# 11. Cuplarea la un port serial


**Cuprins**

**Cuprins Laborator 11**

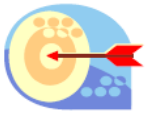
11.1.Portul serial
11.2.Desfăşurarea lucrării
11.3.Exemple de programe

**Cunoştinţe preliminare**

Limbaj de asamblare MCS 51

**Introducere**

Această lucrare familiarizează studenţii cu portul serial. Este descris portul integrat în microcontroller, modurile în care lucrează registrele de control şi programarea. În cadrul lucrării se simulează portul serial, apoi se programează pentru a îndeplini diferite funcţii. Programul final testează portul prin transmisia unui caracter, recepţia lui şi compararea cu caracterul trimis. Două exemple de programe ajută studenţii să poată scrie propriul program.



**Obiective**

După parcurgerea acestui modul studenţii vor înţelege în detaliu funcţionarea şi programarea portului serial. Ei vor putea să:

- Programeze registrele de control pentru a obţine diverse moduri de lucru;
- Programeze diferite viteze de transfer;
- Să simuleze portul;
- Să programeze un transfer de date cu parametri ceruţi de o anumită aplicaţie.

**Durata medie de studiu individual**

Durata medie de studiu individual este de 2 ore.

### 11.1. Portul serial

Portul serial este de tip full duplex, permițând transmisia şi recepția simultană a datelor. Receptorul este dublu bufferat, ceea ce permite începerea recepționării unui octet înainte ca precedentul să fi fost citit din registrul de recepție. Totuşi, dacă citirea primului octet nu se efectuează în timpul recepției celui de-al doilea octet, unul dintre octeți se va pierde. Portul serial cuprinde structuri logice distincte pentru transmisie şi recepție, conectate la magistrala internă. Controlul funcționării celor două părți ale portului se realizează prin intermediul registrului S0CON (Serial 0 CONtrol register), care permite stabilirea modurilor de operare.

Ambele părți, de transmisie şi de recepție, posedă registre distincte pentru date: S0BUF-T, respectiv S0BUF-R, dar care sunt văzute de către utilizator ca un singur registru – S0BUF (Serial 0 BUFfer register). Separarea comenzilor se realizează prin registre controlate prin program. Astfel, la o comandă "Write to S0BUF" data este înscrisă în buffer-ul de transmisie şi se comandă transferul serial pe linia TXD. La recepția unui caracter pe linia RXD, acesta se încarcă într-un registru de deplasare şi apoi, printr-o comandă "Load S0BUF", în bufferul de recepție, din care poate fi citită printr-o comandă "Read S0BUF". În funcție de modul de operare, linia RXD poate fi utilizată şi ca linie de ieşire (în Modul 0), iar viteza de transfer poate fi controlată prin program, după cum frecvența de tact se obține de la Timerul 1 sau de la oscilator (fosc/2). Interfața serială poate genera o cerere de întrerupere către CPU, într-o logică SAU, fie la recepția, fie la transmisia completă a unui caracter.

Modurile de operare ale portului serial sunt în esență de două tipuri:

a) registru de deplasare (mod 0);

b) comunicație serială asincronă (UART) cu 8 biți (mod 1) sau cu 9 biți (mod 2 şi 3).

Aceste moduri pot fi programate prin intermediul registrului S0CON. De asemenea, prin S0CON se pot controla întreruperile portului, atât la transmisie (TI) cât şi la recepție (Rl), precum şi cel de-al 9-lea bit (TB8, RB8) la funcționarea UART cu 9 biți.

*Modul 0* asigură o funcționare de tip registru de deplasare, având intrarea şi ieşirea datelor pe linia RxD/P3.0, iar tactul de deplasare primit/generat pe linia TxD/P3.1. Rata de comunicație în acest mod este fixă şi are valoarea fosc/12, fiind de 1Mbps la fosc=12MHz.

Transmisia este inițiată de orice instrucțiune care foloseşte ca registru destinație S0BUF. De asemenea, blocul de control al transmisiei forțează în "1" logic cea de-a 9-a poziție a registrului de deplasare. Transmisia începe cu bitul cel mai puțin semnificativ al octetului de date şi se face în ritmul de 1 bit/ciclu. Odată cu deplasarea biților la ieşire, în ritmul

fixat prin TxD, registrul se completează cu zerouri. Când şi cel mai semnificativ bit al cuvântului a fost transmis, cel de-al 9-lea bit, care a fost setat pe "1", ajunge în poziția cea mai puțin semnificativă a registrului de deplasare şi blocul de control dezactivează transmisia. Totodată se realizează o ultimă deplasare şi se setează fanionul de întrerupere la transmisie - TI din S0CON, în al 10-lea ciclu maşină.

Recepția este inițiată printr-o comandă cu REN=1 în S0CON, precedată de resetarea bitului Rl. Blocul de control al recepției înscrie biții 11111110b în registrul de deplasare. Inițierea recepției are loc la detectarea bitului de start al unui caracter. La fiecare bit recepționat pe linia RxD, în acest registru are loc o deplasare la stânga cu o poziție, tot în ritmul de 1 bit/ciclu stabilit prin TxD. Atunci când bitul "0" ajunge în poziția din extrema stângă a registrului de deplasare, blocul de control comandă încă o deplasare şi încărcarea registrului S0BUF cu valoarea recepționată, în cel de-al 10-lea ciclu maşină de la înscrierea comenzii în S0CON recepția este oprită şi se setează fanionul de întrerupere RI.

**S0CON**

| **(msb)** | | | | | | | **(lsb)** |
|---|---|---|---|---|---|---|---|
| SMO | SM1 | SM2 | REN | TB8 | RB8 | TI | Rl |

| SMO | SM1 | Modul | Descriere | Viteza serială |
|---|---|---|---|---|
| 0 | 0 | 0 | registru cu deplasare | fosc /12 |
| 0 | 1 | 1 | UART - 8 biți | variabilă |
| 1 | 0 | 2 | UART - 9 biți | fosc/32 sau fosc/64 |
| 1 | 1 | 3 | UART -9 biți | variabilă |

SM2 - activează comunicația multiprocesor, în modurile 2 şi 3.
REN - activează recepția serie.
TB8 - cel de-al 9-lea bit transmis în modurile 2 şi 3.
RB8 - cel de-al 9-lea bit recepționat în modurile 2 şi 3.
TI - fanion de întrerupere la transmisie. Este setat HW în funcție de modul de operare. Trebuie resetat SW.
RI - fanion de întrerupere la recepție. Este setat HW, în funcție de modul de operare.Trebuie resetat SW.

Modurile 1, 2 şi 3 corespund unei funcționări de tip UART ("Start-Stop"), cu 8 biți (Modul 1) sau cu 9 biți (Modurile 2 şi 3).

***Modul 1*** permite o funcționare de tip asincron (UART) cu un bit de START ("0" logic), 8 biți de date şi un bit de STOP ("1" logic). Viteza de comunicație este variabilă (110 Baud ÷ 62,5 KBaud) şi este controlată de Timerul 1. De obicei, Timerul 1 se foloseşte în modul

2, cu auto-încărcare, caz în care viteza de comunicație se calculează cu relația:

$$V_{[Baud]} = \frac{2^{SMOD}}{32} \times \frac{f_{OSC}}{12 \cdot [256 - (TH1)]},$$

în care SMOD este bitul cel mai semnificativ al registrului PCON, iar (TH1) este valoarea de reîncărcare, memorată de TH1.

Transmisia în modul 1 este inițiată de o înscriere în registrul S0BUF, care determină şi o poziționare pe "1" logic al celui de-al 9-lea bit (TB8) din registrul cu deplasare, însă începerea efectivă a transmisiei este declanşată de următoarea depăşire a unui contor divizor prin 16.

Recepția în modul 1 este activată de o tranziție 1→0 pe linia RxD dacă REN=1 în S0CON. Pentru sesizarea tranziției, linia RxD este eşantionată de 16 ori într-o perioadă corespunzătoare vitezei de comunicație. Pentru fiecare bit de date recepționat, blocul de control comandă eşantionarea sistematică a liniei RxD de 3 ori succesiv, în cea de a 7-a, a 8-a şi a 9-a stare din cele 16 ale numărătorului de eşantionare. Este acceptată valoarea găsită pe linia RxD în ultimele 2 eşantionări din cele 3. Această procedură asigură rejecția perturbațiilor.

Dacă valoarea acceptată la eşantionarea primului bit pe linia RxD nu este "0" bitul de START), logica de recepție se resetează şi se aşteaptă o nouă tranziție 1→ 0. Procedura menționată asigură rejecția unor biți de start falşi. Dacă bitul de start este corect, acesta este transmis la intrarea registrului de deplasare şi apoi sunt transmişi şi restul biților (data şi bitul de STOP). Când bitul de START ajunge în poziția extremă-stânga a registrului cu deplasare, blocul de control al recepției determină încă o deplasare la stânga, încărcarea registrului S0BUF cu octetul recepționat, setarea fanionului RI şi încărcarea bitului de STOP în RB8. Încheierea recepției este realizată aşa cum s-a arătat mai sus, dacă şi numai dacă sunt îndeplinite condițiile:

1) Rl = 0
2) SM2 = 0 sau STOP BIT = 1.

***Modul 2*** corespunde funcționării de tip UART cu 11 biți: un bit de START ("0"), 8 biți de date (lsb este primul), un al 9-lea bit de date programabil şi un bit de STOP ("1"). La transmisie, cel de-al 9-lea bit de date (TB8) poate fi forțat fie pe "0", fie pe "1". La recepție, cel de-al 9-lea bit de date ajunge în RB8 din S0CON.

Viteza de comunicație în acest mod este selectabilă, fie fosc/32 sau fosc/64. Astfel, la fosc=12MHz viteza de comunicație maximă poate fi de 375 KBauds şi cea minimă de 187,5 KBauds.

Ca şi în modul 1, transmisia este inițiată de orice instrucțiune care utilizează S0BUF ca

registru destinație. Un semnal "Scrie în S0BUF" realizează şi încărcarea bitului TB8 din S0CON în cea de a 9-a poziție a registrului de deplasare. Pe linia TxD se transmite bitul de START şi apoi cei 8 biți de date, prin deplasare la dreapta. Când bitul TB8 este la ieşire, bitul de STOP ajunge chiar în poziția din registru a lui TB8. Celelalte poziții sunt completate cu zerouri, în această situație, blocul de control al transmisiei mai inițiază o ultimă deplasare, după care se dezactivează transmisia şi se setează TI, în cea de-a 11-a stare a contorului divizor prin 16 de după semnalul "Scrie în S0BUF".

Recepția este inițiată la detectarea unei tranziții 1→0 pe linia RxD, dacă REN=1 în S0CON. Ca şi în modul 1, linia RxD este eşantionată cu o frecvență de 16 ori mai mare decât frecvența corespunzătoare vitezei de comunicație. La detectarea unei tranziții, contorul divizor prin 16 este resetat şi în registrul cu deplasare se înscrie 1FFh. Apoi începe recepția serială a celor 9 biți de date, de la dreapta la stânga, cu verificarea validității de 3 ori la fiecare bit. Când bitul de START ajunge la extrema stângă a registrului de deplasare, acesta semnalizează blocului de control al recepției să execute o ultimă deplasare, încărcarea S0BUF şi RB8, şi setarea RI. Semnalele de încărcare a S0BUF şi RB8 şi de setare a fanionului RI vor fi generate numai dacă pe durata ultimei deplasări sunt îndeplinite simultan condițiile:

1) RI = 0
2) SM2 = 0 sau cel de-al 9-lea bit recepționat este "1".

După un tact, indiferent dacă condițiile mai sus menționate sunt sau nu îndeplinite, blocul de control al recepției reîncepe detectarea unei tranziții 1→0 pe linia RxD.

***Modul 3*** este identic cu modul 2, cu deosebirea că viteza de comunicație poate fi variabilă şi stabilită de Timerul 1, la fel ca în modul 1.

### 11.2. Desfăşurarea lucrării

După scrirea programului, se poate face o simulare a portului UART cu ajutorul programului ProView lansând *View → Hardware → UART*, după care va apare o fereastră precum cea din figura 11.1., în care este vizibil buffer-ul portului serial şi un câmp de intrare în care pot fi introduse caractere, care mai apoi sunt trimise în buffer. Se poate face o simulare pas cu pas pentru a se prelua valorile introduse (simularea nu este în timp real).

Se recomandă schimbarea valorii timerului folosit pentru portul serial. Fereastra pentru vizualizarea timerului se lansează tot din cadrul meniului *View*, şi se modifică valoarea TH(L)X cât mai aproape de valoarea maximă (0xFF).

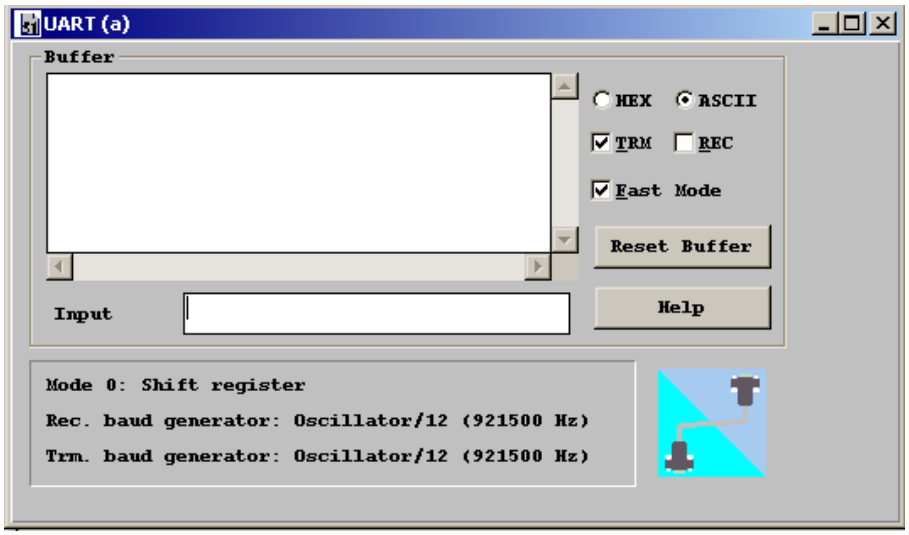



Figura 10.1. Fereastră simulare UART în ProView

Se cere studenților să realizeze un program care trimite un caracter pe serială, recepționează caracterul trimis şi verifică dacă recepția a fost corectă. Pentru aceasta se unește firul RXD cu TXD în exteriorul plăcii.

### 11.3.Exemple de programe

**1.Să se realizeze un program care să facă transmisia unui caracter de la placa de dezvoltare IMC 500 catre PC.**

```
IEN0    EQU  0A8h          ; registru pentru validarea intreruperilor
S0CON   EQU  098h          ; registrul de control al portului serial
S0BUF   EQU  099h          ; registrul tampon al portului serial

 CSEG AT 8000h         ; punct de intrare la resetare
 LJMP    main          ; salt peste zona vectorilor de intrerupere


 CSEG AT 8090H
main:
        mov     R0,#10h        ; cate caractere se vor transmite
        mov     R1,#'a'
        callinit_seriala       ; apelare rutina de initializare a serialei
        mov     S0BUF,#0Ah     ; line feed
        jnb TI,$               ; se asteapta transmiterea caracterului
        clr  TI                ; se curata indicatorul de transmisie
```

```
loop:       mov     S0BUF,R1      ; se transmite pe serial caracterul
            jnb TI,$
            clr  TI
            inc R1
            djnz        R0,loop
            ljmp        0000h         ; iesire din program

init_seriala:
            mov         PCON,#00h
            mov         IEN0,#00000000b     ; invalidarea tuturor intreruperilor
            mov         TMOD,#00100000b   ; Timerul 1 generator de tact in modul 2
            mov         TCON,#00h
            mov         TH1,#0FDh           ; viteza de comunicare este 9600 Baud
            mov         TL1,#0FDh           ; initializarea registrului de reincarcare a
        Timerului 1
            setb        TR1                 ; pornirea portului serial
            mov         S0CON,#01000000b  ; programare port serial in modul 1
            ret
end
```

**2. Să se realizeze un program care să recepționeze caractere (scrise la tastatura PC-ului) şi apoi, pe măsură ce le recepționează să le transmită înapoi PC-ului.**

```
IEN0        EQU   0A8h   ; registru pentru validarea intreruperilor
S0CON       EQU   098h    ; registrul de control al portului serial
S0BUF       EQU   099h    ; registrul tampon al portului serial


ORG          8000h          ; adresa de la care se poate introduce cod
LJMP        main           ; salt peste zona vectorilor de intrerupere

ORG         8023h          ; adresa folosita de placa IMC500 pentru intreruperea
seriala
LJMP        int_seriala

CSEG AT 8090H
main:
            nop
            mov         PCON,#00h
            mov         IEN0,#10010000b     ; validarea intreruperii seriale
            mov         TMOD,#00100000b   ; Timerul 1 generator de tact in modul 2
            mov         TH1,#0FDh     ; viteza de comunicare este 9600 Baud
            mov         TL1,#0FDh      ; initializarea registrului de reincarcare a Timerului
```

```
    1
        setb    TR1             ; prnirea portului serial
        mov     S0CON,#01010000b; programare port serial in modul 1
        mov     S0BUF,#'>'      ; prompter
        sjmp    $               ; bucla infinita

int_seriala:
        jb   RI,receptie
        jnb TI,$            ; se asteapta transmiterea caracterului
        jb   TI,end_int
receptie:
        clr  RI
        mov     R7,S0BUF
        mov     A,R7
        cjne    A,#13,NewLine       ; comparatie cu new line
        mov     S0BUF,#0ah          ; se trimite Line Feed
        jnb     TI,$                ; se asteapta transmiterea caracterului

NewLine:
        mov     S0BUF,R7    ; line feed
        jnb TI,$                    ; se asteapta transmiterea caracterului
        end_int:
        clr   TI
        reti

end
```

Trebuie ca studenții să țină minte toate instrucțiunile microcontrollerului MCS 51?

Nu, o listă cu instrucțiunile poate fi folosită când se scriu programele, inclusiv la test.

**Rezumat**

Această lucrare are ca scop înțelegerea funcționării şi programării portului serial integrat în microcontrollerele familiei MCS 51. O descriere detaliată a portului urmată de descrierea registrelor importante şi a modurilor de funcționare sunt necesare pentru înțelegerea programării portului. O simulare a funcționării este urmată de două exemple de programare.

**Bibliografie**

1. C. Gerigan, P. Ogrutan, *Tehnici de interfațare*, Editura Universitatii Transilvania Brasov, 2000, pag. 201-229 online la: http://vega.unitbv.ro/~ogrutan/ti/cap10.pdf
2. http://www.fsinc.com/devtools/index.htm
3. http://www.nxp.com/documents/data_sheet/80C552_83C552.pdf
4. http://www.keil.com/dd/docs/datashts/intel/ism51.pdf