Lucrare Lab 9

PORTUL SERIAL (COM)

Scopul lucrării: Studiul principiilor comunicației seriale și a structurii porturilor seriale ale sistemelor IBM PC.

1. Considerații generale

Transfer paralel versus ***transfer serial*** al unui **octet (8 biți)** de date, de ex. **'A'=41h=0100 0001b**:

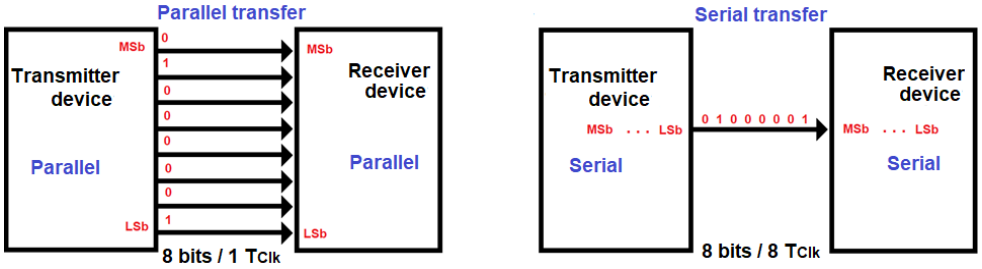


Fig.1. Transfer paralel versus ***transfer serial*** pentru 8 biți de date, 41h='**A**'

Pentru a asigura transferul serial al datelor, există circuite de interfață (cu perifericele) programabile pt comunicații ***asincrone***/ ***sincrone***:
U**A**RT (Universal Asynchronous Receiver – Transmitter) (exemplu: **UART 8250**)
U**SA**RT (Universal Synchronous- Asynchronous Receiver – Transmitter) (exemplu: USART 8251).

UART - Universal Asynchronous Receiver/Transmitter

Porturile seriale standard de pe PC-urile IBM au fost implementate cu **UART 8250** sau cu **noile succesoare compatibile 16N50** (unde N este de 4,5,6,7,8 sau 9). Aceste controlere specializate îndeplinesc mai multe sarcini, cum ar fi:
- ***serializează datele care trebuie transmise***, adică realizează conversia din format paralel în format serial pentru caracterele de date primite de la CPU (fiecare octet scris în UART este trimis bit-cu-bit pe linia serială, cu LSb mai întâi);
- ***deserializează datele recepționate***, adică realizează conversia din serie în paralel a caracterelor de date primite de la un dispozitiv periferic sau modem (după primirea tuturor biților unui caracter, aceștia sunt asamblați într-un singur octet, oferit apoi CPU pt citire);
- adaugă sau elimină informațiile de încadrare (bit de start, bit/biți de stop, (opțional) bit de paritate). Biții de date sunt încadrați înainte de transmiterea lor pe linia serială și sunt extrași din cadru după recepția lor de pe linia serială;
- implementează sistemul de întrerupere al procesorului și dialogul de control al modemului conform standardului RS232;
(Interfața externă fol. semnale bipolare și polaritate inversă atât pt semnalele de date seriale, cât și pt semnalele de control modem);
- stochează temporar atât biții de date primiți, cât și cei pe care trebuie să îi transmită (folosind buffere).

Standardul de interfață serială specifică atât semnalele pt transmisia și recepția datelor, cât și semnalele de control al fluxului dintre ***DTE*** - *Data Terminal Equipment* și ***DCE*** - *Data Communication Equipment,* scopul inițial al interfeței fiind interconectarea unui computer (DTE) și a unui modem (DCE). Modemul (MODulator-DEModulator) permite unui PC să trimită date pe o linie telefonică analogică, realizând conexiuni la distanțe mari prin PSTN (Public Switched Telephone Network). Numerele de telefon au jucat rolul adreselor IP de astăzi.

Specificațiile electrice și mecanice ale portului serial sunt definite de EIA (Electronic Industry Association) în **standardul RS232**, care utilizează ***o logică negativă (sau inversată)*** la trecerea informațiilor prin cablul interfeței seriale:
- un bit de date 0 („Space" sau starea activă a unui semnal de interfață) este codat ca un nivel de tensiune pozitiv, între +3 și +25 V;
- un bit de date 1, („Mark" sau semnalul de control al modemului inactiv) este codat ca un nivel de tensiune negativ, între -3 și -25 V;
- regiunea cuprinsă între +3 V și -3 V este exclusă;
- lungimea maximă a cablului este de 15m, iar curentul de scurtcircuit trebuie să fie limitat la 500 mA.

Pe linia de transmisie, un caracter RS-232 constă dintr-**un bit de start** („Mark", cu valoare logică 0), urmat de ***5 până la 8 biți de date*** (trimiși cu LSb mai întâi pe linie), **un bit de paritate opțional** (paritatea se aplică numai biților caracterului) și **1 sau 2 biți de stop** („Space", cu valoare logică 1), așa cum arată Fig. 2 (numărul de biți transmiși sau recepționați pentru fiecare octet este de 10 biți aici). Acronimul de denumire pentru această opțiune de încadrare este **8E2**. Fiecare unitate de date, de la bitul de start până la bitul/ biții de stop, definește ***un „cadru" de biți***.

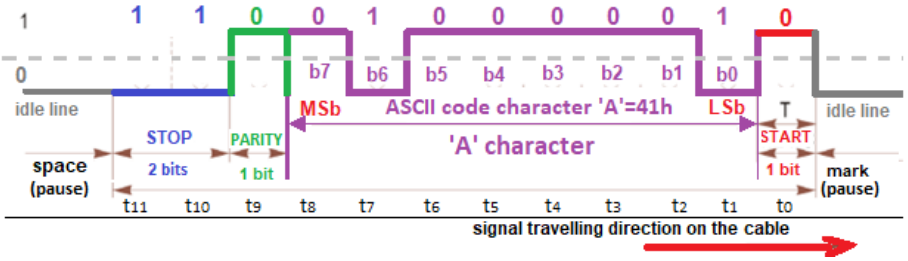


Fig 2. Semnalul de date la transmisia unui caracter 'A', cu bit de start, 2 biți de stop și paritate pară prin cablul serial

Componenta principală a unui port serial este circuitul UART care realizează:
- conversia paralel-serie a datelor de transmis; de exemplu, când CPU execută o instrucțiune de forma ***out AdrPortRegTx$_{UART}$, AL***, octetul din registrul AL al CPU se trimite în mod paralel pe BD al sistemului și se înscrie într-un registru intern al UARTului, care îl serializează în ritmul tactului de emisie;
- conversia serie-paralel a datelor recepționate, adică în formatul paralel utilizat intern de CPU; de exemplu, când CPU execută o instrucțiune de forma ***in AL, AdrPortRegRx$_{UART}$***, octetul din registrul intern al UARTului va ajunge în format paralel, prin BD, în reg. AL.
La transmisie, UARTul adaugă în fața biților de date un bit de start, iar după ele bitul de paritate (opțional) și bitul/ biții de stop.
La recepție, UARTul extrage datele utile din cadrul sosit, verificând că bitul de paritate recalculat la recepție este egal cu cel transmis.
Pentru comunicația serială, este necesar ca cele două terminale (transmițător și receptor) să fie configurate cu aceleași valori ale parametrilor de transmisie/ recepție înainte de transmisia/ recepția efectivă a caracterelor. Astfel, trebuie configurate:

1. Formatul caracterului: numărul de biți de date, numărul de biți de stop, dacă se folosește sau nu paritate (și tipul parității);
2. Viteza de transmisie/recepție (biți pe secundă) a datelor: cu ce viteză să circule biții pe linia serială.

1. Formatul caracterului:

Codificarea caracterului la transmisia serială se poate realiza ***prin 5, 6, 7 sau 8 biţi*** (încadraţi de 1 bit de START şi 1...2 biţi de STOP). De exemplu, standardul de codificare ASCII foloseşte 7 biţi, standardul ASCII Extins foloseşte 8 biţi, iar pentru date provenite de la tastatură se pot folosi 5 sau 6 biţi per caracter (pt codurile scan).
În sistemele mai vechi se foloseau ***2 biţi de STOP*** pentru a acorda dispozitivului timp suficient pentru a se pregăti de recepţia următorului caracter, dar în sistemele mai noi se folosește ***1 bit de STOP***.
În realitate, pe lângă biţii de START şi STOP, sistemele actuale mai folosesc **bit de PARITATE** pentru a verifica integritatea datelor într-un cadru. În general, paritatea utilizată pe linia serială este pară (EVEN) sau impară (ODD).

- La paritatea **pară**, nr de biţi de 1 ai caracterului împreună cu bitul de paritate, trebuie să fie un număr ***PAR***;
- La paritatea **impară**, nr de biţi de 1 ai caracterului împreună cu bitul de paritate, trebuie să fie un număr ***IMPAR***;
- Există și posibilitatea ca bitul de paritate să fie blocat în ***0*** sau ***1***, indiferent de paritatea efectivă a caracterului.

În cazul comunicației asincrone, sincronizarea la nivel de bit este asigurată numai pe durata transmisiei /recepției efective a fiecărui caracter. O asemenea comunicație este orientată pe caractere individuale și are dezavantajul că necesită informații suplimentare în proporție de cel puțin 25% pentru identificarea fiecărui caracter.

2. Viteza de transmisie/recepție a datelor (Rata de transfer a datelor):

Sincronizarea la nivel de bit între transmițător (Tx) și receptor (RX) se realizează cu ajutorul semnalelor de ceas locale având aceeași frecvență la Tx și Rx. Atunci când receptorul detectează începutul unui caracter indicat prin bitul de START, ***pornește un oscilator de ceas local***, care permite eșantionarea corectă a biților individuali ai caracterului. Eșantionarea biților se realizează aproximativ la mijlocul intervalului corespunzător fiecărui bit.
Rata de transfer a datelor în comunicaţia serială se măsoară în ***bps*** (biţi pe secundă) sau „***baud rate***" (rată de baud)[1].
Componentele principale ale UART sunt: 1) Generator de rată de baud, 2) Transmiţător, 3) Receptor

2. Circuite UART

La calculatoarele IBM PC și IBM PC/XT originale au fost utilizate circuite UART din familia 8250. Începând cu primele sisteme de 16 biți, s-a introdus circuitul UART 16**4**50. Acest circuit este compatibil înapoi cu 8250 la nivelul regiștrilor, dar permite comunicația cu viteze mai ridicate. La calculatoarele IBM PS/2 a fost utilizat circuitul UART 16**5**50, care a fost adoptat și pentru sistemele bazate pe procesorul 80386 și următoarele. La 16550 s-a adăugat câte o memorie FIFO de 16 octeți pentru transmisie și recepție, permițând viteze de comunicație superioare față de vitezele permise de circuitele anterioare. Începând cu sistemele bazate pe procesorul Pentium, circuitul UART 16550 (sau o versiune mai recentă a acestuia) a fost inclus în setul de circuite de pe placa de bază. Versiuni îmbunătățite ale circuitului UART 16550 sunt circuitele 16**6**50, 16**7**50, 16**8**50 și 16**9**50.

- Circuitul **8250 UART** a fost cel original, utilizat în sistemele IBM PC și PC-XT și era capabil de viteze până la 9600 bps cu un FIFO de 1 octet;
- Circuitul **8250A UART** a fost o versiune revizuită a modelului 8250 cu un registru suplimentar care a permis verificarea software-ului;
- Circuitul **16450 UART** a fost puțin mai rapid decât UART-urile anterioare, utilizat în primele versiuni ale sistemelor PC-AT;
- Circuitul **16550 UART** are două memorii FIFO de 16 biți (dar cu un bug FIFO), cu o viteză maximă de 115,2 Kbps;
- Circuitul **16550A UART** a avut aceleași caracteristici ca și 16550 UART anterior, dar cu bug-ul remediat;
- Circuitul **16650 UART** are două memorii FIFO de 32 de biți cu rate Tx / Rx până la 230,4 Kbps;
- Circuitul **16750 UART** are două memorii FIFO de 64 de biți cu rate Tx / Rx până la 460,8 Kbps;
- Circuitul **16950 UART** are două memorii FIFO de 128 de biți cu rate Tx / Rx până la 921,6 Kbps.
- Circuitul **Hayes ESP (**Porturi seriale îmbunătățite**)** - card de 8 biți cu 2 UART-uri 16550A și un procesor care permite transferul DMA.

[1] Rata de baud sau nr. de simboluri pe secundă este o terminologie a modemului şi se defineşte ca nr. de schimbări ale parametrilor semnalului pe secundă (de ex. frecvența, faza, amplitudinea), dar la sistemele de transmisie în banda de bază unde avem 1 singur bit/simbol, cele 2 terminologii se consideră interschimbabile.

3. Semnalele UART 8250/16550

3.1. Diagrama bloc

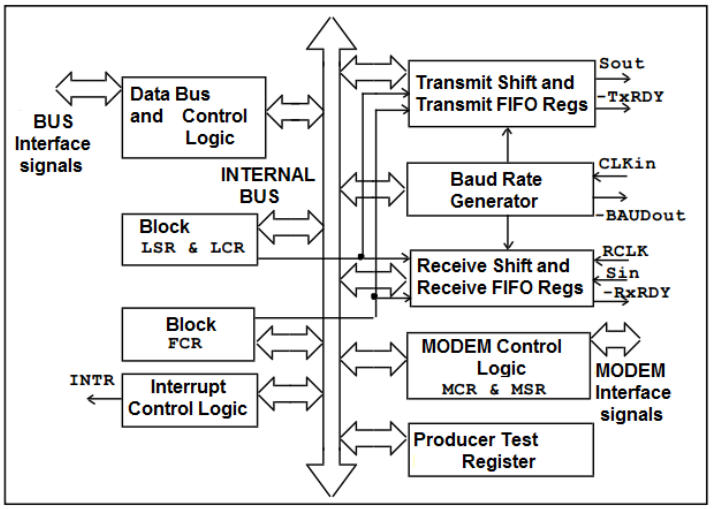


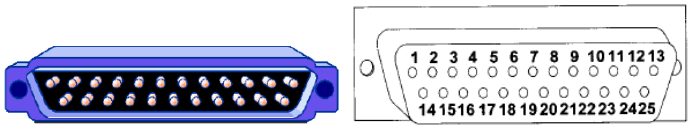
Fig. 3. Schema bloc internă a circuitului UART 16550

Tabel 1. Asignarea pinilor la circuitul UART 16550A

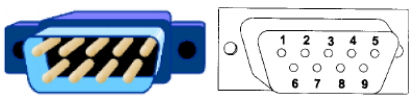
Nr.Pin	Nume	Descriere
1-8	D0:D7	Data Bus (bidirectional)
9	RCLK	Receiver Clock Input. The frequency should be the receiver's baud rate*16
10	**RxD**	**Receive Data**
11	**TxD**	**Transmit Data**
12	CS0	Chip Select 0 – Active High
13	CS1	Chip Select 1 – Active High
14	-CS2	Chip Select 2 – Active Low
15	-BAUDOUT	Baud Output – Output from Programmable BaudRate Generator, Frequency=BaudRate x16
16	XIN	External Crystal Input – used for Baud Rate Generator Oscillator
17	XOUT	External Crystal Output
18	-WR	Write Line - Inverted
19	WR	Write Line – Not Inverted
20	VSS	Connected to Common Ground GND
21	RD	Read Line - Not Inverted
22	-RD	Read Line – Inverted
23	DDIS	Driver Disable. This pin goes low when CPU is reading from UART. It can be connected to Bus Transceiver in case of high capacity data bus.
24	-TXRDY	Transmit Ready
25	-ADS	Address Strobe – used if signals are not stable during read or write cycle
26-28	A2-A0	Address Bits for registers selection, allowing 8 port addresses
29	-RXRDY	Receive Ready (not used for 8250)
30	INTR	Interrupt Output
31	-OUT2	User Output 2
32	**-RTS**	**Request To Send**
33	**-DTR**	**Data Terminal Ready**
34	-OUT1	User Output 1
35	MR	Master Reset
36	**-CTS**	**Clear To Send**
37	**-DSR**	**Data Set Ready**
38	**-DCD**	**Data Carrier Detect**
39	**-RI**	**Ring Indicator**
40	VDD	+5 Volts

3.2. Conectori folosiți pentru porturile seriale (COM) în PC

Standardul original în IBM-PC a definit conectorul DB-25 P (tată) la DTE și conectorul de tip S (mamă) la DCE, cu 25 de pini, aceștia fiind înlocuiți ulterior de conectorii mai ieftini DB-9.



Conector **DB 25 P**



Conector **DB 9 P**– folosit pt porturile seriale in sistemele IBM-PC

Fig. 4. Conectori folosiți pentru porturile seriale COM în PC

Porturile seriale pot utiliza unul din două tipuri de conectori: **conectorul serial tată DB-25** a fost utilizat la calculatoarele din generațiile anterioare, gen PC, PC-XT, PS2, care era d.p.d.v. mecanic identic cu cel folosit pentru portul paralel (dar acela avea conector mamă). La calculatoarele ulterioare, gen PC-AT, acesta a fost înlocuit de **conectorul tată DB-9** care are doar 9 pini. Pentru echipamentele periferice sau cabluri se utilizează conectori mamă (de exemplu, putem conecta 2 PC-uri între ele pentru a transfera date, cu un **cablu null modem,** având conectori mamă DB-9 la ambele capete).

3.3. Semnalele interfeței seriale

Din cele 25 de semnale ale conectorului DB-25, se utilizează cel mult 9 semnale pentru o conexiune serială obișnuită. Tabelul 2 arată numele acestor semnale și pinii asociați la conectorii DB-25 și DB-9.

Tabel 2. Funcțiile semnalelor interfeței seriale

Pin semnal (DB25/DB9)	Nume	Funcție	Tip semnal
TxD (2/3)	Transmit Data	Data Output (TXD)	Out
RxD (3/2)	Receive Data	Data Input (RXD)	In
DCD (8/1)	Data Carrier Detect	Carrier Detect in modem line	In
DTR (20/4)	Data Terminal Ready	UART is ready for connection	Out
DSR (6/6)	Data Set Ready	Modem is ready for communication	In
RTS (4/7)	Request To Send	DTE is ready to change data	Out
CTS (5/8)	Clear To Send	Modem (DCE) is ready to change data	In
RI (22/9)	Ring Indicator	RI=1 when DCE detect a ring signal from PSTN	In
SG(7/5)	Signal Ground	GND	

Semnalele implicate în comunicația serială, așa cum apar la conectorul DB-9 sunt:

Pinul 1 – CD sau DCD Carrier Detect (Detecție purtătoare): Prin activarea acestui semnal, modemul semnalează calculatorului faptul că a detectat semnalul purtător al altui modem pe linia telefonică, deci există o conexiune cu un modem aflat la distanță. Într-o legătură serială fără modemuri, semnalul CD nu se folosește.

Pinul 2 – RxD Receive Data (Recepție date): Această linie este utilizată de calculator pentru recepția datelor de la modem sau de la un echipament extern care folosește protocolul UART (în gen. microcontrolerele implementează interfața UART printre perifericele lor on-chip).

Pinul 3 – TxD - Transmit Data (Transmisie Date): Datele sunt transmise serial pe această linie de către calculator. În general, pentru transmisia datelor este necesar ca semnalele RTS, CTS, DTR și DSR să fie active, acestea fiind activate fie în cadrul unei secvențe de stabilire a legăturii cu modemul ("dialogul modem-calculator"), fie printr-o cuplă de rebuclare.

Pinul 4 – DTR Data Terminal Ready (Terminal de date gata): Atunci când calculatorul este operațional și pregătit pentru comunicația de date, activează semnalul DTR. Modemul va răspunde la semnalul DTR prin activarea semnalului DSR.

Pinul 6 – DSR Data Set Ready (Modem operațional): Atunci când modemul sau echipamentul extern este operațional și pregătit pentru comunicația de date, activează semnalul DSR. Acest semnal este activat de modem ca răspuns la activarea semnalului DTR de către calculator. Calculatorul va transmite date către modem doar în cazul în care semnalul DSR este activ.

Pinul 7 – RTS Request To Send (Cerere de emisie): Atunci când calculatorul este pregătit pentru transmisia datelor, activează semnalul RTS. Acest semnal indică modemului faptul că celălalt sistem (calculatorul) vrea să înceapă transmisia. Un semnal RTS inactiv va preveni modemul de a transmite date către calculator. Aceasta permite calculatorului să controleze fluxul datelor transmise de către modem. Răspunsul la semnalul RTS se recepționează de calculator pe linia CTS. În cazul transmisiei asincrone, semnalele RTS și CTS se pot ignora.

Pinul 8 – CTS Clear To Send (Gata de emisie): Prin activarea acestui semnal, modemul sau echipamentul extern indică faptul că este pregătit pentru recepția datelor de la calculator. Semnalul CTS este activat de modem ca răspuns la activarea semnalului RTS de către calculator. Un semnal CTS inactiv va preveni calculatorul de a transmite date către modem. Aceasta permite modemului să controleze fluxul datelor transmise de calculator.

Pinul 9 – RI Ring Indicator (Indicator de apel): Atunci când modemul detectează pe linia telefonică semnalul de apel de la centrala telefonică, el activează semnalul RI, acesta permițând programului de comunicație să intre în procedura de răspuns automat.

Pentru a fi posibilă comunicația între dispozitive cu viteze diferite, proiectanții interfeței seriale au prevăzut semnale speciale pentru controlul fluxului de date. Aceste semnale permit unui echipament oprirea și apoi reluarea transmiterii datelor la cererea echipamentului de la celălalt capăt al liniei de comunicație serial ("backpressure").

Stabilirea legăturii între calculator și modem presupune utilizarea unui **protocol de comunicație** cu ajutorul semnalelor de control ale interfeței seriale.

Etapele acestui ***protocol (de tip HANDSHAKING) sunt*** (Fig.5):

1) **programul activează semnalul DTR și așteaptă răspunsul de la modem,**
reprezentat de activarea semnalului DSR (pentru cuplarea la linia telefonică).

2) **programul activează semnalul RTS și așteaptă răspunsul de la modem,**
reprezentat de activarea semnalului CTS (pentru începerea transmisiei efective).

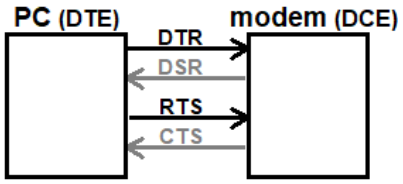


Fig. 5. Semnalele necesare stabilirii legăturii seriale între un calculator și un modem

- Calculatorul trebuie să detecteze activarea semnalului DSR (și CTS în cazul transmisiei sincrone) înainte de a transmite date către modem. Dezactivarea semnalului DSR va opri, de obicei, fluxul de date de la calculator.
- Modemul verifică activarea semnalului DTR (și RTS în cazul transmisiei sincrone) și se cuplează la linie (confirmand aceasta prin activarea semnalului DSR) înainte de a transmite date pe linia serială sau către calculator. Dezactivarea semnalului DTR va cauza decuplarea modemului de la releul de linie.
- La anumite sisteme, semnalul CD trebuie să fie active ca modemul să înceapă recepția datelor.

3.4. Cabluri Există mai multe variante de cabluri care se pot utiliza pentru comunicația serială. Semnalele interfeței seriale au fost prevăzute în scopul conectării unui echipament DCE la un echipament DTE.

1) Atunci când se conectează două asemenea echipamente, de exemplu, un calculator cu un modem, care dispun de conectori de același tip (de exemplu, DB-25), este necesar un cablu care conectează pinii cu același număr ai conectorilor de la cele două capete. Acesta este un **cablu direct** sau cablu modem. Într-un cablu direct, fiecare pin corespunzător unui anumit semnal de la un capăt este conectat cu pinul corespunzător aceluiași semnal de la celălalt capăt (pin 2 cu pin 2, etc)-Fig.6.
2) Dacă se conectează două echipamente cu conectori diferiți (unul cu DB-25, altul cu DB-9), este necesar un **cablu adaptor** și se va ține cont de relația (numerotarea diferită) între pinii celor 2 conectori (Tabelul 2). Deși la calculatoarele IBM PC se utilizau numai conectori DB-9, la unele periferice ca imprimante seriale, plottere sau modemuri, se utilizau conectori DB-25.
3) Dacă se conectează două echipamente DTE între ele, de exemplu, două calculatoare, conexiunile sunt realizate astfel încât, din punctul de vedere al calculatorului, comunicația să se desfășoare ca și cum la celălalt capăt s-ar afla un modem și nu un alt calculator. Datele transmise de primul calculator trebuie recepționate de al doilea calculator, astfel încât pinul TxD de la primul calculator este conectat cu pinul RxD de la cel de-al doilea calculator și invers (pinul 2 se va conecta cu pinul 3, pinul 3 cu pinul 2). Cei doi pini pentru masa electrică trebuie conectați împreună. Deoarece conexiunile acestor pini trebuie inversate la cele două capete ale cablului, un asemenea cablu este numit **cablu inversor** (numit și cross cable) – Fig. 7.

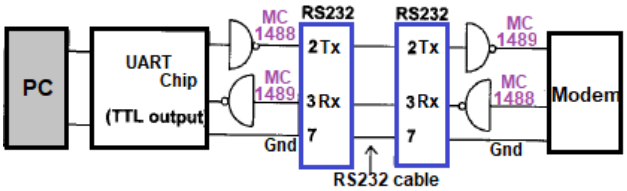


Fig.6. Legătura dintre 2 echipamente DTE şi DCE prin cablu RS232 („cablu modem") sau cablu direct

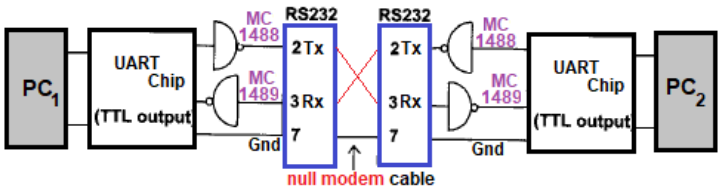


Fig.7. Legătura dintre 2 dispozitive DTE prin „cablu NULL modem", cablu „cross" sau cablu inversor

Astfel, avem o conexiune (în ambele cazuri) de minim 3 semnale/ 3 pini: Tx, Rx şi Gnd: o legătură de bază RS-232 necesită doar trei conexiuni: una pentru transmisie, una pentru recepție și una pentru masa electrică comună.

În general, există 2 tipuri de cabluri inversoare:

- Primul tip de cablu inversor se poate utiliza atunci când controlul fluxului de date se realizează printr-o ***metodă software***: nu se utilizează toate semnalele pentru controlul fluxului de date, astfel *conexiunile putând fi simplificate.* Sunt necesare *doar **3 conexiuni** între cele 2 echipamente DTE:* pentru datele transmise, datele recepționate și masa electrică (pinii 2/3, 3/2 și 7/5). La ambii conectori, pinul DTR este conectat cu pinii DSR și CD de la același capăt al conexiunii. Astfel, atunci când semnalul DTR este activat, semnalele DSR și CD vor fi activate și ele (semnalele DSR și CD se activează deci în același mod ca și cum la celălalt capăt al conexiunii s-ar afla un modem). În mod similar, la ambii conectori, pinul RTS este conectat cu pinul CTS de la același capăt al conexiunii (**cablul null-modem**) – Fig.8.

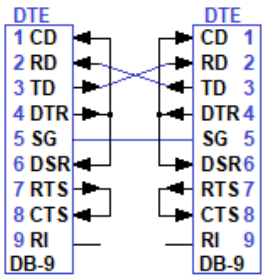


Fig. 8. Conectarea a două echipamente DTE printr-un **cablu null-modem** (DB-9)

- Cel de-al doilea tip de cablu inversor se poate utiliza atunci când controlul fluxului de date se realizează printr-o ***metodă hardware***: cele trei linii ale cablului null-modem pentru conectarea a 2 echipamente DTE între ele prezentat anterior nu mai sunt suficiente (se va utiliza un **cablu inversor cu conexiuni suplimentare** pentru această metodă de control). Pinii de date și pinul pentru masa electrică sunt conectați în același fel ca și la cablul null-modem, dar la ambii conectori, pinul DTR este conectat cu pinii DSR și CD de la celălalt capăt al conexiunii. Prin această conexiune, fiecare din cele două echipamente terminale de date poate determina momentul în care celălalt echipament terminal de date este pregătit. În plus, semnalele utilizate pentru controlul fluxului de date (RTS și CTS) sunt conectate între cei 2 conectori între ele: pinul RTS este conectat cu pinul CTS de la celălalt capăt al conexiunii – Fig.9.

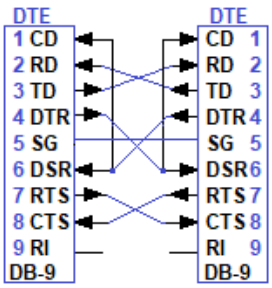


Fig. 9. Conectarea a două echipamente DTE printr-un **cablu inversor** cu controlul fluxului de date prin hardware (DB-9)

Trecerea de la nivel TTL la RS232 se face fol. circuite de linie sau convertoare de nivel: 1489 la Rx și 1488 la Tx.

4. Programarea UART. Regiștrii 8250/16550

4.1. Adresele de bază și liniile de întrerupere atribuite porturilor seriale COM din PC

Regiștrii cipurilor UART sunt accesibili prin instrucțiuni I/ O (de intrare/ieșire) la 8 adrese de port diferite (interfața are, de asemenea, unii regiștri care nu sunt accesibili prin software), pornind de la AB (adresa de bază) a portului COM respectiv.

ROM BIOS-ul computerelor IBM PC originale a permis utilizarea a două porturi seriale, denumite COM1 și COM2. Pentru primul adaptor serial (sau port) COM1, regiștrii portului pot fi accesați la adrese cuprinse între 3F8h și 3FFh (deci pornind de la AB 3F8h), iar pentru al doilea, COM2, regiștrii portului pot fi accesați la adrese între 2F8h și 2FFh (deci pornind de la AB 2F8h). Ulterior, numărul de porturi a fost extins cu încă alte două porturi, denumite COM3 și COM4 (adresele porturilor COM3 și COM4 pot să difere, nu au fost standardizate, adesea fiind 3E8h și 2E8h). Când există mai mult de două porturi seriale în computer, acestea vor partaja unele nivele de întrerupere. În sistemele PC-AT (Fig. 10), portul COM3 partajează întreruperea de nivel 4 (IRQ4) cu portul COM1, iar portul COM4 partajează întreruperea de nivel 3 (IRQ3) cu portul COM2. De la S.O. Windows 95, numărul de porturi seriale a fost extins în continuare (la 128). Magistrala PCI permite partajarea nivelelor de întrerupere, astfel încât pentru plăci de expansiune bazate pe magistrala PCI (sau PCI Express) este posibilă utilizarea unui singur nivel de întrerupere fără a genera conflict.

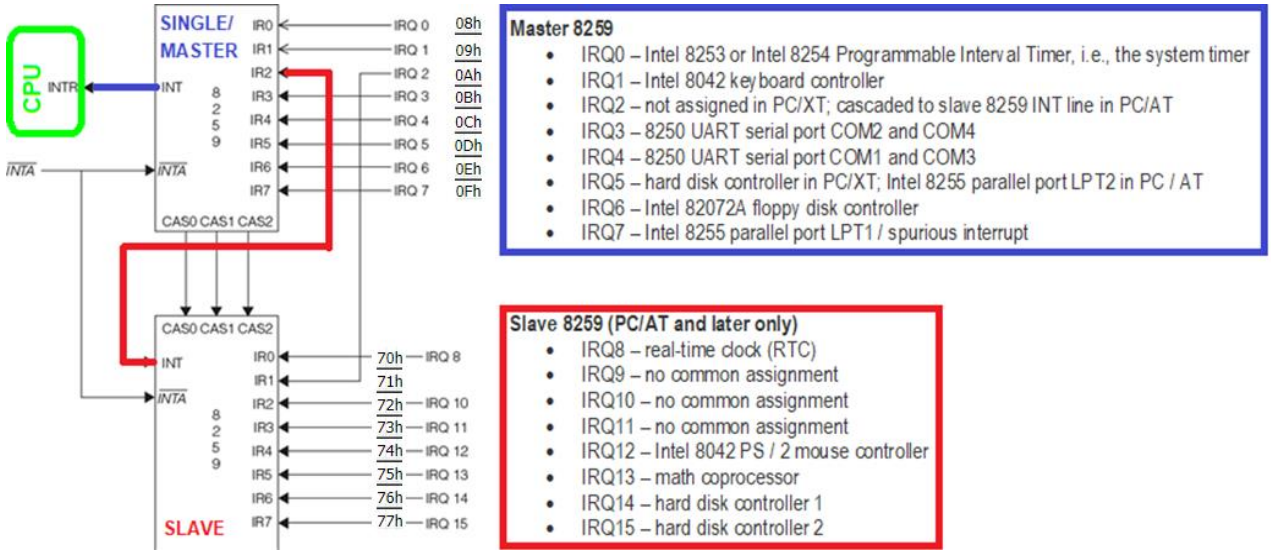


Fig. 10. Conectarea dispozitivelor seriale de tip COM într-un PC-AT pe liniile IRQ3 (COM2 și COM4) și IRQ4 (COM1 și COM3)

4.2. Regiștrii cipului UART

Primele 2 adrese ale unui port COM permit accesul la mai mult de un registru al cipului. Tabelul 3 prezintă adresele regiștrilor accesibili software pentru primele două porturi seriale COM, bazat pe o valoare de offset care indică deplasamentul adresei fiecărui registru, în raport cu adresa de bază a portului serial (AB). Modurile de acces pentru fiecare registru al cipului UART sunt, de asemenea, prezentate în Tabelul 3. Regiștrii circuitelor UART (cei accesibili software) se pot accesa prin instrucțiuni de citire şi scriere pe port.

Tabel 3. Regiștrii circuitelor 8250 / 16550 pentru COM1 și COM2 (cu 3F8h, resp. 2F8h adresa de bază)

Adresa de bază (AB)	Offset	DLAB (b_7 din LCR)	Operație suportată	Registru
3F8h (2F8h)	+0	0	Scriere	**THR** (Transmit Holding Register)
		0	Citire	**RBR** (Receive Buffer Register)
		1	Citire/Scriere	**DL** (Divisor Latch Register)-**LSB**
3F9h (2F9h)	+1	0	Citire/Scriere	**IER** (Interrupt Enable Register)
		1	Citire/Scriere	**DL** (Divisor Latch Register)-**MSB**
3FAh (2FAh)	+2	x	Citire	**IIR** (Interrupt Identification Register)
		x	Scriere	**FCR** (FIFO Control Register) - numai la 16550
3FBh (2FBh)	+3	x	Citire/Scriere	**LCR** (Line Control Register)
3FCh (2FCh)	+4	x	Citire/Scriere	**MCR** (Modem Control Register)
3FDh (2FDh)	+5	x	Citire	**LSR** (Line Status Register)
3FEh (2FEh)	+6	x	Citire	**MSR** (Modem Status Register)
3FFh (2FFh)	+7	x	Citire/Scriere	**SPR** (Scratchpad Register)- numai la 16550

Observație: Circuitele 16650, 16750 și 16850 conțin regiştri suplimentari față de cei indicaţi în **Tab.3**, iar structura acestora poate varia de la un producător la altul.

Primele două adrese ale portului (AB și AB+1) permit accesul la mai mult de un registru al interfeţei, discriminarea realizându-se pe baza operaţiei realizate pe port (citire sau scriere) și/ sau prin intermediul bitului 7 al registrului LCR (Line Control register) aflat la adresa AB+3, bit denumit DLAB (***Divisor Latch*** Access Bit). Atunci când bitul DLAB este setat la 1, se permite accesul la doi regiştri (DL_{LOW} la adresa AB și DL_{HIGH} la adresa AB+1) utilizaţi pentru setarea vitezei de comunicație, iar când acest bit este resetat (în 0), se asigură accesul la regiștrii de date THR și RBR (în funcție de operația pe port) la adresa AB, resp. se asigură accesul la registrul IER la adresa AB+1. Tabelul 4 sintetizează selecția regiștrilor de la adresele AB și AB+1.

Tabel 4. Discriminarea între regiștrii circuitelor 8250 / 16550 pe baza bitului $b7_{LCR}$ și a adresei de bază a portului

Instrucțiune	Conținutul registrului DX	Registrul ce se va accesa
in AL, DX	**AB**	**RBR (dacă $b7_{LCR}$=0) sau DL_{LOW} (dacă $b7_{LCR}$=1)**
out DX, AL	**AB**	**THR (dacă $b7_{LCR}$=0) sau DL_{LOW} (dacă $b7_{LCR}$=1)**
in AL, DX	**AB +1**	**IER (dacă $b7_{LCR}$=0) sau DL_{HIGH} (dacă $b7_{LCR}$=1)**
out DX, AL	**AB +1**	**IER (dacă $b7_{LCR}$=0) sau DL_{HIGH} (dacă $b7_{LCR}$=1)**

DLR (Divisor Latch Register) – Citire sau Scriere la AB (DL_{Low}), resp. la AB+1 (DL_{High})

Cei 2 regiștri **DL_{Low}** și **DL_{High}** (care pot fi accesați folosind adresele de port 3F8h (2F8h) și respectiv 3F9h (2F9h)) conțin valoarea cu care frecvența de ceas internă (1.8432 MHz) a circuitului 8250/16550 trebuie divizată pentru a obține rata binară dorită. Registrul **DL_{Low}** conține cel mai puțin semnificativ octet al divizorului, iar registrul **DL_{High}** conține cel mai semnificativ octet al divizorului. Cei doi regiștri divizor sunt accesibili dacă ***b7 este 1 în registrul LCR.***

Pentru calcularea valorii divizorului, trebuie luat în considerare factorul de ceas al cipului 8250/16550. De obicei, acest factor este 16 (rata binară este de 16 ori mai mică decât frecvența obținută prin divizare sau, cu alte cuvinte, sunt necesare 16 T_{OscClk} pentru a eșantiona bitul la mijlocul nivelului său). Tabelul 5 conține divizorii corespunzători diferitelor rate binare.

Se poate folosi următoarea formulă: **Divizor = 1,843,200 / (RataBinară * 16)**

Tabel 5. Diferiți divizori pentru obținerea unor rate binare variate

Rata Binară (bps sau biți/s)	Divizor		Rata Binară (bps sau biți/s)	Divizor
150	300h		4,800	18h
300	180h		7,200	10h
600	C0h		9,600	0Ch
1,200	60h		19,200	06h
1,800	40h		38,400	03h
2,400	30h		115,200	01h
3,600	20h			

La frontul căzător al bitul de START, UARTul resetează un numărător intern tactat cu $16*f_{bit}$, care ajungând la conținut=8 indică mijlocul bitului de start (dacă s-a ales aceeași f_{bit} la transmisie și la recepție). Dacă linia nu e gasită în 0 logic, UARTul consideră eveniment "False Start bit", îl ignoră, pt. că a fost doar un zgomot aditiv, un impuls scurt pe linie și începe să "vâneze" următorul front căzător pe pinul RxD. După găsirea START bit, alte 16 perioade de ceas sondează linia la mijlocul primului bit de date, LSB-ul. Astfel continuă citirea restului de biți din cadru, încheind cu bitul de stop. Mecanismul asigură citirea fiecărui bit în momentul optim, adică la mijlocul duratei de bit, chiar dacă frecvența de cuarț de la recepție diferă ușor de cea de la emisie. Precizia cuarțurilor se garantează cu 5 cifre, de ex. 1.8432MHz. Pentru că receptorul are nevoie de un tact de $16f_{bit}$, viteza maximă de transmisie este 1.8432MHz/16 adică 115200bps; MODEMurile în bandă vocală de 300-3400Hz (pe linie telefonică) nu pot realiza debite binare mai mari de 56kbps.

Registrul de date (Reg. De Transmisie / Reg. de recepție): Registrul de date constituie de fapt 2 regiștri separați: registrul de transmisie și registrul de recepție. ***Registrul de transmisie*** este selectat prin scrierea datelor la adresa I/O, în timp ce ***registrul de recepție*** este selectat prin citirea datelor de la adresa I/O (ambele la AB+0). Presupunând că registrul de transmisie este gol, scrierea în registrul de transmisie începe o transmisie de date pe linia serială. Presupunând că bufferul de recepție este plin, citirea registrului de recepție returnează datele care au ajuns pe linia serială.

Registrul THR (Transmiter Holding Register) – Scriere la AB+0

Este bufferul de transmisie și este selectat ***dacă b7 este 0 în registrul LCR***. Caracterul care trebuie trimis trebuie înscris în acest registru. Dacă registrul THR este gol (deci poate fi înscris cu un caracter nou), cipul generează o cerere de întrerupere, dacă generarea acestui tip de întrerupere este activată. Starea acestui registru poate fi determinată prin ***testarea bitului 5 al reg. LSR***.

Registrul TSR (Transmiter Shift Register) – Este un registru intern, neaccesibil prin software.

Când transmisia unui caracter este finalizată, deci se poate trimite un carcater nou, conținutul registrului THR (sau, atunci când sunt activate memoriile FIFO, un octet din memoria FIFO a transmițătorului) este transferat automat în registrul TSR și începe transmisia. Dacă registrul TSR este golit (adică poate începe transmiterea unui nou caracter), iar registrul THR este gol (sau, atunci când sunt activate memoriile FIFO și memoria FIFO a emițătorului este golită sub un anumit nivel de prag), cipul generează o cerere de întrerupere (când acest tip de întrerupere este activat). Starea acestui registru poate fi determinată ***testând bitul 6 al LSR***.

Registrul RBR (Receiver Buffer Register) – Citire la AB+0

Este selectat ***dacă b7 este 0 în registrul LCR***. Reprezintă bufferul receptorului, astfel că atunci când interfața recepționează un caracter, acesta este plasat în acest registru. Caracterul recepționat trebuie citit de către software din registrul RBR înainte de a veni un caracter nou; în caz contrar, va apărea o eroare de depășire sau de suprapunere. Recepția unui caracter generează o cerere de întrerupere (dacă acest tip de întrerupere este activat). (La 16550, dacă memoriile FIFO sunt activate, este posibil ca întreruperea de la recepție să fie generată numai după ce a primit un anumit număr de caractere în memoria FIFO de recepție. Această valoare de prag poate fi programată prin setarea bitului 7 și a bitului 6 din registrul FCR.) Prezența unui caracter în registrul RBR poate fi determinată prin ***testarea bitului 0 al registrului LSR***.

Registrul RSR (Receiver Shift Register) - Este un registru intern, neaccesibil prin software.
Fiecare caracter este prima dată recepționat aici. Când s-a terminat recepția unui nou caracter, conținutul registrului RSR este transferat automat în registrul RBR. (La 16550, dacă FIFO este activat, caracterele recepționate sunt plasate în memoria FIFO de recepție.)

Registrul IER (Interrupt Enable Register) - Citire sau Scriere la AB+1

Este selectat dacă b7 este 0 în registrul LCR. Când funcționează cu întreruperi, cipul UART poate genera patru tipuri de solicitări de întrerupere, cu nivele de prioritate diferite: întrerupere la recepția unui caracter (când RBR este plin), întrerupere la transmisia unui caracter (când THR este gol), întrerupere la schimbarea stării liniei de comunicatie (atunci când LSR se schimbă) și întrerupere la apariția unei erori pe linia de comunicație (atunci când MSR se schimbă). Registrul IER permite activarea sau dezactivarea individuală a acestor întreruperi (prin scrierea unei valori de 0 sau 1 pe bitul corespunzător din IER). Registrul permite operație de citire/ scriere, astfel încât setările curente pot fi interogate în orice moment (de exemplu, dacă dorim să mascăm o anumită întrerupere fără a le afecta pe celelalte). Structura acestui registru este ilustrată în Tabelul 6.

Tabel 6. Conținutul registrului IER (Interrupt Enable Register)

7	6	5	4	3	2	1	0
0	**0**	**LPM* (Low Power Mode)**	**SM* (Sleep Mode)**	**Modem Status Modificat**	**Line Status Modificat**	**THR gol**	**Dată disponibilă la recepție**
		0 – nu permite 1 – permite modul respectiv pt circuitul 16750		0 – dezactivată 1 - activată generarea întreruperii la **modificarea MSR**	0 – dezactivată 1 - activată generarea întreruperii la **modificarea LSR**	0 – dezactivată 1 - activată generarea întreruperii când **THR e golit**	0 – dezactivată 1 - activată generarea întreruperii când **RBR e umplut**

Bit 1 (*Enable THR Empty Interrupt*) permite generarea unei cereri de întrerupere atunci când registrul THR este gol, adică după transferul conținutului acestui registru în registrul TSR. Când are loc această întrerupere, un nou caracter poate fi scris în reg. THR.
Bitul 0 (*Enable Received Data Available Interrupt*) permite generarea unei cereri de întrerupere la primirea unui caracter. (La 16550, dacă memoriile FIFO sunt activate, acest bit permite, de asemenea, generarea întreruperii la depășirea intervalului de timp alocat.)

Registrul IIR - Interrupt Identification Register – Citire la AB+2

IIR este un registru care poate fi doar citit, acesta specificând dacă există vreo întrerupere în așteptare și mai mult, arată care dintre cele patru surse de întrerupere necesită atenție la un moment dat. Deși întreruperile generate de cipul UART partajează un singur nivel de întrerupere (IRQ4 pentru primul port serial și IRQ3 pentru al doilea), aceste întreruperi pot avea patru cauze diferite. Deoarece IIR poate raporta o singură întrerupere la un moment dat și este cu siguranță posibil să existe 2 sau mai multe întreruperi în așteptare, cipul UART le prioritizează cu diferite nivele de prioritate: (**0110**-cea mai prioritară) LSI>DAI>THREI>MSI (**0000**-cea mai puțin prioritară). Identificarea cauzei de întrerupere poate fi realizată prin testarea biților 3...0 din registrul IIR. Ceilalți biți ai registrului IIR (care nu sunt utilizați la 8250) indică starea memoriilor FIFO la cipul 16550. Semnificația biților din registrul IIR care permit identificarea cauzei unei întreruperi și modul în care se poate „șterge" valoarea întreruperii din IIR sunt prezentate în Tabelul 7 și Tabelul 8. Dacă există două sau mai multe întreruperi în așteptare și sistemul deservește întreruperea cu prioritatea mai mare, cipul UART înlocuiește valoarea din IIR cu codul de identificare al următoarei surse de întrerupere cu prioritate maximă.

Tabelul 7. Conținutul registrului IIR (Interrupt Identification Register)

7	6	5	4	3	2	1	0
FIFO enabled (16550)	**FIFO enabled (16550)**	**0**	**0**	**ID2 întrerupere (16550)**	**ID1 întrerupere**	**ID0 întrerupere**	**În așteptare**
				x x x **1** – nu există întrerupere în așteptare 0 **1 1 0** – modificare a stării liniei la recepție -> **LSI** **1 1 0 0** – depășire de timp la recepție (16550) 0 **1 0 0** – RBR plin (dată disponibilă la receptor) -> **DAI** 0 **0 1 0** – THR gol (s-a terminat transmisia datei)->**THREI** 0 **0 0 0** – modificare a stării modemului -> **MSI**			

Tabelul 8. Biții 3…0 ai registrului IIR utilizați pentru identificarea cauzei unei întreruperi

Biții 3 …0	Prioritate	Tip întrerupere	Cauza întreruperii	Resetarea întreruperii
0001	-	-	Nu există întrerupere în așteptare	-
0000	3 (minimă)	***Modificare stare modem (MSR)***	Modificare semnal CTS / DSR / RI sau DCD	***Citire registru MSR***
0010	2	***Terminare transmisie caracter (THR empty)***	Registrul THR e gol	***Citire reg. IIR*** sau ***scrierea unui caracter în THR***
0100	1	***Recepţie caracter (RBR full)***	Registrul RBR conţine un caracter recepţionat Saumem. FIFO de recepție e plină peste val. de prag	***Citire registru RBR*** sau golire memorie $FIFO_{RX}$ sub valoarea de prag
1100	1	*Time-out (16550)*	No characters have been read or placed from/into the receiver FIFO during 4 characters and there is at least one character in the FIFO memory	Citire registru RBR
0110	0 (maximă)	***Modificare stare linie (LSR)***	Eroare de suprapunere, de încadrare, de paritate sau de transmisie spații (BREAK)	***Citire registru LSR***

LCR (Line Control Register) – Citire sau Scriere la AB+3

Prin înscrierea registrului LCR, se permite ***setarea formatului caracterului*** și ***setarea accesului la diferiți regiștri la AB+0 și AB+1***. Prin citirea registrului LCR se returnează ultima valoare înscrisă în acesta. Structura registrului LCR este prezentată în Tabelul 9.

Tabelul 9. Conținutul LCR (Line Control Register)

7	6	5	4	3	2	1	0
DLAB (*Divisor Latch Access Bit*)	**(*Set Break*)**	**(*Stick Parity*)**	**(*Even Parity Select*)**	**(*Parity Enable*)**	**(*Nb. of Stop Bits*)**	**(*Word Length Select*)**	
0 -> acces la ***THR / RBR*** (la AB), ***IER*** (la AB+1) 1 -> acces la **DLR** ***DL_{Low}*** (at BA), ***DL_{High}*** (at BA+1)	0 -> stare normală a liniei 1 -> forțează linia la 0 sau space ("break")	x x 0 – *fără paritate* 0 0 1 – paritate ***impară*** 0 1 1 – paritate ***pară*** 1 0 1 – paritate **zero** 1 1 1 - paritate **unu**			0-> **1 stop bit**; 1-> **2 stop biți** pt. car. de 6/7/8 biți (sau **1,5 stop biți** pt. car. de 5 biți)	date codificate folosind 00 -> 5 biți/ caracter 01 -> 6 biți/ caracter 10 -> 7 biți/ caracter 11 -> 8 biți/ caracter	

Prin înscrierea LCR se poate stabili formatul caracterului la comunicația serială.

Semnificația biților registrului **LCR** este următoarea:

Bitul **b7** (***Divisor Latch Access Bit***) fixează accesul la regiștrii interfeței disponibili la adresele 3F8h (2F8h) și 3F9h (2F9h).

Dacă bitul 7 este 0, regiștrii accesibili sunt THR/RBR (la AB)***, respectiv IER*** (la AB+1)***.***

Dacă bitul 7 este 1*, regiștrii accesibili sunt cei utilizați pentru înscrierea divizorului (DL)* care stabilește debitul binar (***DL_{LOW} la AB și respectiv DL_{HIGH} la AB+1***).

Bitul **b6** (***Set Break***). Dacă este setat la 1, circuitul forțează linia de comunicație la nivelul 0 logic (space). Aceasta corespunde stării "break" a liniei, care permite atenționarea unui terminal aflat la distanță printr-o întrerupere generată la detectarea acestei stări a liniei. Linia poate fi adusă în starea normală prin setarea bitului 6 la 0.

Bitul de pauză transmite un semnal către sistemul de la distanță, atât timp cât există un bit de 1 programat pe bitul 6 în LCR. Această opțiune „break enable" nu trebuie utilizată atunci când se încearcă transmisia datelor. Semnalul de „break" provine din zilele comunicației teletype și se presupune că va întrerupe un program care rulează pe sistemul de la distanță.

Biții **b5b4b3** pot fi interpretați ca un singur câmp:
xx0 – *fără paritate*
001 – paritate ***impară***
011 – paritate ***pară***
101 – paritate **zero**
111 - paritate **unu**

Bitul **b5** (***Stick Parity***) permite transmiterea sau așteptarea unor biți de paritate cu valoare fixă, 0 sau 1:
0: Paritatea este testată în mod obișnuit, conform biților *Parity Enable* și *Even Parity Select* (b3 și b4);
1: Dacă bitul *Parity Enable* este 1, se transmit sau se verifică biți cu valoare fixă în locul bitului de paritate, conform bitului *Even Parity Select*. Dacă bitul Even Parity Select este 0, bitul de paritate este întotdeauna 0, iar dacă bitul Even Parity Select este 1, bitul de paritate este întotdeauna 1.

Bitul **b4** (***Even Parity Select***) indică tipul parității utilizate, dacă generarea și verificarea parității este validată prin bitul *Parity Enable*:
0: Paritate impară;
1: Paritate pară.

Bitul **b3** (***Parity Enable***) validează sau invalidează generarea și verificarea parității:
0: Generarea și verificarea parității este invalidată;
1: Generarea și verificarea parității este validată.

Scopul principal al bitului de paritate este detectarea unei posibile erori de transmisie. Dacă există un canal de comunicare serială lung, zgomotos sau necorespunzător, este posibil să se piardă informații în timpul transmisiei, astfel încât este puțin probabil ca suma biților să corespundă valorii parității. Site-ul destinatar poate detecta această „eroare de paritate" și poate raporta eroarea la transmițător. Valoarea „stick parity" în 1 produce întotdeauna un bit de paritate de zero sau unul (așa cum este ales bitul 4), astfel încât este posibil să se transmită în locul bitului de paritate întotdeauna un zero sau unul în timpul transmisiei (va fi un bit blocat în 0 sau în 1 în locul bitului de paritate real). Un posibil exemplu este atunci când se trimit caractere Ascii Extinse între mașini folosind seturi de simboluri / caractere diferite - în diferite țări de exemplu (acele caractere cu cod Ascii mai mare de 128 care sunt diferite vor apărea diferit pe cele 2 mașini). Setând dimensiunea caracterului la 7 biți și paritatea pentru a activa blocarea parității în 0, se poate elimina automat bitul cel mai semnificativ în timpul transmisiei, înlocuindu-l cu zero.
Bitul **b2** (***Number of Stop Bits***) indică numărul biților de stop generați sau așteptați de circuitul UART:
0: se va folosi **1 bit** de stop;
1: se vor folosi **2 biți** de stop (sau **1,5 biți** dacă lungimea caracterelor este de 5 biți).
Receptorul testează doar primul bit de stop, indiferent de numărul biților de stop selectați (restul biților sunt folosiți doar pentru a asigura o mică întârziere, necesară dispozitivului periferic mai lent).
Bitul START este un semnal special care informează dispozitivul că sosesc datele pe linia serială. Biții de STOP reprezintă, în esență, absența unui bit de START, pentru a oferi o cantitate mică de timp între sosirea caracterelor consecutive pe linia serială. Adăugarea unui al doilea bit de STOP crește timpul de transmisie cu aproximativ 10%.
Biții **b1b0** (***Word Length Select***) determină lungimea caracterelor transmise sau recepționate (pe câți biți se vor codifica „datele"):
00: datele se vor codifica folosind **5** biți/caracter;
01: datele se vor codifica folosind **6** biți/caracter;
10: datele se vor codifica folosind **7** biți/caracter;
11: datele se vor codifica folosind **8** biți/caracter.
Se folosește deseori notația **"nrBițiPerCar. Paritate NrBițiStop"**, de exemplu **7N1** denotă **caractere codificate pe 7 biți, fără paritate (None), 1 bit de stop**, **8E2** denotă **caractere codificate pe 8 biți, paritate pară (Even), 2 biți de stop**.
UARTul poate transmite date seriale ca grup de 5, 6, 7 sau 8 biți. Totuși, la citire, se vor prelua întotdeauna 8 biți din registrul de recepție, deoarece UARTul înlocuiește toți cei mai semnificativi biți cu 0 (dacă primește mai puțin de 8 biți).
Comunicarea serială va rula cu aproximativ 10% mai rapid la o transmisie cu 7 biți/caracter, decât cu transmisie cu 8 biți/caracter.

LSR (Line Status Register) – Citire la AB+5

Acest registru indică starea liniei de comunicație. Biții 6..5 se referă la transmisie, iar biții 4..0 se referă la recepție (Tabel 10).

Tabel 10. Conținutul LSR (Line Status Register)

7	6	5	4	3	2	1	0
RxFIFO holds error (16550)	**Transmit empty**	**THR empty**	**Break detected**	**Framing error**	**Parity error**	**Overrun error**	**Receive data available**
	THR and TSR are empty	THR is empty	Break signal received				There is data available in RBR

Semnificația biților registrului LSR:
Bitul 0 (***Data Ready***) ***este setat la 1 atunci când a fost recepționat un caracter și acesta a fost transferat în registrul RBR*** sau în memoria FIFO. Acest bit este resetat automat prin citirea caracterului din registrul RBR sau în urma citirii tuturor caracterelor din memoria FIFO de recepție. Recepția unui caracter va genera o cerere de întrerupere dacă acest tip de întrerupere este validată.
Bitul 1 (***Overrun Error)*** indică o eroare de suprapunere. Este setat la 1 dacă se recepționează un nou caracter înaintea citirii caracterului din registrul RBR de către program. În acest caz, se pierd unul sau mai multe caractere. Eroarea de suprapunere, ca și celelalte erori, generează o cerere de întrerupere. Acest bit este resetat 0 prin citirea registrului LSR. Dacă memoriile FIFO sunt validate și memoria FIFO de recepție se umple peste nivelul de prag, o eroare de suprapunere va fi semnalată doar după ce memoria FIFO s-a umplut și următorul caracter a fost recepționat în registrul RSR.
Bitul 2 (***Parity Error***) indică o eroare de paritate, fiind setat la 1 dacă se recepționează un caracter cu paritatea diferită de cea așteptată. Acest bit este resetat prin citirea registrului LSR.
Bitul 3 (***Framing Error***) indică o eroare de încadrare, fiind setat la 1 dacă un caracter se recepționează fără biții de stop corespunzători. Receptorul testează numai primul bit de stop, indiferent de numărul biților de stop programați. La detectarea acestei erori, circuitul încearcă să se resincronizeze. Acest bit este resetat la 0 prin citirea registrului LSR.
Bitul 4 (***Break Interrupt***) este setat la 1 dacă sunt sesizate spații (0 logic) pe linie pentru o perioadă mai mare decât cea necesară pentru transmisia unui caracter. În acest caz, se înscrie un octet cu valoarea 0 în bufferul de recepție și se generează o cerere de întrerupere. Acest bit este resetat prin citirea registrului LSR.

Bitul 5 (***THR Empty***) ***este setat atunci când conținutul registrului THR este transferat în registrul TSR și se începe transmisia caracterului.*** Aceasta indică faptul că circuitul UART este pregătit pentru a accepta un nou caracter pentru transmisie. Atunci când registrul THR se golește, circuitul UART generează o cerere de întrerupere dacă acest tip de întrerupere este validată. Acest bit este resetat la 0 atunci când se înscrie un nou caracter în registrul THR. Dacă memoriile FIFO sunt validate, acest bit este setat atunci când memoria FIFO de transmisie se golește și este resetat la 0 atunci când se înscrie cel puțin un caracter în memoria FIFO de transmisie.
Bitul 6 (***TSR Empty***) ***este setat dacă atât registrul THR cât și registrul TSR s-au golit***. Dacă memoriile FIFO sunt validate, acest bit este setat la 1 atunci când atât memoria FIFO de transmisie cât și registrul TSR s-au golit.
Bitul 7 (***Error in Receiver FIFO***) este setat atunci când memoriile FIFO sunt validate și a apărut cel puțin o eroare de paritate, eroare de încadrare sau o condiție de "break" în timpul recepției caracterelor aflate în memoria FIFO de recepție.

MCR - Modem Control Register - Read or Write at BA+4

Registrul de control al modemului MCR se utilizează pentru controlul comunicației cu modemul. Structura registrului MCR este ilustrată în Tabelul 11.

Tabel 11. Conținutul MCR (Modem Control Register)

7	6	5	4	3	2	1	0
0	**0**	**0**	**Loopback test**	**OUT2**	**OUT1**	**Request to send**	**Data Terminal Ready**
reserved	reserved	reserved	Loopback mode if 1 (DTR, RTS, RI and DCD not connected)	AUX OUT2 (Interrupt enable)	AUX OUT1	RTS	DTR

Bit 3 - Bit-ul de întrerupere este un element specific PC-ului, deoarece designerii IBM au conectat această ieșire la o poartă externă pentru a activa sau dezactiva toate întreruperile de la UART. Acest bit trebuie setat (la 1) pentru a permite întreruperile.

MSR - Modem Status Register – Citire la AB+6

Acest registru conține informații despre starea modemului. Semnificația biților registrului MSR este dată în Tabelul 12.
Biții 7..4 indică starea curentă a semnalelor de handshake CD, RI, DSR, respectiv CTS. Un semnal activ este indicat prin bitul corespunzător din registrul MSR setat la 1.
Biții 3..0 indică modificarea stării semnalelor CD, RI, DSR, respectiv CTS, de la ultima citire a registrului MSR. Acești biți sunt resetați la citirea registrului MSR.

Tabel 12. Conținutul MSR (Modem Status Register)

7	6	5	4	3	2	1	0
Data Carrier detect	**DSR**	**RI**	**CTS**	**Delta DCD**	**Delta DSR**	**Delta RI**	**Delta CTS**

FCR (FIFO Control Register) – Scriere la AB+2 – doar la 16550

Reprezintă registrul de control al memoriilor FIFO, fiind disponibil la circuitul UART 16550 și la următoarele. Structura registrului FCR este ilustrată în Figura 11.

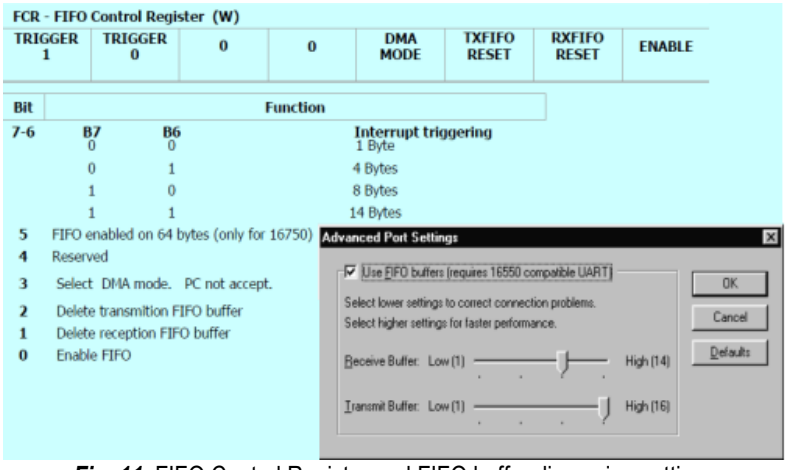


Fig. 11. FIFO Control Register and FIFO buffer dimension settings

SPR (Scratchpad Register) – doar la 16550

Acest registru nu este utilizat pentru comunicație; poate fi utilizat pt memorarea temporară a unui octet. Scopul real al utilizării lui a fost de a discerne tipurile diferite de UART 8250/8250B sau 8250A/16450 (deși 8250/8250B nu au fost proiectate pt uz în PC-AT).

D7	D6	D5	D4	D3	D2	D1	D0

Fig. 12. Conținutul registrului SPR

4.3. Programarea UART

Secvența de programare UART implică în practică următorii pași:

Setarea parametrilor liniei de comunicație (aceleași informații) între transmițător (Tx) și receptor (Rx) înainte de transmisia/ recepția efectivă a caracterelor:

1. Se configurează ***Rata de transfer a datelor***: cu ce viteză să circule biții pe linia serială -> se va fixa acces la DL și apoi se vor programa DL_{LOW} și DL_{HIGH} cu constantele specifice ratei de transfer dorite;
2. Se configurează ***Formatul caracterului***: se va fixa acces la ceilalți regiștri ai interfeței (de Tx și Rx, pregătind astfel pasul 3) și se va programa: numărul de biți folosiți pentru date, numărul de biți de stop, dacă se folosește sau nu paritate (și tipul parității);
3. Se programează ***Transmisia/ recepția efectivă a caracterelor*** între cele 2 entități implicate în comunicație.

În plus, în caz că se dorește ***funcționarea interfeței prin întreruperi la Tx sau Rx***, se vor mai programa și regiștrii specifici generării de întreruperi (înainte de pasul 3):

- in the **IER** register, if b_1 is 1, an interrupt will appear on transmission and if b_0 is 1, an interrupt will appear on reception;
- in the **MCR** register, b_3 should be 1 to enable interrupts.

Sub formă de pseudoinstrucțiuni, putem rescrie acești pași astfel:

Step	Pseudoinstrucțiune	Explicații
1. Programăm **Rata TX/RX**	(BA+3) <- un octet cu b_7 = 1	scrie LCR: b_7 = 1 (nu conteză b_6…b_0 pt că sunt suprascrișI în pas 2) Pt a fixa acces la DL (16 biți, deci DL_{LOW} și apoi DL_{HIGH})
	(BA) <- low DL	scrie DL_{LOW} cu octetul Low al constantei pt a obține rata dorită
	(BA+1) <- high DL	scrie DL_{HIGH} cu octetul High al constantei pt a obține rata dorită
2. Programăm **Formatul caracterului**	(BA+3) <- un octet cu b_7 = 0	scrie LCR: b_7 = 0 pt a fixa acces la reg de date THR și RBR (necesari în pas 3) b_6 forced break, $b_5b_4b_3$ parity, b_2 nb. of stop bits, b_1b_0 nb. of bits/character
3. **Transmisie un caracter**	(BA) <- AL	(accesul la THR este garantat în pas 2) Scrie octet din registrul AL în THR
3. **Recepție un caracter**	(BA) -> AL	(accesul la RBR este garantat în pas 2) Citește octet din RBR în registrul AL

5. Adrese și întreruperi rezervate portului serial

Programul BIOS al calculatoarelor IBM PC originale permitea utilizarea a două porturi seriale, cu numele COM1 și COM2. Ulterior, numărul porturilor a fost extins cu încă 2, cu numele COM3 și COM4. Începând cu sistemul de operare Windows 95, numărul porturilor seriale a fost extins la 128, aceste porturi fiind gestionate prin driverele de dispozitiv care le controlează.

Accesul la porturile seriale se poate realiza prin funcții BIOS (întreruperea 14h), prin funcții ale sistemului de operare, sau direct prin regiștrii circuitelor UART. Fiecare circuit UART asociat unui port serial dispune de un număr de opt regiștri începând de la adresa de bază (AB) a portului serial.

Programul BIOS memorează adresele de bază ale porturilor seriale COM1…COM4 în patru cuvinte succesive de 16 biți, începând cu adresa 0000:0400h corespunzătoare portului COM1. Adresele de bază ale porturilor seriale COM1…COM4 sunt indicate în Tabelul 13. În general, adresele porturilor COM1 și COM2 sunt fixe (având valorile indicate în tabel), dar adresele porturilor COM3 și COM4 pot fi diferite de cele indicate. În Tabelul 13 se indică și nivelele de întrerupere utilizate de porturile seriale COM1…COM4 (adresele porturilor paralele încep de la adresa 0000:0408h). Următorul program arată cum putem citi aceste locații de memorie pentru a obține adresele porturilor seriale instalate în sistem.

Tabel 13. Asignarea standard a adreselor de bază și a nivelelor de întrerupere la porturile seriale

Name	Base addr.	Intr. (IRQ)	COM addr. in BIOS area
COM 1	3F8	4	0000:0400
COM 2	2F8	3	0000:0402
COM 3	3E8	4	0000:0404
COM 4	2E8	3	0000:0406

```
#include <stdio.h>
#include <dos.h>
void main(void)
{
unsigned int far *indadr; /* An indicator for the
location of the port address */
unsigned int adr_port; /* The port address */
int i;
indadr=(unsigned int far *)0x00000400;
for (i = 0; i < 4; i++)
{
adr_port = *indadr;
if (adr_port == 0)
printf("There is no port for COM%d \n",i+1);
else
printf("The address assigned for COM%d is
%Xh\n",i+1,adr_port);
*indadr++;
}
}
```

6. Servicii BIOS pentru PORTUL SERIAL (INT 14h)

Funcțiile BIOS pentru comunicația serial asincronă sunt accesibile prin întreruperea 14h, iar pentru controlul prin întreruperi al interfeței seriale este necesară scrierea unor rutine proprii, deoarece funcțiile BIOS permit doar transmisia și recepția prin testarea stării. În general, funcțiile returnează starea liniei în registrul AH și cea a modemului în registrul AL. Semnificația biților celor doi octeți de stare este următoarea:

AH (Line status)		AL (Modem Status)	
Bit	**Description**	**Bit**	**Description**
7	(Time-Out Error): Time-out for transmission or reception	**7**	(Carrier Detect): CD status
6	(TSR Empty): the character from TSR is transmitted	**6**	(Rind Indicator): RI status
5	(THR Empty): the THR register is empty	**5**	(Data Set Ready): DSR status
4	(Break Detect): the line is kept to "0"	**4**	(Clear To Send): CTS status
3	(Framing Error): the stop bit is not received	**3**	(Delta Carrier Detect): change of the DCD status
2	(Parity Error): parity error	**2**	(Trailing Edge Ring Indicator): RI = 1
1	(Overrun Error): overrun error	**1**	(Delta Data Set Ready): change of the DSR status
0	(Data Ready): a character is stored in the Reception register	**0**	(Delta Clear To Send): change of the CTS status

Funcțiile BIOS apelabile cu INT 14h sunt :

Serviciu	Funcție	Intrare	Ieșire
AH = 00	**Inițializare Port Serial**	**AL** = parametrii comunicației Biții 1 – 0: character length (10 – 7 bits; 11 – 8 bits) Bitul 2: stop bits (0 – 1 stop bit; 1 – 2 stop bits) Biții 4 – 3: parity control (01 – odd parity, 11 – even parity, x0 – no parity) Biții 7 – 5: baud rate (000 – 110 bps, … 111 – 9600 bps) **DX** = numărul portului serial (0-3)	**AH** = Line Status **AL** = Modem Status
AH = 01	**Transmisie caracter prin port serial**	**AL** = caracter **DX** = numărul portului serial (0-3)	**AH**: b7 = 0 if correct b7 = 1 if error **AH**: b6…0 = status
AH = 02	**Recepția caracter prin prin port serial**	**DX** = numărul portului serial (0-3)	**AH**: b7 = 0 if correct b7 = 1 if error **AH**: b6…0 = status **AL** = character
AH = 03	**Starea portului serial**	**DX** = numărul portului serial (0-3)	**AH** = Line Status **AL** = Modem Status

Setarea param liniei folosind int 14h, cu serviciul 0h (Inițializare port serial):

mov AH, 0
mov DX, 0 ; se vrea COM1, valorile sunt între 0-3, în funcție de portul COM1,2,3,4
mov AL, **1000 1110**b ;**1200 baud**, **odd parity**, **2b of STOP**, **7 bits/character**; **D2h -> 7200 baud**, **no parity**, **1b of STOP**, **7 bits/character;**
int 14h
; pt mov AL, **111**0 0**111**b => 9600 baud, fără paritate, 2b STOP, 8 biți/car

Transmisie caracter folosind int 14h, cu serviciul 1h:

mov AH, 1
mov DX, 0 ; dacă se vrea COM1, valorile sunt între 0-3, în funcție de portul COM1,2,3,4
mov AL, **caracter** ; caracterul ce se dorește a fi transmis prin linia serială
int 14h

Recepție caracter folosind int 14h, cu serviciul 2h:

mov AH, 2
mov DX, 0 ; se vrea COM1, valorile sunt între 0-3, în funcție de portul COM1,2,3,4
int 14h
; în AL vom avea **caracterul** venit (recepționat) pe linia serială

Citirea stării liniei folosind int 14h, cu serviciul 3h:

mov AH, 3
mov DX, 0 ; se vrea COM1, valorile sunt între 0-3, în funcție de portul COM1,2,3,4
int 14h
; în AL vom avea **starea modemului**
; în AH vom avea **starea liniei**
Dacă în continuare vom folosi instrucțiuni de testare a anumitor biți din AH, putem interpreta starea liiei
TEST AH, 1Eh ; testarea biților de eroare | TEST AH, 01h ; testarea receptorului | TEST AH, 20h ; testarea transmițătorului

APLICAȚII în LAB:

PC1 (comx) -> it transmits A, n, and it receives 1,2	PC2 (comx) -> it transmits 1,2, and it receives A, n
Type a character at the keyboard (ESC when done) Transmitted: A Received: 1 Transmitted: n Received: 2	Type a character at the keyboard (ESC when done) Transmitted: 1 Received: A Transmitted: 2 Received: n

Application 1.

COMX.cpp - > /*The Program sends characters from the keyboard on COM1 and receives them also on COM1, using in/out instructions in a specified order in the program: it alternates transmission with reception of characters; pins 2-3, 4-5 are connected to COM1 connector*/

```
#define COM1_PORT 0x3F8
#define TXD COM1_PORT; THR
#define LSR (COM1_PORT+5) //LSR at 0x3FD
#define LCR (COM1_PORT+3) //CLR at 0x3FB
#define ESC 0x1b
#include <conio.h>
#include <stdio.h>
#include <stdafx.h>
#include<stdlib.h>
#include <dos.h>
#include <windows.h>
#include "pt_ioctl.c"
/*The Program send and receive character on the COM1 - TxD conected to RxD */

void init_com1 (void);          //baud, line param
void txd_com1 (char);           //transmit char
char rxd_com1(void);            //receive char

int main (void)
{ char car_in, car_out;
  char str[256];

OpenPortTalk();
init_com1();
sprintf_s(str, "\nType a character at the keyboard (ESC when done) : \n %d \n", 0);
OutputDebugString(str);
  do {cout<<"Transmitted :";
          car_out=getche();
          txd_com1(car_out); //-> THR
          car_in=rxd_com1();
          printf("\tReceived:%c \n",car_in); }
  while (car_out!=ESC);

ClosePortTalk();
return(0);
          }

void init_com1(void)
{  outportb(LCR,0x80);  //B7=1 to acces the divisors
   outportb(TXD,0x0c);  //load divisor registers
   outportb(TXD+1,0);  //9600bps
   outportb(LCR,3);  //B7=0-acces TDR,RDR;
                     // 8b/char,2b Stop,no parity
}

void txd_com1(char car)
{ char state;
do  { state=inportb(LSR) & 0x20; } // is bit 5 (LSR)=1?
          while (state!=0x20);
          outport(TXD,(char)car);
}

char rxd_com1(void)
{ char state;
   do { state=inportb(LSR) &0x01; } // is bit 0 (LSR)=1?
      while (state!=0x01);
          return((char) inportb(TXD));
}
```

Application 2.

SERIAL .asm -> /*The Program sends characters from the keyboard on COM1 and receives them also on COM1 by using interupts; pins 2-3, 4-5 are connected to COM1 connector*/

```
STIVA SEGMENT PARA STACK 'STACK'
        DW 256 DUP (?)
STIVA ENDS
DATA SEGMENT PARA PUBLIC 'DATA'
MESSAGE DB 'Serial Communication program by using interrupts '
 DB  0dh,0ah
 DB  'to finish press ESC !',0DH,0AH,'$'
IRQ4_old dw 2 dup(?); COM1 (0Ch)
DATA ENDS

CODE SEGMENT PARA PUBLIC 'CODE'
ASSUME CS:CODE,DS:DATA,SS:STIVA,ES:NOTHING
COM1 equ 3F8h

START:  PUSH DS             ; DS init
        XOR  AX,AX
        PUSH  AX
        MOV  AX,DATA
        MOV  DS,AX
        MOV AX,0600h        ; clear screen
        MOV DX,0
        MOV CX,164Fh
        MOV BH,7
        INT 10h
        MOV DX,OFFSET MESSAGE;print message
        MOV  AH,9
        INT  21H
; continues on next page …
```

; initialise serial line

```
        MOV DX,COM1+3 ; baud rate
        MOV AL,80H
        OUT DX,AL    ; b7=1 -> "DL"
        MOV DX, COM1
        MOV AL,0CH
        OUT DX,AL    ; DL low
        MOV DX, COM1+1
        MOV AL,0
        OUT DX,AL    ; DL high => 9600bps
        MOV DX, COM1+3   ; LCR b7=0, b6…b0
        MOV AL,00010011B ; even parity
        OUT DX,AL
        MOV DX, COM1+4 ;
        MOV AL,00001011B
        OUT DX,AL
**************************************
CLI    ; IF=0 – > do not accept interrupts yet
;modify the address of the IRQ4 (COM1) interrupt
        MOV AX,350Ch
        int 21h
        MOV IRQ4_old, BX
        MOV IRQ4_old+2, ES

        PUSH DS
        MOV DX,OFFSET INT_COM1
        MOV AX,seg INT_COM1
        MOV DS, AX
        MOV AX, 250Ch
        int 21h
STI
        MOV DX,COM1+1    ; IER – interrupt on RBR full
        MOV AL,1
        OUT DX,AL
CLI
        MOV AL,0A0h ; 1010 0000b
        OUT 21h,AL
STI
WAIT:   MOV AH,0
        INT 16h        ;read char from the keyboard
        CMP AL,1Bh  ;test if ESC was pressed
        JZ  END

        MOV  DX, COM1  ;send char on the serial line
        OUT DX,AL
        JMP     WAIT

END:    MOV AL,0B0h         ;remask  IRQ4
        OUT 21h, AL
CLI
PUSH DS
        MOV AX, 250Ch
        LDS DX, dword ptr IRQ4_old
        int 21h
POP DS
STI

        MOV  AX,4C00h      ;quit the program
        INT 21h
```

```
INT_COM1 PROC FAR ;intrerrupt on IRQ4 line at PIC8259
    MOV DX, COM1
    IN AL,DX         ;read char from serial
PUSH AX
        MOV AH, 0Eh
        MOV BX, 0
        INT 10h      ;print char
POP AX
    CMP  AL,0Dh   ;if CR, add also LF
    JNZ EOI
    MOV  AL,0Ah
    MOV BX,0
    MOV AH,0Eh
    INT 10h

EOI:
        MOV AL,20h    ;send EOI to PIC8259
        OUT 20h,AL
    IRET                  ;end of the interrupt
INT_COM1  ENDP
CODE ENDS
        END START  ;end program
```

Study the application and answer the following:

- What represent the code written with blue color?
- When is an interrupt appearing? At transmission (when a character has been sent, i.e. THR or TSR is empty?), at reception (when a character has been received, i.e. RBR is full?), or at both events?
- What is the main purpose of STI and CLI instructions?
- When is the INT_COM1 routine executed in the program?
- What is the purpose of the pseudointruction (21h)<-A0h ?

Application 3: Make a serial connection between two PCs using COM1 ports.
The application allows to change text messages between 2 users (chat) **using INT 14h sevices**.

Homework: Rewrite the application using hardware interrupts.
Data: **COM1: 9600Baud, 2 STOP bits, No parity;**

```
code segment para 'code'
assume cs:code,ds:code,ss:nothing

org 100h

startHere: mov ah,0              ;initialise UART
           mov dx,0 ;COM1
           mov al,11100111b ;9600 baud,no parity.,2STOP, 8 bits/char
           int 14h
state:     mov ah,3 ; UART state service
           mov dx,0 ;com1 – get state of the line in AH
           int 14h
           test ah,1eh   ;test the error bits + break
           jnz err1    ; if any of the bits is set, somewhere was an error
           test ah,01h   ;test the receiver(b0) – is there any character
                         ; received in the RBR buffer?
     jnz receiver ; if b0=1, there is a character in RBR -> we go to
                  ; take it and print it
           test ah,20h  ;test buffer tr. (b6b5)- transmission succeeded?
jz state     ; THRE=1? (NOT YET? Then we wait, go back in loop)
           mov ah,1              ;wait for a key
           int 16h     ; z=0
           jz state
           mov ah,0     ;read a key
           int 16h       ;al= key pressed
           cmp al,1Bh  ;al= ESC ?
           jz done      ;we send the character
           mov ah,1  ;the service for transmission of the character
           mov dx,0
           int 14h
           jmp state

receiver:
           mov ah,2  ; the service for reception of the character
           mov dx,0
           int 14h
           push ax   ;save the character
           mov bx,0    ;printing attribute
           mov ah,0eh
           int 10h       ;print character (the one received) on the screen
           pop ax
           cmp al,0Dh  ;is it CR?
           jnz state
           mov al,0ah  ;print LF (enter)
           mov bx,0
           mov ah,0eh
           int 10h
           jmp state
err1:      mov ah,2 ; take the character from the receiver buffer
           mov dx,0
           int 14h
           mov al,'?' ; it was a character with an error
           mov bx,0
           mov ah,0eh
           int 10h
           jmp state
done:      int 20h
code ends
end startHere
```

HomeWork: Find and study materials about the VIRTUAL COMMUNICATIONS PORT (VCP).

```
#include <bios.h>
#include <conio.h>
#define COM1 0
#define DATA_READY 0x100
#define SETTINGS ( 0x80 | 0x02 | 0x00 | 0x00)
int main(void)
{ int in, out, status;
bioscom(0, SETTINGS, COM1); /*initialize the port*/
cprintf("Data sent to you: ");
while (1)
{ status = bioscom(3, 0, COM1); /*wait until get a data*/
  if (status & DATA_READY)
     if ((out = bioscom(2, 0, COM1) & 0x7F) != 0) /*input a data*/
        putch(out);
     if (kbhit())  {  if ((in = getch()) == 27) /* ASCII of Esc*/
                         break;
                         bioscom(1, in, COM1); /*output a data*/
                   }
}
return 0;
}
```

Temă: Studiați următoarele exemple simple

Exemplul 1: presupunând că se folosesc 8b/caracter, 2b de STOP, arătaţi cum se încadrează fiecare caracter; cu cât (%) se suplimentează la transfer, informația?
Sol: 8b/car, 1b START, 2 b STOP => 11b în loc de 8b pentru a transmite un caracter => mai mult de 30%
8b ... 100%
3b ... x
=> x= 37,5% , deci cu 37,5% se transmit mai mulţi biţi (se încarcă suplimentar linia de comunicație serială)

Exemplul 2: Să se calculeze nr de biţi folosit în transferul a 5 pagini de date, fiecare pagină cu 80x25 caractere. Se pp. fol. a 8b/car şi 1 bit de STOP.
Sol: pt fiecare caracter => 10b/caracter
Nr de biţi pt o pagină: 80x25x10 = 20.000 biţi/ pagină şi se transmit 5 pagini => 100.000 biţi în total

Exemplul 3: Calculaţi timpul necesar pt a transfera un nr de 5 pagini de date pt Ex 1-1, folosind: a) 2400 bps, b) 9600bps
Sol:
a) 100 000 / 2400 =41,67 sec
b) 100 000 / 9600 =10,4 sec

Exemplul 4: Analizați scenariile extreme de încărcare ale liniei: cum aflăm încărcarea minima, resp. maximă?
Sol:
Min: 1b start, 1 b stop, fără paritate, 8b/car (încărcare minimă, cu date de dimensiune maximă)=> 10b în loc de 8b => x=2*100/8 = 25%
Max: 1b start, 2 b stop, cu paritate, 5b/car (încărcare maximă, cu date de dimensiune minimă) => 9b în loc de 5b => x=4*100/5 =80%

Exemplul 5 Dacă ar fi încărcată în Divisor Latch valoarea 12h, cât ar fi debitul setat?
Sol: **Divizor = 1.843.200 / (DebitBinar * 16)** sau **DebitBinar = 1.843.200 / (Divizor * 16); => DebitBinar = 1.843.200 / (18 * 16)=6400** (valoarea nu este în Tabelul cu valorile uzuale pentru debit)

Exemplul 6: Dacă s-a programat valoarea 01101**01**b în LCR, cu câți biți s-a setat transmisia/recepția caracterului? Câți biți de STOP?Cum e paritatea? La care reg s-a fixat acces?
Sol: b1b0=01=> 6 biți ; b2=1 și sunt 6 biți/car => 2b STOP; b543=101 => paritate zero, care nu depinde de character; b7=0, vom avea acces la THR, RBR și IER

Exemplul 7: Dacă s-a programat valoarea 11001**00**b în LCR, cu câți biți s-a setat transmisia/recepția caracterului? Câți biți de STOP? Cum e paritatea? La care reg s-a fixat acces?
Sol: b1b0=00=> 5 biți ; b2=1 și sunt 5 biți/car => 1,5b STOP; b543=001 => paritate impară, care depinde de character; b7=1, vom avea acces la regiștrii Divisor Latch (Low și High)

Exemplul 8: Dacă s-a ales comunicarea pe linia serială folosind 7b/car, 1 b stop, paritate impară, cum se va transmite prin pinul TX (format RS232) caracterul 'B'? (problema poate fi data și invers, se dă o formă de undă -> ce caracter s-a transmis ? vezi pb 3, pct c), d))

Exemplul 9: Dacă se pp. că se transmit caracterere pe 7 biți, cu 1 bit de Stop și bit de paritate, câte caractere se vor putea transmite într-o secundă pe linia serială la un baud de 300 bps? Dar dacă am codifica aceleași caractere pe 8 biți?
Sol: a) Baud-ul (debitul la transmisia caracterelor aici) implică transmiterea tuturor celor 10 biți corespunzători unui character, nu doar cei utili (7). Astfel, într-o secundă, se vor putea transmite 30 caractere.
b) vom putea transmite maxim 27 caractere. (pt că 300 : 11 = 27,27 caractere)

Exemplul 10: Un inginer de sistem cunoscut la nivel național a aruncat o privire folosind DEBUG la locațiile de memorie începând de la adresa 0040: 0007 și a obținut următoarele informații. El dorește să afle numărul de porturi COM instalate în acest computer și care anume este instalat. Care credeți că ar trebui să fie răspunsul lui?
C>DEBUG
-d 0040:0000 L08
0040:0000 F8 03 00 00 00 00 00 00
Sol: Dispunerea memoriei a arătat că există un singur port COM (COM 1 este la adresa 0000: 04000) instalat în acest computer, cu adresa de pornire sau de bază 03F8h.

Temă: Studiați următoarele exemple

Pb1) Să se scrie o secvență de program care să inițializeze o linie de comunicație serială să lucreze cu rata de baud de 9600 bps și să transmită fiecare caracter codificat pe 7 biți, cu 1 bit de stop și paritate pară; se știe că adresa portului serial este dată de variabila COM1 definită în program cu adresa portului COM1 unde este legat dispozitivul de comunicare serială, de ex. la adresa de bază 3F8h.

Sol: Secvența de ***setare rată de baud***:

A)

```
MOV   DX,COM1+3 ; trebuie să facem b7 (LCR)=1 pt a fixa acces la Divizor Latch
MOV    AL,80H
OUT    DX,AL

MOV   DX, COM1 ; trimitem în DL partea low a valorii necesare pt un baud rate de 9600 bps: 0Ch
MOV    AL,0CH
OUT    DX,AL

MOV   DX, COM1+1 ; trimitem în DL partea high a valorii necesare pt un baud rate de 9600 bps: 00h
MOV    AL,0
OUT    DX,AL
```

Secvența de ***setare parametri de încadrare***:

B)

```
MOV    DX, COM1+3; trebuie să facem b7 (LCR)=0 pt a fixa acces la ceilalți regiștrii: THR și RBR, sau IER, și tot aici specificăm param de la încadrare: 7 biți per caracter, 1 bit de stop, paritate pară, fără break
MOV    AL,00011010B
OUT    DX,AL
```

;dacă s-ar fi specificat paritate zero, atunci am fi folosit: **101**, iar la transmisia sau recepția caracterului pe linie, bitul de paritate ar fi fost întotdeauna de valoare 0 (la paritatea de tip *zero* sau *unu*, se vede că bitul 5 Parity Stick e blocat în 1, ceea ce arată că bitul de paritate nu mai depinde de caracter, e fixat: în 0 sau 1); pt paritate 1, am fi folosit **111**
Dacă am fi avut fără paritate, atunci am fi folosit **xx0** și la încadrare am fi "economisit" un bit (pe linie, se transmitea cu 1 bit mai puțin: în loc de 10 biți, am fi transmis doar 9 biți pt fiecare caracter)

C) Considerând că linia a fost inițializată, să se scrie o secvență care să transmită caracterul ‚A' pe linie

```
Sol:    mov AL, 'A'
        mov  DX, COM1        ; trimite caracterul pe linia serială
        out   DX, AL
```

Să se scrie o secvență prin care să se citească un caracter de pe linia serială:

```
Sol:    mov  DX,COM1        ; preia caracterul de pe linia serială
        in   AL,DX
```

Pb2: Fie conținutul memoriei așa cum e ilustrat în tabel:

Adresa, începând de la 0000:0400h	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D
Conținutul	F8h	03h	F8h	02h	E8h	03h	E8h	02h	BCh	03h	78h	03h	78h	02h

a) Specificați/ Încercuiți în tabel adresa de bază a portului COM2. Scrieți adresa portului și aici: _____

b) Se consideră că se dorește inițializarea liniei de comunicație serială astfel încât să poată recepționa date la un debit de 4800bps, cu 2 biti de stop, paritate pară, iar caracterele se vor codifica pe 8 biți. Scrieți secvența de inițializare a liniei de comunicație serială:

```
b1) mov DX, _____h      mov DX, _____h
    mov AL, _________b  mov AL, _________b
    out DX, AL          out DX, AL
    mov DX, _____h      mov DX, _____h
    mov AL,_________    mov AL,_________
    out DX, AL          out DX, AL
```

Se cere să explicați cum se calculează:

b2) valoarea înscrisă: _____ pt a stabili rata de baud

b3) calcul bit de paritate: _____ ;

Ceilalți parametri ai liniei:

b4) scrieți o secvență de instrucțiuni pentru a transmite un caracter al cărui cod Ascii se află în registrul DL pe linia serială: (cu comentarii):

Soluție:

a) BIOS-ul, pe durata POST (Power On Self Test) depune în memoria RAM, de la adresa 0400h, în format Little END-ian, adresele porturilor seriale de tip COM, în ordine: COM1, apoi COM2, apoi COM3 și în final, COM4 dacă sunt instalate.

Pt COM2, avem:

F8h	02h

Deci adresa de port va fi **02F8**h, numită deseori și "adresă de bază"

b) **b2)** pt un debit de **4800** bps, vom avea relația: divisor = 1.843.200 / (**debit** *16) = 24 = 18h=**0018**h

astfel, va trebui să încărcăm în DL (Divisor Latch) valoarea 0018h; cum interfața are doar 8 biți de date, vom scrie în 2 reprize; DL se află la adresa de bază (partea low) și respectiv la adresa de bază +1 (partea high):

```
mov DX, 2F8h
mov AL, 18h
out DX, AL
mov DX, 2F9h
mov AL,0
out DX, AL
```

DAR, înainte de a trimite această valoare înspre DL, trebuie să ne asigurăm că este fixat acces înspre el; aceasta se face prin înscrierea bitului 7 din LCR (de la adresa de bază +3) în 1:

```
mov DX, 2FBh          ; LCR<- b7=1
mov AL, 1xxx.xxxxb ; obligatoriu b7=1, restul nu contează – în gen. x=0
out DX, AL
```

b3) ceilalți param ai liniei (precum: 2 biti de stop, paritate pară, caractere codificate pe 8 biți) se vor specifica pe biții 5,4,3,2,1,0 ai octetului care va fixa acces spre ceilalți registri ai interfeței UART (Reg. THR, RBR sau IER)- deci acest octet va trebui să aibă b7=0.
Biții 5,4,3 specifică paritatea: 001 – paritate impară, resp **011** – ***paritate pară***
Bitul 2 trebuie să fie 1 pt a specifica 2 b STOP, iar biții 1 și 0 trebuie să fie 11 pt a codifica un caracter pe 8 biți => **0**0011111b
Deci vom avea:

```
mov DX, 2FBh    ; LCR
mov AL, 00011111b
out DX, AL
```

b4) mov AL, DL ; îl punem în AL – obligatoriu, nu se poate folosi alt registru pt transfer pe 8 biți cu portul

```
mov DX, 2F8h    ; îl trimitem la adresa de bază, în THR
out DX, AL
```

Pb3: Al patrulea port serial dintr-un PC IBM are adresa de bază 2E8h**.**
a1) La ce adresă se va accesa LCR (Line Control Register) ?
Sol: Base Address+3 => 2E8h+3=> 2EBh
a2) De la ce adresă de port trebuie luat octetul recepționat prin linia serială?
Sol: de la adresa de bază (unde este registrul de recepție date RBR) => 2E8h
a3) Scrieți o secvență de program care să fixeze apariția unei întreruperi înspre PIC8259 la TX unui caracter pe linia serială:
Sol: din Interrupt Enable Register (IER) => "THR empty" (bit 1) = 1

```
mov AL 0xxx xxxxb   ; b7(LCR)=0 pt a stabili acces la IER
mov DX, 2EBh        ;
out DX, AL
mov AL, 02h         ; când "THR is empty" se trimite un semnal de întrerupere pe linia IRQ 3
mov DX, 2E9h        ; conținutul IER trebuie să fie 02h pt a avea întrerupere la Tx
out DX, AL
; COM1 și COM3 partajează aceeași linie de întrerupere: IRQ4    ; COM2 și COM4 partajează aceeași linie de întrerupere: IRQ3
```

a4) presupunând că apar consecutiv mai multe întreruperi la PIC (de la timer, de la tastatură, de la FD, COM1 și COM2 și că IF=1), scrieți masca ce trebuie trimisă în registrul IMR al PIC 8259 a.î. să fie luată în tratare întreruperea necesară programului dv (ea prima):
Sol: IRR: 0101 1011
Mask : IMR: XXXX 0X11, x-nu contează deoarece *ori* nu s-a recepționat întrerupere de la acel dispoz, *ori* are o linie cu prioritate mai mică (și deci va fi tratată după întreruperea noastră, care apare pe linia IRQ3 – pt COM 2 – aceasta în cazul că întreruperile apar deodată)
În reg IRR și IMR, *bitul x* corespunde la:

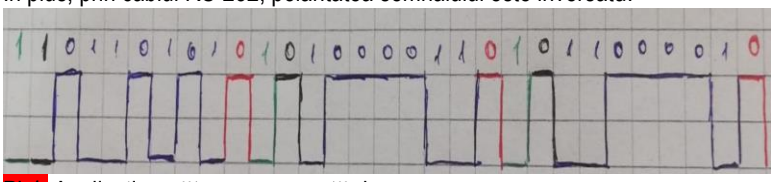
Bit 0 : timer
Bit 1: keyboard
Bit 2: PIC slave or left empty
Bit 3: COM2
Bit 4: COM1
Bit 5: HDD
Bit 6: FD
Bit 7: device connected on parallel port (LTP device)

- dacă cerința spune: "doar ea, restul NU", atunci se vor masca toate celelalte care apar, înafară de cea dorită, indiferent de prioritate
- în plus, există riscul ca o altă întrerupere, deși mai puțin prioritară să apară și să ne încurce – și atunci mascăm tot, indiferent de prioritate, o păstrăm doar pe a noastră, probabil și pe cea de la tastatură ca să putem ieși afară din program la apăsarea unei taste și probabil o păstrăm și pe cea de la timer dacă avem nevoie de noțiunea de timp în aplicație
b) Se consideră următoarea secvență de instrucțiuni:

```
mov DX, 2EBh
mov AL, 10000000b
out DX, AL
mov DX, 2E8h
mov AL, 12h
out DX, AL
mov DX, 2E9h
mov AL, 0
out DX, AL
mov DX, 2EBh
mov AL,00001010b
out DX, AL
```

În care noi trebuie să observăm că (fără să ne spună problema):
; primele 3 instrucțiuni coresp. unei operații în care setăm b7(LCR)=1, și deci fixăm acces la Divisor Latch: la AB și la AB +1 vom avea: DL-Low, resp.DL-High
; urm. 3 instrucțiuni coresp. unei operații în care scriem în Divisor Latch- partea low
; urm. 3 instrucțiuni coresp. unei operații în care scriem în Divisor Latch- partea high
; ultimele 3 instrucțiuni coresp. unei operații în care facem b7(LCR)=0, deci fixăm acces la registrii THR, RBR sau IER dar de asemenea scriem și parametrii liniei de comunicație serială
Se cere să aflați:
b1) valoarea înscrisă pt a stabili rata de baud:
Sol: **0012h** = **18** (zecimal)
b2) calculați rata de baud
Sol: se poate folosi relația de mai jos, unde 1.843.200 este frecvența de bază a oscilatorului pt canalul de Tx/Rx
Debit= 1.843.200 / (Divisor*16) = 1.843.200/(**18***16)=6400bps
b3) nr de biți de stop:
Sol: b2=0 => 1 stop bit (din LCR)
b4) lungimea caracterului (în biți):
Sol: b1=**1,** b0=**0** => **7b**/caracter (din LCR)

b5) considerând că avem un caracter recepționat pe linia serială, scrieți o secvență de instrucțiuni care îl aduce în registrul DL. (cu comentarii):
Sol:
in AL, 2E8h ; de la adresa de bază (RBR) luăm caracterul recepționat în reg AL
mov DL, AL ; îl mutăm în reg DL
c) Specificați conținutul în binar dacă pe linia serială se dorește transmiterea unui șir de 3 caractere: „aC5".Desenați cum va arăta semnalul corespunzător punctului c) la transmisia prin pinul TX, în format RS232:
Sol: primul caracter este "a", al doilea este "C" și al treilea este "5"
a="61h", C= "43h", 5= "35h" (on **7 bits each**) =>
1101101010|1010000110|1011000010
"5" "C" "a"
(Tx) ------------------->
Fiecare carcater este "încadrat") de: un singur **bit de START** (întotdeauna bit de 0), și (după cum am aflat) tot un singur **bit de STOP** (întotdeauna bit de 1); în plus, avem **paritate impară** (se aplică doar biților caracterului); aveți grijă că prima dată se transmite pe linie bitul LSb; în Figură, de la stânga spre dreapta, vom avea:
Stop bit , Parity bit: odd parity , Character on 7 bits , Start bit for each character
În plus, prin cablul RS-232, polaritatea semnalului este inversată:



Pb4. Analizați următoarea secvență de program:

```
;Secvență de program care face polling pe TX (pe COM1):
iar:    mov AH,0
        cmp AL, 1Bh    ; testează dacă tasta apăsată e ESC
        JZ gata        ; dacă DA, ieșim afară din buclă
        ; dacă NU, deci e un alt character tastat
        ; îl trimitem pe linia serială
        mov DX, 3F8h
        out DX, AL
JMP iar         ; reluăm bucla, până la apăsarea lui ESC
gata: …
```

Pb5. Analizați următoarea secvență de pseudoinstrucțiuni care se fol. pt a programa apariția unei întreruperi la RX unui caracter (pe COM1):
1) Se va programa MCR <- 0Bh
2) în TVI, se va înlocui vectorul corespunzător î hard 0Ch cu adresa unei noi rutine utilizator, de exemplu "întrerupereRx"
3) în IER se setează b0=1 pt ca să fixăm apariția întreruperii de pe COM1 la Rx unui caracter
4) în registrul de măști PIC(reg. IMR), ne asigurăm că nu e mască pe întreruperea noastră (linia IRQ4), nici pe cele mai prioritare (dacă e nevoie de ele, de ex. Timer, tastatură, COM2) dar și mascăm întreruperile mai puțin prioritare (precum HDD, port paralel), pt că în caz că ar apărea vreo astfel de întrerupere (de la HDD, de ex.) desi e mai puțin prioritară, poate apărea când noi încă nu am recepționat character pe COM1 și va fi tratată (și e posibil să dureze mult, deci ne-ar cam încurca/ încetini în programul nostru)
Deci masca poate fi : 1010 0000b
Inainte de a umbla la IMR, obligatoriu să facem IF=0, iar după ce am terminat cu IMR, să punem înapoi IF=1 ca să poată fi luate în considerare întreruperile (toate întreruperile de la PIC vin pe linia INTR, deci sunt sensibile la IF); dacă ar fi venit pe NMI, nu ar fi contat IF sau masca din IMR
5) în programul principal inserăm o buclă "infinită" în care să stăm și să așteptăm apariția caracterelor în bufferul nostru de Rx : aici am putea avea un polling pe Tx, de exemplu, sau pur și simplu să nu facem nimic, doar să așteptăm … DAR, în timp ce noi așteptăm, vin caractere în bufferul de RX și atunci se va intra în rutina noastră "întrerupereRx"- vezi pct 7)
6) înainte de ieșirea din program, să remascăm IRQ4
7) procedura noastră de tratare a întreruperii, denumită "întrerupereRx", trebuie scrisă (în gen, la sfârșitul programului principal), înainte de
code ends;
end start
această procedură poate fi (obligatoriu se va declara)

```
întrerupereRx PROC FAR
; fiind întreruperea coresp. Recepției unui character, vom lua caracterul din bufferul de Rx
mov DX, 3F8h
in AL, DX
; ce facem apoi cu el? Eventual, îl putem afișa, sau îl putem depune într-o zonă din memorie
...
; când e gata, se face și EOI, adică anunțarea PIC-ului că s-a terminat RTI curente, prin (20h)<-20h
IRET
întrerupereRx ENDP
```

IMR:

			IRQ4	*IRQ3*	*IRQ2*	***IRQ1***	***IRQ0***
LPT	*FD*	*HDD*	*COM1* *COM3*	*COM2* ***COM4***		*keyb*	*Ch 0 timer*
x	*x*	*x*	*x*	*0*	*~~0~~ 1*	*1*	*1*

1. Ce conector avem la PC pentru portul paralel si ce conector pt portul serial ??
2. Cand doresc sa scriu o aplicatie pe paralel vs serial, care sunt registrele ce trebuie considerate la paralel(SPP) si serial??
3. Dati adrese concrete pentru adresele de baza folosite la port paralel /port serial ...
4. Exista o modalitate sa vad daca s-a conectat ceva (un dispozitiv) la portul serial sau paralel ??
5. La paralel in memoria RAM la adresa 0:408 hexa ce pune acolo BIOS-ul pe durata POST? .. exista ceva echivalent la portul serial ?
6. Servicii Bios avem pt portul paralel (test imprimanta, printare caracter, etc -> int 17h, serv 00h,01h,02h), exista si pt serial ??
7. Portul serial are legatura cu PIC-UL (8259) ???
8. Pe ce linie vine intrerupere de la serial,care va fi vectorul de intrerupere asociat ??

Intrebari:

1. La ce se referă afirmația „interfața UART serializează datele care trebuie transmise” ?
R: adică realizează conversia din format paralel în format serial pentru caracterele de date primite de la CPU (fiecare octet scris în UART este trimis bit-cu-bit pe linia serială, cu LSb mai întâi);
2. Dintre biții caracterului, care se transmite primul pe linia serială ?
R: LSb
3. La ce se referă afirmația „interfața UART deserializează datele recepționate” ?
R: adică realizează conversia din serie în paralel a caracterelor de date primite de la un dispozitiv periferic sau modem (după primirea tuturor biților unui caracter, aceștia sunt asamblați într-un singur octet, oferit apoi CPU pt citire);
4. (A/F) Este rolul interfeței seriale UART de a adăuga sau elimina informațiile de încadrare (bit de start, bit/biți de stop, (opțional) bit de paritate).
R: A
5. Cum sunt gestionați biții caracterului de către interfața UART?
R: Biții de date sunt încadrați înainte de transmiterea lor pe linia serială și sunt extrași din cadru după recepția lor de pe linia serială;
6. (A/F) Interfața UART implementează sistemul de întrerupere al procesorului și dialogul de control al modemului conform standardului RS232;
R: A
7. (A/F) Interfața UART externă folosește semnale bipolare și polaritate inversă atât pt semnalele de date seriale, cât și pt semnalele de control modem).
R: A
8. (A/F) Interfața UART stochează temporar atât biții de date primiți, cât și cei pe care trebuie să îi transmită (folosind buffere).
R: A
9. (A/F) Standardul de interfață serială specifică atât semnalele pt transmisia și recepția datelor, cât și semnalele de control al fluxului dintre DTE - Data Terminal Equipment și DCE - Data Communication Equipment.
R: A
10. Care a fost scopul inițial al interfeței UART?
R: scopul inițial al interfeței a fost interconectarea unui computer (DTE) și a unui modem (DCE). Modemul (MODulator-DEModulator) permite unui PC să trimită date pe o linie telefonică analogică, realizând conexiuni la distanțe mari prin PSTN (Public Switched Telephone Network). Numerele de telefon au jucat rolul adreselor IP de astăzi.
11. (A/F) în cadrul Standardului RS232 sunt definite specificațiile electrice și mecanice ale portului serial.
R: A
12. (A/F) Standardul RS232 utilizează o logică negativă (sau inversată) la trecerea informațiilor prin cablul interfeței seriale.
R: A
13. (A/F) Standardul RS232 codează un bit de date 0 ca un nivel de tensiune pozitiv, între +3 și +25 V;
R: A
14. (A/F) Standardul RS232 codează un bit de date 1, („Mark” sau semnalul de control al modemului inactiv) ca ca un nivel de tensiune negativ, între -3 și -25 V;
15. (A/F) Semnalul „Space” se mai numește „starea activă a unui semnal de interfață serială”
R: A
16. (A/F) Semnalul Mark” se mai numește „semnalul de control al modemului inactiv”.
R: A
17. Pe linia de transmisie, un caracter RS-232 constă dintr-un bit de start („Mark”, cu valoare logică 0), urmat de (minim)___, (maxim)____biți de date (trimiși cu LSb mai întâi pe linie), un bit de paritate opțional (paritatea se aplică numai biților caracterului) și (minim)___, (maxim)____biți de stop („Space”, cu valoare logică 1),
R: 5, 8, 1, 2
18.La ce se referă notația / acronimul 8E2?
R: caracterul e codificat cu 8 biti, are paritate pară (even) si se fol 2 biti de stop
19. La ce se refera un „cadru de biti”?
R:. Fiecare unitate de date, de la bitul de start până la bitul/ biții de stop, definește un „cadru” de biti.
20. Când CPU execută o instrucțiune de forma out $AdrPortRegTx_{UART}$, AL, octetul din registrul AL al CPU se trimite în mod ____(paralel/serial) pe BD al sistemului și se înscrie într-un registru intern al UARTului, care îl trimite în format (serial/ paralel) pe linie;
R: paralel, serial
21. când CPU execută o instrucțiune de forma in AL, $AdrPortRegRx_{UART}$, octetul din registrul intern al UARTului va ajunge în format ___(paralel/serial), prin BD, în reg. AL
R: paralel
22. (A/F) La recepție, UARTul extrage datele utile din cadrul sosit, verificând că bitul de paritate recalculat la recepție este egal cu cel transmis.
R: A
23. (A/F) Pentru comunicația serială, nu este neapărat necesar ca cele două terminale (transmițător și receptor) să fie configurate cu aceleași valori ale parametrilor de transmisie/ recepție înainte de transmisia/ recepția efectivă a caracterelor, se poate și după.
R: F, obligatoriu înainte
24. (A/F) Formatul caracterului specifică: numărul de biți de date, numărul de biți de stop, dacă se folosește sau nu paritate (și tipul parității)

R: A
25. (A/F) Viteza de transmisie/recepție (biți pe secundă) a datelor specifica viteza cu care vor circula biții pe linia serială.
R: A
26. (A/F) În sistemele mai vechi, se foloseau 2 biţi de STOP pentru a acorda dispozitivului timp suficient pentru a se pregăti de recepţia următorului caracter, dar în sistemele mai noi se foloseşte 1 bit de STOP.
27. La paritatea pară, nr de biţi de 1 ai caracterului împreună cu bitul de paritate, trebuie să fie un număr PAR;
R: A
28. La paritatea impară, nr de biţi de 1 ai caracterului împreună cu bitul de paritate, trebuie să fie un număr IMPAR;
R: A
29. Există și posibilitatea ca bitul de paritate să fie blocat în 0 sau 1, indiferent de paritatea efectivă a caracterului.
R: A
30. (A/F) În cazul comunicației asincrone, sincronizarea la nivel de bit este asigurată numai pe durata transmisiei /recepției efective a fiecărui caracter. O asemenea comunicație este orientată pe caractere individuale și are dezavantajul că necesită informații suplimentare în proporție de cel puțin 25% pentru identificarea fiecărui caracter.
R: A
31. Atunci când receptorul detectează începutul unui caracter indicat prin bitul de START, pornește un oscilator de ceas local, care permite eșantionarea corectă a biților individuali ai caracterului.
R: A
32. Eșantionarea biților se realizează aproximativ la mijlocul intervalului corespunzător fiecărui bit.
R: A
33. Standardul original în IBM-PC a definit conectorul DB-25 P (tată) la DTE și conectorul de tip S (mamă) la DCE, cu 25 de pini, aceștia fiind înlocuiți ulterior de conectorii mai ieftini DB-9
R: A
34. în protocolul de tip HANDSHAKING, programul activează semnalul DTR și așteaptă răspunsul de la modem,
reprezentat de activarea semnalului _____ (pentru cuplarea la linia telefonică).
R: DSR
35. în protocolul de tip HANDSHAKING, programul activează semnalul RTS și așteaptă răspunsul de la modem,
reprezentat de activarea semnalului _____(pentru începerea transmisiei efective).
R: CTS

36.(A/F) Regiștrii cipurilor UART sunt accesibili prin instrucțiuni I/ O (de intrare/ieșire) la _____ adrese de port diferite
R: 8
37. LA cipurile UART exista si regiștri care nu sunt accesibili prin software, precum TSR și RSR.
R: A
38. (A/F) ROM BIOS-ul computerelor IBM PC originale a permis utilizarea a două porturi seriale, denumite COM1 și COM2, dar
Ulterior, numărul de porturi a fost extins cu încă alte două porturi, denumite COM3 și COM4.
R: A
39. (A/F) Pentru primul adaptor serial (sau port) COM1, regiștrii portului pot fi accesați la adrese cuprinse între 3F8h și 3FFh
R: A
40. (A/F) Pentru al doilea adaptor serial (sau port) COM2, regiștrii portului pot fi accesați la adrese cuprinse între 2F8h și 2FFh
R: A
41. (A/F) Când există mai mult de două porturi seriale în computer, acestea vor partaja unele nivele de întrerupere.
42. în PC_XT, PC-AT, câte nivele de întrerupere sunt partajate de porturile seriale ?
R: 2 – sunt 2 linii de întrerupere dedicate porturilor seriale : IRQ3 și IRQ4
43. (A/F) În sistemele PC-AT, portul COM3 partajează întreruperea de nivel 4 (IRQ4) cu portul COM1,
R: A
44. (A/F) În sistemele PC-AT, portul COM4 partajează întreruperea de nivel 3 (IRQ3) cu portul COM2.
R: A
45. (A/F) De la S.O. Windows 95, numărul de porturi seriale a fost extins la 128.
R: A
46. (A/F) Primele 2 adrese ale unui port COM permit accesul la mai mult de un registru al cipului.
R: A
47. Daca se stie ca b7 din registrul LCR al UART este 1 și că anterior s-a executat instrucțiunea mov DX, 3F9h, care este registrul accesat la execuția următoarei instrucțiuni?
in AL, DX ; ?

R: DL High
48. Daca se stie ca b7 din registrul LCR al UART este 0 și că anterior s-a executat instrucțiunea mov DX, 3F9h, care este registrul accesat la execuția următoarei instrucțiuni?
in AL, DX ; ?
R: IER
49. Daca se stie ca b7 din registrul LCR al UART este 1 și că anterior s-a executat instrucțiunea mov DX, 3F9h, care este registrul accesat la execuția următoarei instrucțiuni?
out DX, AL ; ?
R: DL High
50. Daca se stie ca b7 din registrul LCR al UART este 0 și că anterior s-a executat instrucțiunea mov DX, 3F9h, care este registrul accesat la execuția următoarei instrucțiuni?
out DX, AL ; ?
R: IER
51. Daca se stie ca b7 din registrul LCR al UART este 1 și că anterior s-a executat instrucțiunea mov DX, 3F8h, care este registrul accesat la execuția următoarei instrucțiuni?
in AL, DX ; ?
R: DL Low
52. Daca se stie ca b7 din registrul LCR al UART este 0 și că anterior s-a executat instrucțiunea mov DX, 3F8h, care este registrul accesat la execuția următoarei instrucțiuni?
in AL, DX ; ?
R: RBR
53. Daca se stie ca b7 din registrul LCR al UART este 1 și că anterior s-a executat instrucțiunea mov DX, 3F8h, care este registrul accesat la execuția următoarei instrucțiuni?
out DX, AL ; ?
R: DL Low
54. Daca se stie ca b7 din registrul LCR al UART este 0 și că anterior s-a executat instrucțiunea mov DX, 3F8h, care este registrul accesat la execuția următoarei instrucțiuni?
out DX, AL ; ?
R: THR
55. Care este relatia Pentru calcularea valorii divizorului?

R: Divizor = 1,843,200 / (RataBinară * 16)

56. (A/F) viteza maximă de transmisie este 1.8432MHz/16 adică 115200bps
R: A
57. Caracterul care trebuie trimis trebuie înscris în reg. ____
R: THR
58. Caracterul care trebuie receptionat trebuie luat din reg. ____
R: RBR
59. cipul UART poate genera ____ tipuri de solicitări de întrerupere, cu nivele de prioritate diferite:
R: 4
60. Care sunt cele 4 tipuri de întreruperi posibil a fi generate de UART ?
R: întrerupere la recepția unui caracter (când RBR este plin), întrerupere la transmisia unui caracter (când THR este gol), întrerupere la schimbarea stării liniei de comunicatie (atunci când LSR se schimbă) și întrerupere la apariția unei erori pe linia de comunicație (atunci când MSR se schimbă).
61. (A/F) Registrul IER permite activarea sau dezactivarea individuală a acestor întreruperi (prin scrierea unei valori de 0 sau 1 pe bitul corespunzător din IER).
R: A
62. (A/F) Registrul IER permite operație de citire/ scriere, astfel încât setările curente pot fi interogate în orice moment (de exemplu, dacă dorim să mascăm o anumită întrerupere fără a le afecta pe celelalte).
R: A
63. Care este ordinea priorității celor 4 surse de întrerupere la UART?
R: (0110-cea mai prioritară) LSI>DAI>THREI>MSI (0000-cea mai puțin prioritară).
64. Cum se poate seta o intrerupere la UART?
R: se seteaza prin reg IER, daca punem bit în 0 – resetata întreruperea resp., daca punem bit in 1 – setata intreruperea respectiva:

3	2	1	0
Modem Status Modificat	Line Status Modificat	THR gol	Dată disponibilă la recepție

65. Cum se poate reseta o intrerupere la UART?
R: se reseteaza prin citirea THR sau RBR, LSR sau MSR, in functie de intreruperea aparuta
66. cum se poate seta o intrerupere din soft, fara sa afecteze celelalte posibile intreruperi ?
R: se citeste reg IER si se modifica doar bitul asupra carui vrem sa setam o noua intrerupere, inscriind valoarea modificata in IER

!!! a se vedea si exemplele date in lucrare