?
```

.....

Modificare:

```
Initialize:
      MOV   AH,0FH
      CALL  Video
      CMP   AL,7
      JE    @@1
      CMP   AL,3
      JBE   @@1
      MOV   AX,3
      CALL  CrtInit
@@1:  CALL  CrtSetup
      MOV   AH,8
      XOR   BH,BH
      CALL  Video
      MOV   AL,AH
      AND   AL,7FH
      MOV   NormAttr,AL
      MOV   TextAttr,AL
  XOR AX,AX
      MOV   CheckEOF,AL
      MOV   ScanCode,AL
      MOV   BreakFlag,AL
      INC   AX
      MOV   CheckBreak,AL
      MOV   ES,Seg0040
      MOV   DI,OFFSET Timer
      MOV   BL,ES:[DI]
@@2:  CMP   BL,ES:[DI]
      JE    @@2
```

```
      MOV   BL,ES:[DI]
      MOV   AX,-28
      CWD
      CALL  DelayLoop
      NOT   AX
      NOT   DX
      MOV   CX,55
      DIV   CX
      MOV   DelayCnt,AX
    IF DPMIVersion
      MOV   AX,dpmiGetRMCB
      MOV   SI,OFFSET CtrlBreak
      MOV   DI,OFFSET RealModeRegs
      PUSH  DS
      POP   ES
      PUSH  CS
      POP   DS
      INT   DPMI
      PUSH  ES
      POP   DS
      MOV   AX,dpmiSetRealInt
      MOV   BL,1BH
      INT   DPMI
    ELSE
      PUSH  DS
      PUSH  CS
      POP   DS
      MOV   DX,OFFSET CtrlBreak
      MOV   AX,dosSetInt*256+1BH
      INT   DOS
      POP   DS
    ENDIF
      RET
```

in:

```
Initialize:
      MOV   AH,0FH              ;legge la modalita' di visualizz. corrente
      CALL  Video               ;---> INT 10h
      CMP   AL,7                ;Mono
      JE    @@1
      CMP   AL,3                ;EGA/VGA
      JBE   @@1
      MOV   AX,3
      CALL  CrtInit
@@1:  CALL  CrtSetup
      MOV   AH,8                ;legge un carattere con i suoi attributi
      XOR   BH,BH               ;pagina zero dello schermo
      CALL  Video               ; AH --> 07h
      MOV   AL,AH               ;AH:Byte attributi; AL:car.ASCII letto
      AND   AL,7FH              ;AL=07h
      MOV   NormAttr,AL
      MOV   TextAttr,AL         ;07 = bianco su sfondo nero
      XOR   AX,AX               ;azzera AX
      MOV   CheckEOF,AL         ;=0
      MOV   ScanCode,AL         ;=0
      MOV   BreakFlag,AL        ;=0
      INC   AX                  ;=1
      MOV   CheckBreak,AL       ;=1
      MOV   ES,Seg0040          ;ES = 0040h
      MOV   DI,OFFSET Timer     ;Timer --> EQU (DWORD PTR 6Ch)
      MOV   BL,ES:[DI]          ;trova il valore di [DI]
@@2:  CMP   BL,ES:[DI]          ;confontalo con il precedente
      JE    @@2                 ;salta se e' ancora uguale (entro i 55ms)
      MOV   BL,ES:[DI]          ;parte il conteggo quando [DI] e' cambiato
      MOV   AX,-28              ;AX = FFE4h = 65508
      CWD                       ;word in dword: AX=FFE4h, DX=FFFFh
      CALL  DelayLoop           ;chiama la sub. di ritardo per impostare
                                ;DelayCnt tramite successiva divisione per CX.
```

```
      NOT   AX                  ;complementa AX bit per bit
      NOT   DX                  ;complementa DX bit per bit
      .386                      ;## direttiva Assembler 80386 a 32 bit
      ror eax,16                ;## porta in alto i primi 16 bit in EAX
      mov ax,dx                 ;## carica in basso di EAX il registro DX
      ror eax,16                ;## ruota ancora per avere DX alto e AX basso in
                                ;## EAX
      mov edx,0                 ;## azzera EDX per sicurezza
      mov ecx,55                ;## poni il Tick in ECX
      div ecx                   ;## dividi per il Tick
      mov [DelayCnt],eax        ;## il quoziente EAX in [DelayCnt]
      .8086                     ;## ripristina l'assembler 8086 a 16 bit
    IF DPMIVersion
      MOV   AX,dpmiGetRMCB
      MOV   SI,OFFSET CtrlBreak
      MOV   DI,OFFSET RealModeRegs
      PUSH  DS
      POP   ES
      PUSH  CS
      POP   DS
      INT   DPMI
      PUSH  ES
      POP   DS
      MOV   AX,dpmiSetRealInt
      MOV   BL,1BH
      INT   DPMI
    ELSE
      PUSH  DS
      PUSH  CS
      POP   DS
      MOV   DX,OFFSET CtrlBreak
      MOV   AX,dosSetInt*256+1BH
      INT   DOS
      POP   DS
    ENDIF
      RET
.....................................................................
```

Modificare:

```
; Delay specified number of milliseconds
Delay:
      MOV   BX,SP
      MOV   CX,SS:[BX+4]
      JCXZ  @@2
      MOV   ES,Seg0040
      XOR   DI,DI
      MOV   BL,ES:[DI]
@@1:  MOV   AX,DelayCnt
      XOR   DX,DX
      CALL  DelayLoop
      LOOP  @@1
@@2:  RETF  2
```

in:

```
; Ritarda di uno specificato numero di millisecondi
Delay:
      MOV   BX,SP                     ;carica in BX lo Stack Point
      MOV   CX,SS:[BX+4]              ;valore di CX all'indir. SS:[BX+4]
      JCXZ  @@2                       ;salta se CX = 0
      MOV   ES,Seg0040
      XOR   DI,DI                     ;DI=0 punta all'inizio di 0040h
      MOV   BL,ES:[DI]                ;dove e' il valore fisso F8h
@@1:
      mov  ax,WORD PTR DelayCnt       ;## carica in AX e DX (32 bit)
      mov  dx,WORD PTR DelayCnt+2     ;## la DWORD DelayCnt
      CALL  DelayLoop                 ;chiama il ciclo di ritardo
      LOOP  @@1                       ;continua finche' CX = 0
```

```
@@2:  RETF  2
```
. . . . . .

Non vi è altro da fare. La nuova CRTN.ASM deve essere ora salvata e compilata mediante il TASM (Turbo Assembler) per ottenere il file oggetto: **CRTN.OBJ (non usare Tlink!).** Nella directory di lavoro è necessaria la presenza del file SE.ASM. Durante la compilazione compaiono dei "warning" (avvisi) che si riferiscono a SE.ASM che però non sono importanti. Non bisogna tenerne conto poiché il file CRTN.OBJ viene comunque creato in modo corretto.
Infine, nella **CRT.PAS,** in ambiente Pascal, dopo essere certi che CRTN.OBJ stia nella cartella "Bin", bisogna eseguire le seguenti semplici operazioni.

Modificare:

Unit Crt;

in:

Unit Crtn;

. . . . . .

Modificare:

```
implementation
{$IFDEF DPMI}
{$L CRT.OBP}
{$ELSE}
{$L CRT.OBJ}
{$ENDIF}
```

in:

```
IMPLEMENTATION   { modificata la CRT.ASM in CRTN.ASM }
{$IFDEF DPMI}
{$L CRTN.OBP}
{$ELSE}
{$L CRTN.OBJ}
{$ENDIF}
```
. . . . . .

Dopo le modifiche salvare il file CRT.PAS con il nuovo nome **CRTN.PAS** in modo da poterlo **compilare** come **CRTN.TPU** (usare "Compile" non "Run"!)**.** Questo è tutto.

Molti hanno abbandonato il Pascal 7 perché, installato su elaboratori più recenti con CPU veloci, mostrava all'avvio dei programmi la scritta fatale "Division by zero error", senza nessuna possibilità di rimedio. Con la Unit **CRTN.TPU** ciò non avviene più e si può lavorare in Pascal sotto DOS fino a WindowsXP.
Sono disponibili alcune versioni di Pascal 7 free. Le ho provate con risultati negativi, purtroppo, perché mi hanno dato gravi problemi per la compilazione dei miei programmi praticamente incompatibili. Perciò preferisco ancora utilizzare il mio vecchio e sicuro Borland Pascal 7 con la CRT modificata in CRTN, fin quando potrò.

----*----

Qui termino, nella speranza che a qualcuno possa interessare, per qualche motivo, quanto ho scritto. Questi brevi appunti, sebbene incompleti, su una porta nata nel 1969, ormai storica ma che in certe situazioni è ancora oggi molto valida, potrebbero essere utili come traccia di lavoro.

Ing. Nicola del Ciotto

Ortona, Novembre 2015