I Manuali di Giobe2000

PORTA SERIALE

HW **Quarta Parte**

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

Dentro il Sistema	Porta Seriale

- Dentro il Sistema - Area di Comunicazione Bios

1.1 Variabili di Sistema per la Porta Seriale

- Le *variabili di sistema* (*locazioni* dell'**Area di Comunicazione BIOS**) associate alle **porte seriali** sono elencate qui di seguito.

0400H - Indirizzo Base Porta Seriale

- Le *4 variabili di sistema* poste da 0000:0400H sono l'**interfaccia software** usata dal *Sistema Operativo* (o dai nostri programmi) per sapere **quante** *porte seriali* sono presenti nel sistema; il loro contenuto è naturalmente predisposto dalla **Procedura POST** nelle fasi preliminari che seguono l'accensione del computer.

0000:0400H	2 bytes	Indirizzo Base Porta Seriale n°1 (**COM1**/porta **0**)
0000:0402H	2 bytes	Indirizzo Base Porta Seriale n°2 (**COM2**/porta **1**)
0000:0404H	2 bytes	Indirizzo Base Porta Seriale n°3 (**COM3**/porta **2**)
0000:0406H	2 bytes	Indirizzo Base Porta Seriale n°4 (**COM4**/porta **3**)

- Subito dopo l'accensione del computer la **Procedura POST** scrive nei *registri dato* di ciascuna *possibile* porta seriale, tentandone subito dopo la lettura: se il dispositivo restituirà il medesimo valore scrittovi in precedenza manifesterà palesemente la sua presenza nel sistema.
- Ciascuna voce della tabella contiene l'**Indirizzo Base** (il primo di ciascun gruppo) dei **Registri** associati alle porte individuate nel sistema; oppure il numero **0000**H se la porta cercata non c'è.
- Nei moderni computer è molto probabile che le 4 *word* inserite in tabella siano, in sequenza: **03F8H**, **02F8H**, **0000H**, **0000H**, a indicare la presenza di 2 sole porte seriali, la **COM1** o porta**0** e la **COM2** o porta**2**.
- L'eventuale presenza delle altre 2 porte seriali riconoscibili dalla procedura POST (la **COM3** o porta **2** e la **COM4** o porta **3**) lascerebbe in tabella la sequenza **03F8H**, **02F8H**, **03E8H**, **02E8H**.

0410H - Lista dell'hardware istallato nel sistema (Equipment List)

- La *variabile di sistema* posta a 0000:0410H, pur in grado di fornire informazioni su molti dispositivi eventualmente presenti sul Computer (stampanti parallele o seriali, adattatore per giochi (joystick), numero di porte seriali e di lettori di dischetti, quantità di memoria RAM, presenza del processore matematico, ecc.) destina **3 bit** anche per le ***porte seriali***.

0000:0410H	2 bytes	Lista dell'hardware istallato nel sistema (Equipment List)

- La lettura di queste 2 locazioni è, naturalmente, molto facile ma non bisogna dimenticare che la logica dei processori precede il byte meno significativo in 0000:0410H e quello più significativo (che interessa a noi) in 0000:0411H.

- In particolare, la terna bit**11**, bit**10**, bit**9** esprime il numero (binario) di porte seriali presenti nel Sistema:
  - **000** nessuna porta seriale
  - ...
  - **111** presenti 7 porte seriali
- Tutte queste informazioni sono predisposte durante la fase di inizializzazione (power-up test, POST) subito dopo l'accensione del Computer, ma possono essere assunte anche chiamando l'interfaccia BIOS con **INT 11H**.

047CH - Valore di Time Out Porta Seriale

- Le *4 variabili di sistema* poste da 0000:047CH contengono valori significativi nel contesto delle *porte seriali*:

0000:047CH	1 byte	Valore di Time Out Porta Seriale n°1 (**COM1** o porta **0**)
0000:047DH	1 byte	Valore di Time Out Porta Seriale n°2 (**COM2** o porta **1**)
0000:047EH	1 byte	Valore di Time Out Porta Seriale n°3 (**COM3** o porta **2**)
0000:047FH	1 byte	Valore di Time Out Porta Seriale n°4 (**COM4** o porta **3**)

- Il valore numerico presente in queste locazioni rappresenta, per ciascuna delle porte seriali presenti nel sistema, un numero **proporzionale** ai tentativi esercitati dall'UART a livello BIOS di lettura del **Registro** di Ricezione Dati [**port_8**] con la Funzione **02**H di **INT 14H** o del **Registro** di Stato della Linea [**port_D**] con la Funzione **03**H di **INT 14H**; in entrambi i casi se l'operazione è andata a buon fine il bit**7** (errore di *TimeOut*) del registro **AH** è lasciato a **0**; viceversa, se il bit**7** è trovato a **1** i rimanenti *sette* bit di **AH** riveleranno la causa dell'insuccesso, come indicato nel seguente dettaglio:

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Valore lasciato in AH - Stato della Linea
							1	1 = Received Data Ready o Data Available: *nuovo dato* trasferito dal *registro a scorrimento* RSR al RBR [port_8] o nel FIFO in Ricezione (16550A)
						1		1 = ricezione gravata da errore *di sovrapposizione* (Overrun Error)
					1			1 = ricezione gravata da errore *di parità* (Parity Error)
				1				1 = ricezione gravata da errori *di composizione* (Framing Error)
			1					1 = rilevato un *segnale di Break* (Break Interrupt)
		1						1 = THR [port_8] o FIFO in Trasmissione (16550A) *vuoto* (Transmitter Holding Register Empty)
	1							1 = *non ci sono dati da trasmettere*, *registro* TSR e THR [port_8] o FIFO in Trasmissione (16550A) *vuoti* (Data Holding Register Empty)
1								1 = errore di *timeout*

- Subito dopo l'accensione del computer la procedura **POST** scrive in queste locazioni il valore **1**; nei primi computer (PC XT) tale numero poteva essere ancora inteso come *tempo in secondi* da attendere (ritenendo di consumare circa un secondo per ogni tentativo)
- L'avvento di computer veloci (AT, ATX) ha ridicolizzato questo numero per cui, per poter disporre di un tempo paragonabile è necessario *moltiplicarlo per una costante piuttosto grande*, ma l'operazione non è documentata.

4 - Programmazione - Livello Hardware

1.1 Controllo della Porta Seriale a livello Hardware

- La programmazione **a livello hardware** di una *porta seriale* può sembrare impegnativa rispetto alle altre, ma rimane l'unica veramente affidabile e in grado di gestirla in modo affidabile e completo; la programmazione **a livello DOS** è *pressocchè inesistente* e quella **a livello BIOS** è *limitata* e *insufficiente*.
- Per la corretta programmazione **a basso livello** di una porta seriale è comunque saggio aver letto ogni argomento descritto nelle pagine numerosissime precedenti; la conoscenza dei dettagli gratifica la mente e aiuta a capire.
- Nonostante sia già disponibile l'indice della barra marrone di navigazione (a sinistra) desidero ricordarti la strada percorsa fin ora [se fai click sui link si apriranno pagine alternative, dalle quali riprendere gli argomenti selezionati]:
  - gli **obiettivi della comunicazione seriale** (la comunicazione seriale, il futuro, le regole, l'interfaccia, i compiti i principali Standard, i dettagli, le modalità)
  - i dettagli dello **standard RS232C**, punto di riferimento assoluto della comunicazione seriale asincrona (struttura dati, forma del segnale, specifiche elettriche, convertitori di livello, velocità dell'informazione, modulazione e compressione dati, funzione dei segnali standard, protocollo seriale, controllo hardware e software del flusso dati)

- Connettori e Cavi
- i dettagli del dispositivo UART, punto di riferimento assoluto delle interfacce seriali (compiti, storia, pin out, schemi)
- i dettagli sui Registri del dispositivo UART

- Un cenno merita anche la possibilità di configurare le 2 porte seriali presenti (di norma) sul computer come una qualunque delle 4 porte COM potenzialmente riconoscibili, riassegnandole a livello di **Pannello di controllo** (in *Windows*) o con il il *comando* **MODE** (in *DOS*) o direttamente a livello **CMOS setup**, previste dalla scheda madre dei moderni computer.
- In questo modo è facile evitare qualche *eventuale* conflitto, per esempio se si desidera usare un *Modem Interno* su **COM1** o **COM2**, affidando le porte seriali standard (sulla scheda madre) ad altre porte **COM**.
- Come per le altre periferiche il numero di porte seriali è limitato da quello disponibile degli **IRQ** (interrupt) e degli **indirizzi di I/O**, di solito distribuito con equità tra i vari concorrenti alle risorse del processore.
- Non di rado 2 dispositivi possono **condividere** lo stesso interrupt funzionando correttamente se non vi accedono contemporaneamente: questo è possibile anche per 2 **COM**, spesso associate alla stessa linea di interruzione, o **IRQ3** o **IRQ4** (vedi Tecniche d'Interruzione).

BIOS	Programmazione	Porta Seriale

- Programmazione - Livello BIOS

1.1 Controllo della Porta Seriale a livello BIOS

- Anche la ***porta seriale*** può essere gestita **a Livello BIOS** con il supporto delle Funzioni dell'**INT 14H**; il loro utilizzo sottolinea la *filosofia* di questo tipo di programmazione: pur non scendendo al livello dei registri ne virtualizza la presenza: *non è necessario* conoscere il loro indirizzo o i meccanismi minimi di funzionamento, *basta chiamare la* ***funzione*** *ed essa pensa a tutto*!
- Sfortunatamente nella maggior parte dei casi il loro uso può diventare *critico*, consigliando comunque la programmazione con **accesso diretto** ai **Registri** della *porta seriale* (**UART**) se sono necessarie velocità maggiori di **1200** baud, a causa dell'eccessivo tempo necessario per la loro esecuzione.
- In aggiunta le **Funzioni BIOS** non sono in grado di riconoscere le *richieste di servizio d'interruzione*, attivate dalla *porta* in occasione di *eventi seriali* particolari (come la ricezione o la trasmissione di un carattere o il rilevamento di un errore); in questi casi il ricorso ai **Registri UART** è assolutamente indispensabile.
- Il pacchetto di **procedure BIOS** per le ***porte seriali*** alle 4 Funzioni dell'**INT 14H** previste dal BIOS originale ne aggiunge altre 2, nate a supporto del computer IBM PS2, destinate alla *gestione estesa*; sono tutte raccolte dalla seguente Tabella:

INT 14H	Funzione **00**H	Inizializza i parametri della porta seriale
INT 14H	Funzione **01**H	Trasmette un carattere a una porta seriale
INT 14H	Funzione **02**H	Riceve un carattere da una porta seriale
INT 14H	Funzione **03**H	Restituisce lo stato di una porta seriale
INT 14H	Funzione **04**H	Inizializzazione estesa di una porta seriale (PS2)
INT 14H	Funzione **05**H	Controllo esteso di una porta seriale (PS2)

- Ciascuna Funzione **BIOS** sarà trattata in dettaglio, qui di seguito.

INT 14H - Funzione 00H - Inizializza i parametri della porta seriale

- **Inizializza** una *porta seriale* trasmettendole i parametri previsti dallo standard **RS232** per il *frame seriale*, cioè Numero di **bit di dato** (lunghezza della parola), Numero di **bit di stop**, Tipo di **parità** e velocità (**Baud rate**).
- Ogni possibile *richiesta di servizio d'interruzione* è disabilitata.
- Consente di programmare solo velocità *relativamente basse*: si può notare che la velocità di **19200** baud, piuttosto diffusa nell'utilizzo di una porta seriale, non è supportata da questa Funzione; per una versione più potente consultare la Funzione **04**H, adatta (se disponibile) a soddisfare le elevate velocità dei moderni modem.

- In ingresso:
  - **AH** è posto a **01H**
  - **AL** è posto al valore wwppsddB, in accordo con la seguente tabella:

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	AL - Parametri di ricetrasmissione
						d	d	Numero di **bit di dato** (lunghezza della parola): 00=**5**, 01=**6**, 10=**7**, 11=**8**
					s			Numero di **bit di stop**: 0=**1**, 1=**2** (**1,5** con 5 bit di dato)
			p	p				Tipo di **parità**: x0=**nessuna**, 01=**parità dispari**, 11=**parità pari**
v	v	v						**Baud rate** (in baud) 000= **110**, 001= **150**, 010= **300**, 011= **600** 100=**1200**, 101=**2400**, 110=**4800**, 111=**9600**

  - **DX** indica il numero Bios della porta seriale:
    - 0000=**COM1**
    - 0001=**COM2**
    - 0002=**COM3**
    - 0003=**COM4**
- In uscita lascia in:
  - **AH** il valore dello **Stato della Linea**, del tutto simile al contenuto del **Registro** di Stato della Linea [**port_D**]:
  - **AL** il valore dello **Stato del Modem**, del tutto simile al contenuto del **Registro** di Stato del Modem [**port_E**]:
  - Poiché il contenuto dei registri è lo stesso di quello restituito dalla Funzione **03**H, consultala per saperne di più.

INT 14H - Funzione 01H - Trasmette un carattere a una porta seriale

- **Trasmette** un **carattere** a una *porta seriale*; la trasmissione avviene non appena il **Registro** di Trasmissione [**port_8**] (Transmitter Holding Register, **THR**) (o il **buffer FIFO** in Trasmissione, con **UART 16550A**) è ***vuoto***.
- In ingresso:
  - **AH** è posto a **01H**
  - **AL** viene predisposto con il carattere da trasmettere
  - **DX** indica il numero Bios della porta seriale:
    - 0000=**COM1**
    - 0001=**COM2**
    - 0002=**COM3**
    - 0003=**COM4**
- In uscita lascia in:
  - **AH** il valore **00H** se l'operazione è andata a buon fine; in questo caso owiamente il bit**7** è lasciato a **0** e, viceversa, se è trovato a **1** i 7 bit rimanenti riveleranno la causa dell'insuccesso, riflesso del contenuto del **Registro** di Stato della Linea [**port_D**]
  - In caso di insuccesso per avere un *rapporto* ***completo*** sulle possibili *cause d'errore* è ***necessario consultare*** il valore dei registri restituito dalla Funzione **03**H; sebbene i valori trovati da bit**6** a bit**0** siano gli stessi, solo in questo modo si potrà disporre della segnalazione di *timeout*, non disponibile in questa Funzione per la necessità di usare il bit**7** come generico *segnalatore d'errore*

INT 14H - Funzione 02H - Riceve un carattere da una porta seriale

- **Riceve** un **carattere** da una *porta seriale*; la ricezione avviene non appena ***un nuovo dato*** è stato completamente ricostruito dal *registro a scorrimento* interno (Receive Shift Register, **RSR**) e da esso è stato trasferito nel **Registro** di Ricezione [**port_8**] (Receive Buffer Register, **RBR**) (o nel **buffer FIFO** in Ricezione, con **UART 16550A**).
- In ingresso:
  - **AH** è posto a **02H**
  - **DX** indica il numero Bios della porta seriale:
    - 0000=**COM1**
    - 0001=**COM2**
    - 0002=**COM3**
    - 0003=**COM4**

- In uscita lascia in:
  - **AL** il carattere ricevuto
  - **AH** il valore **00H** se l'operazione è andata a buon fine; in questo caso ovviamente il bit**7** è lasciato a **0** e, viceversa, se è trovato a **1** i 7 bit rimanenti riveleranno la causa dell'insuccesso, riflesso del contenuto del **Registro** di Stato della Linea [**port_D**]
  - In caso di insuccesso per avere un *rapporto* ***completo*** sulle possibili *cause d'errore* è ***necessario*** *consultare* il valore dei registri restituito dalla Funzione **03**H; sebbene i valori trovati da bit**6** a bit**0** siano gli stessi, solo in questo modo si potrà disporre della segnalazione di *timeout*, non disponibile in questa Funzione per la necessità di usare il bit**7** come generico *segnalatore d'errore*

INT 14H - Funzione 03H - Restituisce lo stato di una porta seriale

- **Restituisce** lo **stato** della *linea seriale* e *modem* per un determinato UART (*porta seriale*).
- In ingresso:
  - **AH** è posto a **03H**
  - **DX** indica il numero Bios della porta seriale:
    - 0000=**COM1**
    - 0001=**COM2**
    - 0002=**COM3**
    - 0003=**COM4**
- In uscita lascia in:
  - **AH** il valore dello **Stato della Linea**, del tutto simile al contenuto del **Registro** di Stato della Linea [**port_D**]
  - **AL** il valore dello **Stato del Modem**, del tutto simile al contenuto del **Registro** di Stato del Modem [**port_E**]
  - In dettaglio il contenuto dei registri **AH** e **AL** è proposto qui di seguito:

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Valore lasciato in AH - Stato della Linea
							1	1 = Received Data Ready o Data Available: *nuovo dato* trasferito dal *registro a scorrimento* RSR al RBR [port_8] o nel FIFO in Ricezione (16550A)
						1		1 = ricezione gravata da errore *di sovrapposizione* (Overrun Error)
					1			1 = ricezione gravata da errore *di parità* (Parity Error)
				1				1 = ricezione gravata da errori *di composizione* (Framing Error)
			1					1 = rilevato un *segnale di Break* (Break Interrupt)
		1						1 = THR [port_8] o FIFO in Trasmissione (16550A) *vuoto* (Transmitter Holding Register Empty)
	1							1 = *non ci sono dati da trasmettere*, *registro* TSR e THR [port_8] o FIFO in Trasmissione (16550A) *vuoti* (Data Holding Register Empty)
1								1 = funzione fallita (*timeout*)

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Valore lasciato in AL - Stato del Modem
							1	1 = la linea $\overline{\text{CTS}}$ (Delta Clear to Send) ha *cambiato livello logico* dopo l'ultima lettura
						1		1 = la linea $\overline{\text{DSR}}$ (Delta Data Set Ready) ha *cambiato livello logico* dopo l'ultima lettura
					1			1 = la linea $\overline{\text{RI}}$ (Trailing Edge Ring Indicator) ha *cambiato livello logico* dopo l'ultima lettura
				1				1 = la linea $\overline{\text{CD}}$ (Delta Carrier Detec) ha *cambiato livello logico* dopo l'ultima lettura
			x					*livello logico* presente sulla linea $\overline{\text{CTS}}$ (Clear to Send) nel momento della lettura
		x						*livello logico* presente sulla linea $\overline{\text{DSR}}$ (Data Set Ready) nel momento della lettura
	x							*livello logico* presente sulla linea $\overline{\text{RI}}$ (Ring Indicator) nel momento della lettura
x								*livello logico* presente sulla linea $\overline{\text{CD}}$ (Data Carrier Detect) nel momento della lettura

INT 14H - Funzione 04H - Inizializzazione estesa di una porta seriale (PS2)

- **Inizializzazione *estesa*** di una *porta seriale*; si tratta di una versione più potente della Funzione **00**H, adatta a soddisfare le elevate velocità dei moderni modem.
- Sfortunatamente questa Funzione non è supportata da tutti i BIOS.
- In ingresso:
  - **AH** è posto a **04H**
  - **AL** indica la presenza dello stato di **break**: **00H**=nessun break, **01H**=break
  - **BH** indica il tipo di **parità**: **00H**=nessuna, **01H**=parità dispari, **02H**=parità pari, **03H**=parità bloccata dispari, **04H**=parità bloccata pari
  - **BL** indica il numero di **bit di stop**: **00H**=1, **01H**=2 (**1,5** con 5 bit di dato)
  - **CH** indica il numero di **bit di dato** (lunghezza della parola): **00H**=5, **01H**=6, **02H**=7, **03H**=8
  - **CL** indica il valore della velocità **Baud rate** (in baud), solo se sono supportate dal sistema:
    00H= 110, **01H= 150**, **02H= 300**, **03H= 600**, **04H=1200**, **05H=2400**, **06H=4800**, **07H=9600**, **08H=19200**
    Se ComShare è installato: **00H= 19200**, **01H= 38400**, **02H= 300**, **03H= 14400**, **04H=1200**, **05H=2400**, **06H=28800**, **07H=9600**, **08H=19200**, **09H=38400**, **0AH=57600**, **0BH=115200**
  - **DX** indica il numero Bios della porta seriale: ● 0000=**COM1** ● 0001=**COM2** ● 0002=**COM3** ● 0003=**COM4**
- In uscita lascia in:
  - **AH** il valore dello **Stato della Linea**, del tutto simile al contenuto del **Registro** di Stato della Linea [**port_D**]
  - **AL** il valore dello **Stato del Modem**, del tutto simile al contenuto del **Registro** di Stato del Modem [**port_E**]
  - Poiché il contenuto dei registri è lo stesso di quello restituito dalla Funz.**03**H, consultala per saperne di più

INT 14H - Funzione 05H - Controllo esteso di una porta seriale (PS2)

- **Controllo *esteso*** di una *porta seriale*; consente di *leggere* o di *scrivere* il **Registro** Controllo del Modem [**port_C**].
- Sfortunatamente questa Funzione non è supportata da tutti i BIOS.
- In ingresso:
  - **AH** è posto a **05H**
  - **AL** è posto a **00H** se si desidera leggere il registro; è posto a **01H** se si desidera scrivere il registro
  - **BL** valore da scrivere nel **Registro** di Controllo del Modem [**port_C**], vedi Tabella in fondo alla pagina
  - **DX** indica il numero Bios della porta seriale: ● 0000=**COM1** ● 0001=**COM2** ● 0002=**COM3** ● 0003=**COM4**
- In uscita lascia in:
  - **AH** il valore dello **Stato della Linea**, del tutto simile al contenuto del **Registro** di Stato della Linea [**port_D**]
  - **AL** il valore dello **Stato del Modem**, del tutto simile al contenuto del **Registro** di Stato del Modem [**port_E**]
  - Poiché il contenuto dei registri è lo stesso di quello restituito dalla Funz.**03**H, consultala per saperne di più
  - **BL** valore letto dal **Registro** di Controllo del Modem [**port_C**], descritto in dettaglio dalla seguente Tabella:

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Valore lasciato o letto in BL - Controllo del Modem
							1	1 = *attiva* (mette a 0 logico) la linea DTR (Data Terminal Ready, pin33 in uscita dall'UART) e quindi pone una *tensione positiva* (ON, SPACE, tra +25V e +3V) sul pin4/DB9 o sul pin20/DB25 del connettore; il segnale DTR è di solito attivato dal DTE per *avvisare* il DCE che è *regolarmente collegato alla linea di comunicazione* ed è *pronto a trasmettere o ricevere dati*
						1		1 = *attiva* (mette a 0 logico) la linea RTS (Request To Send, pin32 in uscita dall'UART) e quindi pone una *tensione positiva* (ON, SPACE, tra +25V e +3V) sul pin7/DB9 o sul pin4/DB25 del connettore; il segnale RTS è di solito attivato dal DTE per *avvisare* il DCE che *dispone di dati* ed è *pronto a trasmetterglieli*
					0			0 = condizioni normali 1 = *attiva* (mette a 0 logico) la linea OUT1 (*uscita ausiliaria* di *uso generale*, *attiva bassa*, disponibile sul pin34 dell'UART); si tratta di una linea non utilizzata nei PC IBM e compatibili
				0				0 = condizioni normali (*interruzioni* da parte dell'UART *disabilitate*) 1 = *abilitare* le *interruzioni* da parte dell'UART cioè, *attivando* (0 logico) la linea OUT2 (pin31 in uscita dall'UART), collega la linea INTR (pin30 dell'UART) alla linea IRQ3/IRQ4 del *controller delle interruzioni*
			0					0 = condizioni normali 1 = pone la *porta seriale* in Lookback Mode cioè *collega tra loro* i *registri interni* di *Trasmissione* e di *Ricezione* e le *linee* di *controllo Modem*, RTS con CTS, DTR con DSR , OUT1 con RI e OUT2 con CD; ogni dato erogato può essere immediatamente letto
0	0	0						riservati, non utilizzati, sempre a 0 logico

DOS	Programmazione	Porta Seriale

- Programmazione - Livello DOS

1.1 Controllo della Porta Seriale a livello DOS

- Il DOS associa in nome **COM** ad ognuno dei 4 indirizzi diversi da **0000**H trovato a partire dalla locazione 0000:0400H, ivi inseriti dall'esecuzione della procedura **POST**, subito dopo l'accensione del computer; nei moderni computer è molto probabile trovare la sequenza: **03F8H**, **02F8H**, **0000H**, **0000H**, ad indicare la presenza di 2 sole *porte seriali*, la **COM1** e la **COM2**.
- Il DOS è comunque in grado di supportare anche le porte **COM3** e **COM4**, qualora ne venga rilevata la presenza.
- Per usare una *porta seriale* è possibile usare il *comando DOS* **MODE**, tipico e funzionante ***esclusivamente*** nell'ambiente *DOS*: nel momento in cui altri sistemi operativi (o altri programmi di comunicazione) prendono il *controllo diretto* della *porta seriale* ogni valore predisposto con **MODE** viene sovrascritto e perduto.
- La sintassi del *comando DOS* **MODE** indica la porta a cui ci si vuole riferire, la velocità (**Baud rate** in baud), il Numero di **bit di dato** (lunghezza della parola), il Numero di **bit di stop** e il Tipo di **parità**, in accordo con i parametri previsti dallo standard **RS232** per il *frame seriale*.
- Per esempio il comando **MODE COM1: 9600,N,8,1,P** programma la porta seriale **COM1** per lavorare a **9600** baud, **N**essuna parità, **8** bit di dato e **1** bit di stop; l'ultimo parametro (**P**) ripete automaticamente la connessione in caso di timeout.
- Questo comando può essere digitato direttamente dal *prompt* del *DOS* oppure può essere inserito tra le righe del file di sistema Autoexec.bat (anch'esso non operativo in ambiente Windows).
- La programmazione **a Livello DOS** delle ***Porte seriali*** è possibile ma **poco raccomandabile** e **poco affidabile**; le Funzioni disponibili sono una *virtualizzazione* della ***vera*** *porta seriale* e quindi del tutto impreparate a gestire le complesse procedure di ricetrasmissione di un UART:
  - per la *trasmissione* si dovrebbe far riferimento come minimo alla Funzione **01**H dell'**INT 14H** o meglio all'accesso diretto al **Registro** di Trasmissione [**port_8**] (Transmitter Holding Register, **THR**) (o il **buffer FIFO** in Trasmissione, con **UART 16550A**)
  - per la *ricezione* si dovrebbe far riferimento come minimo alla Funzione **02**H dell'**INT 14H** o meglio all'accesso diretto al **Registro** di Ricezione [**port_8**] (Receive Buffer Register, **RBR**) (o nel **buffer FIFO** in Ricezione, con **UART 16550A**)
- Ecco comunque la raccolta delle possibili Funzioni dell'**INT 21H**:

INT 21H	Funzione **03**H	Legge un carattere al dispositivo standard ausiliario, **COM1**
INT 21H	Funzione **04**H	Scrive un carattere verso il dispositivo standard ausiliario, **COM1**
INT 21H	Funzione **3F**H	Legge da un dispositivo generico (device), con la tecnica dei gestori (handle) standard
INT 21H	Funzione **40**H	Scrive verso un dispositivo generico (device), con la tecnica dei gestori (handle) standard
INT 21H	Funzione **44**H/02H	Legge una stringa di bytes da un dispositivo di tipo "carattere", prelevandoli da un buffer
INT 21H	Funzione **44**H/03H	Scrive una stringa di bytes in un dispositivo di tipo "carattere", inserendoli in un buffer
INT 21H	Funzione **75**H	Accede ad una porta seriale (non documentato)

- Ciascuna Funzione **DOS** sarà trattata in dettaglio, qui di seguito.

INT 21H - Funzione 03H - Legge un carattere al dispositivo standard ausiliario, COM1

- **Riceve** un **carattere** dal *dispositivo* ***standard*** *ausiliario*, **AUX:**, solitamente inteso proprio come **COM1:**; con il *comando DOS* **MODE**, descritto nella pagina precedente, è possibile affidare al dispositivo ausiliario anche la porta **COM2**.

.

- Si tratta di una Funzione **poco raccomandabile** e **poco affidabile** essendo, come si può intuire, una *virtualizzazione* della ***vera*** *porta seriale*, del tutto impreparata a tener conto della complessa struttura del ricevitore di un **UART**; alle normali velocità di ricezione i dati in arrivo saranno certamente perduti.

- Per la ricezione dei caratteri si dovrebbe far riferimento come minimo alla Funzione **02**H dell'**INT 14H** o meglio all'accesso diretto al **Registro** di Ricezione [**port_8**] (Receive Buffer Register, **RBR**) (o nel **buffer FIFO** in Ricezione, con **UART 16550A**).
- In ingresso:
  - **AH** è posto a **03H**
- In uscita lascia in:
  - **AL** valore letto dal *dispositivo **standard** di ausiliario*, **AUX:**, **COM1:**

INT 21H - Funzione **04**H - Scrive un carattere verso il dispositivo standard ausiliario, **COM1**

- **Trasmette** un **carattere** al *dispositivo **standard** ausiliario*, **AUX:**, solitamente inteso proprio come **COM1:**; con il *comando DOS* **MODE**, descritto in una pagina precedente, è possibile affidare al dispositivo ausiliario anche la porta **COM2**.
- Si tratta di una Funzione **poco raccomandabile** e **poco affidabile** essendo, come si può intuire, una *virtualizzazione* della ***vera** porta seriale*, del tutto impreparata a tener conto della complessa struttura del trasmettitore di un UART.
- Per la trasmissione dei caratteri si dovrebbe far riferimento come minimo alla Funzione **01**H dell'**INT 14H** o meglio all'accesso diretto al **Registro** di Trasmissione [**port_8**] (Transmitter Holding Register, **THR**) (o il **buffer FIFO** in Trasmissione, con **UART 16550A**).
- In ingresso:
  - **AH** è posto a **03H**
  - **DL** è posto il dato da trasmettere
- Non restituisce alcuna informazione.

INT 21H - Funzione **3FH** - Legge da **COM1** con la tecnica dei gestori (handle) standard

- Si tratta di una delle filosofie del **DOS**, che ama identificare i suoi *interlocutori* come *dispositivi generici* (*device*), mediante un gestore (***handle***) numerato; il penultimo tra quelli previsti (*handle **3***) è associato proprio al *dispositivo **standard** ausiliario*, **AUX:**, solitamente inteso proprio come **COM1:**; con il *comando DOS* **MODE**, descritto nella pagina precedente, è possibile affidare al dispositivo ausiliario anche la porta **COM2**.
- Nel caso specifico delle *porte seriali* sembra ***piuttosto improbabile*** *"creare"* (Funzione **3C**H) o *"aprire"* (Funzione **3D**H) o *"chiudere"* (Funzione **3E**H) una *porta seriale* (o meglio il *dispositivo standard* n° **0003**), mentre ***è possibile*** virtualmente *"leggere"* (Funzione **3F**H) o *"scrivere"* (Funzione **40**H) una *porta seriale*.
- Si tratta di una Funzioni **poco raccomandabile** e **poco affidabile**, una *virtualizzazione* della ***vera** porta seriale* e quindi del tutto impreparate a gestire le complesse procedure di ricetrasmissione di un **UART**: meglio far riferimento alla Funzione **02**H dell'**INT 14H** o accedere direttamente al **Registro** di Ricetrasmissione [**port_8**].
- In ingresso:
  - **AH** è posto a **3FH**
  - **BX** è posto a **0003H** per indicare che il gestore (*handle*) coinvolto è *handle **3***, associato al dispositivo standard ausiliario, ***seriale***, **AUX**
  - **CX** indica il numero di bytes da leggere
  - **DS:DX** indica l'indirizzo logico dell'area di memoria (buffer) in cui mettere i bytes letti
- In uscita lascia:
  - se **ffC**=0 lascia in **AX** il *numero di bytes effettivamente letti*, consentendo una eventuale verifica a posteriori
  - se **ffC**=1 lascia in **AX** un *codice d'errore* sufficiente per risalire alla causa del mancato funzionamento.
- L'effetto di questa Funzione è sostanzialmente identico a quello della SottoFunzione **02**H della Funzione **44**H

INT 21H - Funzione **40**H - Scrive verso **COM1** con la tecnica dei gestori (handle) standard

- Si tratta di una delle filosofie del **DOS**, che ama identificare i suoi *interlocutori* come *dispositivi generici* (*device*), mediante un gestore (***handle***) numerato; il penultimo tra quelli previsti (*handle **3***) è associato proprio al *dispositivo **standard** ausiliario*, **AUX:**, solitamente inteso proprio come **COM1:**; con il *comando DOS* **MODE**, descritto nella pagina precedente, è possibile affidare al dispositivo ausiliario anche la porta **COM2**.
- Nel caso specifico delle *porte seriali* sembra ***piuttosto improbabile*** *"creare"* (Funzione **3C**H) o *"aprire"* (Funzione **3D**H) o *"chiudere"* (Funzione **3E**H) una *porta seriale* (o meglio il *dispositivo standard* n° **0003**), mentre ***è possibile*** virtualmente *"leggere"* (Funzione **3F**H) o *"scrivere"* (Funzione **40**H) una *porta seriale*.

- Si tratta di una Funzioni **poco raccomandabile** e **poco affidabile**, una *virtualizzazione* della ***vera*** *porta seriale* e quindi del tutto impreparate a gestire le complesse procedure di ricetrasmissione di un **UART**: meglio far riferimento alla Funzione **01**H dell'**INT 14H** o accedere direttamente al **Registro** di Ricetrasmissione [**port_8**].
- In ingresso:
  - **AH** è posto a **40H**
  - **BX** è posto a **0003H** per indicare che il gestore (*handle*) coinvolto è *handle* **3**, associato al dispositivo standard ausiliario, ***seriale***, **AUX**
  - **CX** indica il numero di bytes da scrivere
  - **DS:DX** indica l'indirizzo logico dell'area di memoria (buffer) da cui prelevare i bytes da scrivere
- In uscita lascia:
  - se **ffC**=0 lascia in **AX** il *numero di bytes effettivamente scritti*, consentendo una eventuale verifica a posteriori
  - se **ffC**=1 lascia in **AX** un *codice d'errore* sufficiente per risalire alla causa del mancato funzionamento.
- L'effetto di questa Funzione è sostanzialmente identico a quello della SottoFunzione **03**H della Funzione **44**H

INT 21H - Funzione **44**H - SottoFunz. **02**H - Legge una stringa di bytes da **COM1** prelevandoli da un buffer

- **Legge** una stringa di bytes da un *dispositivo di tipo "carattere",* assumendoli da un buffer; nel caso specifico di una *porta seriale* il **DOS**, in accordo con la sua filosofia di identificare gli *interlocutori* come *dispositivi generici* (*device*) mediante con un gestore (***handle***) numerato, la riconosce come *dispositivo standard* n° **0003**, **AUX**, associandole il penultimo tra i gestori previsti (*handle* **3**).
- Si tratta di una Funzione **poco raccomandabile** e **poco affidabile**, una *virtualizzazione* della ***vera*** *porta seriale* e quindi del tutto impreparate a gestire le complesse procedure di ricetrasmissione di un **UART**: meglio far riferimento alla Funzione **02**H dell'**INT 14H** o accedere direttamente al **Registro** di Ricetrasmissione [**port_8**].
- In ingresso:
  - **AH** è posto a **44H**
  - **AL** è posto a **02H**
  - **BX** è posto a **0003H** per indicare che il gestore (*handle*) coinvolto è *handle* **3**, associato al dispositivo standard ausiliario, ***seriale***, **AUX**
  - **CX** indica il numero di bytes da leggere
  - **DS:DX** indica l'indirizzo logico dell'area di memoria (buffer) in cui mettere i bytes letti
- In uscita lascia:
  - se **ffC**=0 lascia in **AX** il *numero di bytes effettivamente letti*, consentendo una eventuale verifica a posteriori
  - se **ffC**=1 lascia in **AX** un *codice d'errore* sufficiente per risalire alla causa del mancato funzionamento.
- L'effetto di questa SottoFunzione è sostanzialmente identico a quello della Funzione **3F**H.

INT 21H - Funzione **44**H - SottoFunz. **03**H - Scrive una stringa di bytes verso **COM1** inserendola in un buffer

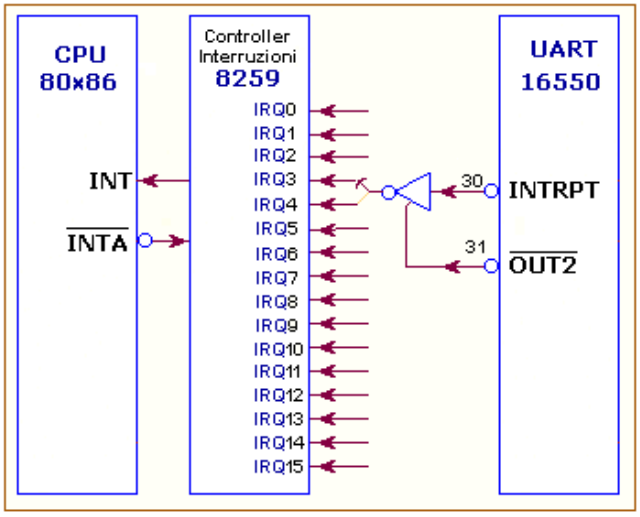
- **Scrive** una stringa di bytes in un *dispositivo di tipo "carattere",* depositandoli in un buffer; nel caso specifico di una *porta seriale* il **DOS**, in accordo con la sua filosofia di identificare gli *interlocutori* come *dispositivi generici* (*device*) mediante con un gestore (***handle***) numerato, la riconosce come *dispositivo standard* n° **0003**, **AUX**, associandole il penultimo tra i gestori previsti (*handle* **3**).
- Si tratta di una Funzione **poco raccomandabile** e **poco affidabile**, una *virtualizzazione* della ***vera*** *porta seriale* e quindi del tutto impreparate a gestire le complesse procedure di ricetrasmissione di un **UART**: meglio far riferimento alla Funzione **01**H dell'**INT 14H** o accedere direttamente al **Registro** di Ricetrasmissione [**port_8**].
- In ingresso:
  - **AH** è posto a **44H**
  - **AL** è posto a **03H**
  - **BX** è posto a **0003H** per indicare che il gestore (*handle*) coinvolto è *handle* **3**, associato al dispositivo standard ausiliario, ***seriale***, **AUX**
  - **CX** indica il numero di bytes da scrivere
  - **DS:DX** indica l'indirizzo logico dell'area di memoria (buffer) da cui prelevare i bytes da scrivere
- In uscita lascia:
  - se **ffC**=0 lascia in **AX** il *numero di bytes effettivamente scritti*, consentendo una eventuale verifica a posteriori
  - se **ffC**=1 lascia in **AX** un *codice d'errore* sufficiente per risalire alla causa del mancato funzionamento.
- L'effetto di questa SottoFunzione è sostanzialmente identico a quello della Funzione **40**H.

INT	Programmazione	Porta Seriale

- Programmazione - Controllo degli Eventi Seriali

1.1 generalità

- La **possibilità di interrompere** il processore per *chiedere la sua attenzione* è un meccanismo di particolare importanza.
- Di fatto ogni CPU dispone di una linea d'ingresso *attiva alta* molto particolare, **INT**, da esso testata **al termine di *ogni* istruzione** che è chiamato ad eseguire; in questo modo la possibilità di rilevarla *attiva* è piuttosto elevata: chiunque la metta a **1 logico** forzerà il processore ad interrompere la sua normale attività obbligandolo ad eseguire una particolare procedura di servizio.



- Poichè la linea prevista per questo scopo è unica fin dai primi computer è stato previsto un componente specializzato (**PIC 8259**, Programmable Interrupt Controller) ad accettare fino a 8 richieste di interruzione, **IRQ** (Interrupt ReQuest), ben presto raddoppiato (con l'avvento dei computer di tipo AT) per la sempre più pressante necessità di queste preziose linee.
- Attualmente il **PIC** accetta fino a **15** diversi **IRQ**, utilizzando la linea **IRQ2** come collettore delle IRQ da 8 a 15; uno dei suoi compiti è quello di associare una precisa **priorità** a ciascuna di esse, fornendo al processore il codice binario della linea che ha richiesto servizio; in ogni ciascuna linea IRQ attiva forzerà un **1 logico** sulla linea **INT** del processore che, se abilitato a rispondere, provvederà al servizio richiesto e porterà bassa la line **INTA**.
- Nella gestione di una *porta seriale* sono numerosi gli ***eventi*** che possono *richiedere l'attenzione* del processore; il problema del programmatore è decidere in che modo dare loro soddisfazione; esistono 2 possibilità:
  - la ***tecnica del polling*** (*interrogazione*): ogni evento viene segnalato dall'**UART** con la modifica di alcuni bit nei suoi 2 Registri di Stato, lasciando il compito di verificarne lo stato logico alla autonoma volontà del processore
  - la ***tecnica di interruzione***: quando l'***evento seriale*** si manifesta l'**UART** stesso provvede a segnalarlo al processore attivando la sua linea d'uscita **INTRPT** (pin**30** dell'**UART**), a sua volta connessa alla linea **IRQ3** o **IRQ4** del gestore delle interruzioni

- Programmazione - Tecnica in Polling

1.2 Gestione degli Eventi Seriali con la tecnica del Polling

- L'attenzione del processore può essere richiesta da ***eventi*** legati alla *ricetrasmissione* (stato della linea) o al valore dei *segnali di handshake* (stato del modem); ogni variazione viene segnalata dall'**UART** direttamente portando a **1** alcuni bit del **Registro** di Stato della Linea [**port_D**] o del **Registro** di Stato del Modem [**port_E**], consultabili anche rispettivamente nei registri CPU **AH** e **AL**, dopo la chiamata della Funzione **03**H dell'**INT 14H**.

- Con riferimento ai bit del **Registro** di Stato della Linea [**port_D**] o del registro CPU **AH**, dopo la chiamata della Funzione **03**H dell'**INT 14H**, la gestione in ***polling*** della *ricetrasmissione seriale* può essere monitorizzata interrogando:
  - il bit**0** (**Received Data Ready** o **Data Available**): se è trovato a **1** ***un nuovo dato*** è stato completamente ricostruito dal *registro a scorrimento* interno (Receive Shift Register, **RSR**) e da esso è stato trasferito nel **Registro** di Ricezione [**port_8**] (Receive Buffer Register, **RBR**) (o nel **buffer FIFO** in Ricezione, con **UART 16550A**); questo bit è riportato a **0** logico non appena il processore estrae il dato dal **RBR** (o dal **FIFO**) oppure se il processore azzera il contenuto del **FIFO** in Ricezione
  - il bit**5** (**Transmitter Holding Register Empty**: se è trovato a **1** il **Registro** di Trasmissione [**port_8**] (Transmitter Holding Register, **THR**) (o il **buffer FIFO** in Trasmissione, con **UART 16550A**) è ***vuoto***: il dato in esso *scritto* in precedenza dal processore è stato trasferito al *registro a scorrimento* interno, **Transmit Shift Register**, **TSR**; l'**UART** è pronto ad accettare nuovi caratteri da trasmettere e, non appena uno di essi entra nel **THR** (o nel **FIFO**), riporta a **0** logico questo bit

- Con riferimento allo stesso **Registro**, la gestione in ***polling*** degli *errori in ricezione* può essere monitorizzata interrogando:
  - il bit**1** (**Overrun Error**): se è trovato a **1** se la ricezione è gravata da **errore** *di sovrapposizione*: con **UART 8250/16450** il processore non ha fatto in tempo a leggere un dato dal **Registro** di Ricezione [**port_8**] (Receive Buffer Register, **RBR**, prima dell'arrivo del successivo, oppure (con **UART 16550**) il **buffer FIFO** in Ricezione è pieno e il dato ricostruito nel *registro a scorrimento* interno (Receive Shift Register, **RSR**) non può esservi inserito; il dato non letto viene perduto, sovrascritto da quello in arrivo
  - il bit**2** (**Parity Error**): se è trovato a **1** se la ricezione è gravata da **errore** *di parità*: prima del bit di stop l'**UART** ha rilevato nel dato ricevuto un numero di *bit a **1*** diverso da quello (*pari* o *dispari*) previsto dalla programmazione del **Registro** di Controllo Linea [**port_B**]
  - il bit**3** (**Framing Error**): se è trovato a **1** se la ricezione è gravata da **errori** *di composizione*: se, dopo l'ultimo bit di dato previsto dalla programmazione del **Registro** di Controllo Linea [**port_B**], viene rilevato uno **0** logico la struttura (*frame*) del dato ricevuto è scorretta perchè priva del bit di stop (notoriamente un bit a **1**)
  - il bit**4** (**Break Interrupt**): se è trovato a **1** se sulla linea d'ingresso seriale (**SIN**, pin**10** dell'**UART**) è presente un *segnale di **Break***, cioè se la linea è tenuta a livello logico **0** (**SPACE**) per un tempo maggiore a quello previsto per ricevere un dato formattato (cioè completo di bit di start, eventuale parità e di stop); nel **Registro** di Ricezione [**port_8**] (o nel **FIFO** in Ricezione, con **UART 16550**) è inserito un byte nullo. Questo *segnale* può essere generato dal *computer remoto* per richiedere attenzione e la ricezione riprenderà solo quando la linea **SIN** torna a livello logico **1**, in attesa del bit di start (notoriamente un bit a **0**) del nuovo dato in arrivo

- Con riferimento ai bit del **Registro** di Stato del Modem [**port_E**] o del registro CPU **AL**, dopo la chiamata della Funzione **03**H dell'**INT 14H**, la gestione in ***polling*** delle variazioni dei *segnali di handshake* può essere monitorizzata interrogando:
  - il bit**0** (**Delta Clear to Send** (**Clear to Send has changed**): se è trovato a **1** la linea **$\overline{\text{CTS}}$** (Clear To Send, pin**36** UART) ha *cambiato livello logico* **dopo** l'ultima lettura del registro
  - il bit**1** (**Delta Data Set Ready**: (**Data Set Ready has changed**)): se è trovato a **1** la linea **$\overline{\text{DSR}}$** (Data Set Ready, pin**37** UART) ha *cambiato livello logico* **dopo** l'ultima lettura del registro
  - il bit**2** (**Trailing Edge** of **Ring Indicator**): se è trovato a **1** la linea **$\overline{\text{RI}}$** (Ring Indicator, pin**39** UART) ha *cambiato livello logico* **dopo** l'ultima lettura del registro
  - il bit**3** (**Delta Data Carrier Detect**): se è trovato a **1** la linea **$\overline{\text{CD}}$** (Data Carrier Detect, pin**38** UART) ha *cambiato livello logico* **dopo** l'ultima lettura del registro

- Questa tecnica ha alcuni **gravi difetti**, sostanzialmente in ricezione più che in trasmissione:
  - obbliga il processore a *numerosi tempi morti*, in contrasto con i *brevi istanti* necessari per *dare servizio* alla richiesta
  - se la ricezione dei dati è *veloce* (basta una velocità *superiore a 2400 bps*!) il tempo necessario per la verifica del bit di stato, tra un polling e il successivo, potrebbe essere sufficiente per provocare la perdita di qualche dato
  - in aggiunta, senza qualche artificio, c'è il rischio (molto frequente) di vedersi *bloccare* il computer se il processore dovesse stare in attesa di dati seriali *che non arrivano*...
- Per questo, sebbene sia sicura e semplice, è consigliabile farne *uso meditato* e *consapevole del contesto* in cui si opera, oppure affidarsi alla tecnica di gestione sotto interrupt.
- In ogni caso le **tecniche di Polling** sono descritte *molto dettagliatamente* nella parte che si occupa dei **sorgenti ASM** dedicati alla porta seriale.

- Programmazione - Tecnica di Interrupt

1.3 Gestione degli Eventi Seriali con la tecnica dell'Interrupt

- Fin dalla sua comparsa la comunicazione seriale (asincrona) è stata affidata a 2 *porte seriali* (**UART**), per le quali il DOS ha previsto queste assegnazioni:
  - **COM1**: Indirizzi da **03F8**H a **03FF**H, interrupt **IRQ 4**
  - **COM2**: Indirizzi da **03E8**H a **03EF**H, interrupt **IRQ 3**
- La possibilità (a partire dalla versione DOS 3) di avere altre 2 *porte seriali* ha richiesto la necessità di nuove risorse:
  - **COM3**: Indirizzi da **02F8**H a **02FF**H, interrupt **IRQ 4**
  - **COM4**: Indirizzi da **02E8**H a **02EF**H, interrupt **IRQ 3**
- Poichè le linee di interruzione hardware *non sono* molto numerose a ciascuna delle 2 previste sono associate 2 **COM**; la *condivisione* non produrrà *conflitti* se le 2 porte non chiederanno servizio contemporaneamente, cosa verosimile visto che dipende dal programma di gestione (che si guarderà bene dal farlo...).
- Naturalmente gli ***eventi seriali*** sono gli stessi gestiti, nelle pagine precedenti, con la tecnica del ***polling***; quando uno uno di essi *richiede attenzione* l'**UART** stesso prowede a segnalarlo al processore attivando la sua linea d'uscita **INTRPT** (pin**30** dell'**UART**), a sua volta connessa alla linea **IRQ3** o **IRQ4** del gestore delle interruzioni.
- Questa tecnica è più complessa della precedente ma anche **altamente efficiente**.
- Poichè la *richiesta di interruzione* può pervenire contemporaneamente *anche da* **più di un** *evento*, per semplificarne il *servizio* sono stati raggruppati in ***4*** *categorie*, associando loro una **priorità**, cioè un numero che stabilisce quale di esse sarà servita per prima:
  - massima priorità per **Receiver Line Status**: durante la ricezione di dati è stata rilevata la presenza di **errori** (*di sovrapposizione*, *di parità* o *di composizione*) o la presenza di un **segnale** *di break*
  - seconda priorità per **Received Data Available** (***dato pronto***): il numero di bytes ricevuti ha superato quello massimo (*trigger level*) previsto per il **buffer FIFO** in Ricezione (si aspetta che il **FIFO** sia *pieno* di dati, per farli leggere *in blocco* dal processore), oppure quando il **FIFO** contiene bytes in misura inferiore al massimo ma il tempo concesso a nuovi arrivi è terminato, *timeout*) oppure quando nel **Receive Buffer Register**, **RBR**, è pronto un singolo dato (con **UART 8250/16450** o con **UART 16550A**, se il **FIFO** è disabilitato)
  - terza priorità per **Transmit Data Empty** (***trasmettitore vuoto***): il **Transmit Holding Register**, **THR,** è *vuoto* (con **UART 8250/16450** o con **UART 16550A**, se il **FIFO** in Trasmissione è disabilitato) oppure quando il **FIFO** ha posti liberi per uno o più bytes
  - priorità più bassa per **Modem Status**: durante la comunicazione con il *Modem*, è stata rilevata la variazione di segnali in arrivo, direzione **DCE>DTE**, come Data Carrier Detect (**CD**, rilevato modem remoto/possibile comunicare), Ring Indicator (**RI**, ricevuto segnale acustico sul canale), Data Set Ready (**DSR**, DCE connesso e pronto a comunicare) e Clear To Send (**CTS**, DCE Pronto a ricevere)
- In ogni caso ciascuno di essi sarà *autorizzato ad interrompere* il processore **solo se** i previsti bit del **Registro** di Abilitazione delle Interruzioni [**port_9**]) sono stati predisposti a **1** logico (è necessario porre a **0** i bit degli *eventi* non autorizzati ad interrompere):

Priorità	bit3	bit2	bit1	bit0	Interrupt Enable Register [port_9]
2				1	1 = interruzione abilitata per Dato Ricevuto (Received Data Available) o per FIFO timeout [16550] [Interrompe se il Receive Buffer Register, RBR, è *pieno*]
3			1		1 = interruzione abilitata per Dato Trasmesso (Transmit Data Empty) [Interrompe se il Transmit Holding Register, THR, è *vuoto*]
1		1			1 = interruzione abilitata per Variazioni dello Stato della Linea (Receiver Line Status) [Interrompe se è stato rilevato un errore nel flusso dati o un Break]
4	1				1 = interruzione abilitata per Variazioni dello Stato del Modem (Modem Status) [Interrompe se stato rilevato un cambiamento nel handshake]

- Se l'evento che ha *richiesto interruzione* è abilitato il processore fa partire la *procedura di servizio* **INT 0CH** (se la richiesta è stata generata dall'**UART** della *porta seriale* **COM1** ed è arrivata da **IRQ4**) o **INT 0BH** (se la richiesta è stata generata dall'**UART** della *porta seriale* **COM2** ed è arrivata da **IRQ3**).
- Da notare che il **vettore** di queste procedure punta normalmente ad una locazione contenente il **JMP** ad una *procedura di default*: è probabile la necessità di scrivere una propria procedura personalizzata per il servizio delle interruzioni seriali.
- In ogni caso le **tecniche di Interrupt** sono descritte *molto dettagliatamente* nella parte che si occupa dei **sorgenti ASM** dedicati alla porta seriale.

- Poichè entrambe, **INT 0BH** e **INT 0CH**, devono comunque provvedere a soddisfare ognuna delle possibili 4 cause, in rigoroso ordine di priorità, è necessario sapere quali eventi *hanno avuto servizio* e quali hanno ancora *richiesta pendente*; per questo è possibile consultare i bit**2**/bit**1**/bit**0** nel **Registro** di Identificazione delle Interruzioni [**port_A**]:

bit2	bit1	bit0	Interrupt Identification Register [port_A]
		0	0 = interruzione pendente: un evento ha richiesto servizio d'interruzione 1 = nessuna interruzione pendente
x	x		Indica (Interrupt ID) quale degli eventi interrompenti è attualmente in attesa di servizio: 11 = Receiver Line Status *Interrupt*, priorità 1, massima, errore in linea o break 10 = Received Data Available *Interrupt*, priorità 2, dato ricevuto pronto 01 = Transmit Holding Register Empty *Interrupt*, priorità 3, dato trasmesso 00 = Modem Status *Interrupt*, priorità 4, minima, variazione segnali CD, RI, DSR o CTS

- Se il bit**0** è **0** significa che c'è un'*interruzione* è ***pendente***, cioè uno o più eventi hanno indotto l'**UART** ha generare una *richiesta* che non è stata ancora stata *servita* (o *servita del tutto*) dal processore.
- Il compito di stabilire quale dei possibili 4 eventi è ***in attesa*** di *essere servito* è affidato ai bit**2**/bit**1**: è prevista una codifica per ciascuno di essi che consente anche di dirimere l'*ordine* con cui dovranno essere serviti, nel caso di 2 o più *richieste contemporanee*: la coppia di bit presente di volta in volta nel registro è quella dell'evento a più alta priorità e, dopo il suo completo servizio, sarà sostituita (dall'**UART**) con quella dell'evento con la successiva più alta prioritaria. Ecco le codifiche, in dettaglio:
  - bit**2**/bit**1** = **11**: durante la ricezione di dati è stata rilevata la presenza di **errori** (*di sovrapposizione*, *di parità* o *di composizione*) o la presenza di un **segnale** *di break*; l'**UART** ha generato una *richiesta di* **Receiver Line Status** *Interrupt*, di priorità massima (prima), ed è in attesa di essere servito
  - bit**2**/bit**1** = **10**: il numero di bytes ricevuti ha superato quello massimo (*trigger level*) previsto per il **FIFO** in Ricezione oppure il **FIFO** contiene bytes in misura inferiore al massimo ma il tempo concesso a nuovi arrivi è terminato (*timeout*) oppure nel Receive Buffer Register è pronto un singolo dato; l'**UART** ha generato una *richiesta di* **Received Data Available** *Interrupt*, di seconda priorità, ed è in attesa di essere servito
  - bit**2**/bit**1** = **01**: il Transmit Holding Register è *vuoto* (con **UART 8250/16450** o con **UART 16550A**, se il **FIFO** in Trasmissione è disabilitato) oppure il **FIFO** ha posti liberi per uno o più bytes; l'**UART** ha generato una *richiesta di* **Transmit Holding Register Empty** *Interrupt*, di terza priorità, ed è in attesa di essere servito
  - bit**2**/bit**1** = **00**: durante la comunicazione con il *Modem* è stata rilevata la variazione dei segnali in arrivo, direzione **DCE>DTE**, come Data Carrier Detect (**CD**, rilevato modem remoto/possibile comunicare), Ring Indicator (**RI**, ricevuto segnale acustico sul canale), Data Set Ready (**DSR**, DCE connesso e pronto a comunicare) e Clear To Send (**CTS**, DCE Pronto a ricevere); l'**UART** ha generato una *richiesta di* **Modem Status** *Interrupt*, di priorità più bassa (quarta), ed è in attesa di essere servito
- Sebbene la cosa sia ora irrilevante ricordo che gli eventi a **priorità 1**, **2** e **3** sono segnalati *anche* da software nel **Registro** di Stato della Linea [**port_D**], bit**0** (*Ricezione dato*), bit**1** (*errore di sovrapposizione*), bit**2** (*errore di parità*), bit**3** (*errore di composizione*), bit**4** (*segnale di Break*) e bit**5** (*dato ricevuto*)] mentre quelli a **priorità 4** sono segnalati anche da software nel **Registro** di Stato del Modem [**port_E**], bit**0** (*variazione* $\overline{CTS}$), bit**1** (*variazione* $\overline{DSR}$), bit**2** (*variazione* $\overline{RI}$) e bit**3** (*variazione* $\overline{CD}$)]

Applicazioni Hardware — Porta Seriale

- Applicazioni Hardtware - Presentazione

1.1 Presentazione dei Progetti Hardware per la Porta Seriale

Applicazioni Software — Porta Seriale

- Applicazioni Software - Presentazione

1.1 Presentazione delle Applicazioni Software per la Porta Seriale