SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1726

KOMUNIKACIJA SERIJSKOM VEZOM

Marko Turk

Zagreb, lipanj 2011.

SADRŽAJ

1. Uvod

Serijska komunikacija je jedan način komunikacije između računala. RS-232 standard definira serijsku vezu između PC računala. Najčešće se pojavljuje DE-9 konektor koji ima devet pinova. Serijski sklop sadrži UART čip koji je moguće programirati i tako podesiti parametre serijske veze. Spajanjem računala null-modem kablom moguće je ostvariti direktni komunikacijski kanal.

Praktična implementacija serijske komunikacije ostvarit će se na Operacijskom sustavu za ugradbena računala (OSzUR). OSzUR već ima implementirane odgovarajuće pogonske programe koji omogućuju pristup sklopovlju serije. Napravit će se jednostavan poslužitelj i klijent koji međusobno komuniciraju preko serijske veze.

2. Serijska veza

Serijska veza je proces slanja podataka preko komunikacijskog kanala slijedno bit po bit. Postoji više standarda serijskog prijenosa podataka kao što su FireWire, I2C, SATA, USB i RS-232.

2.1 RS-232

Standard RS-232 veže se uz serijske priključke na PC računalu. Standard podržava prijenos korisničkih podataka sinkrono i asinkrono. Sinkroni se prijenos kontrolira pomoću signala takta.

Zbog odvojenosti sklopovlja za slanje i primanje, podržan je *full-duplex* način rada što znači da se u isto vrijeme podaci mogu slati i primati.

Kod RS-232 standarda postoje dva tipa uređaja: *Data Terminal Equipment* (DTE) i *Data Communication Equipment* (DCE). Tip uređaja određuje koje će se žice koristiti za slanje, a koje za primanje podataka. Tablica 1. prikazuje uobičajene signale koji se koriste za serijski prijenos[1].

Tablica 1. RS-232 signali

Oznaka	Naziv	Izvor
DTR	Data Terminal Ready	DTE
DCD	Data Carrier Detect	DCE
DSR	Data Set Ready	DCE
RI	Ring Indicator	DCE
RTS	Request To Send	DTE
CTS	Clear To Send	DCE
TxD		DTE
RxD		DCE
GND	Masa	Zajednička
PG	Zaštitna masa	Zajednička

Imena signala iz tablice 1. su iz stajališta DTE-a. Svaki signal ima svoju uobičajenu upotrebu. Najčešće se koristi sljedeće značenje za pojedine signale:

DTR

Označava DCE-u prisutnost DTE-a.

DCD

Aktivan ako je DCE spojen na telefonsku liniju.

DSR

DCE je spreman i može primati podatke ili naredbe.

RI

DCE je primio dolazni *ring signal* na telefonskoj liniji.

RTS

DTE traži od DCE-a da se pripremi na primanje podataka.

CTS

DCE označava da je spreman na primanje podataka.

TxD

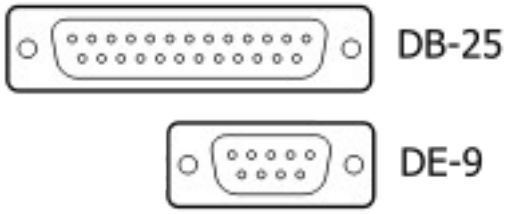
Prenosi podatke s DTE na DCE.

RxD

Prenosi podatke s DCE na DTE.

2.1.1. RS-232 konektori

Postoji više vrsta konektora koji se mogu koristiti za serijski prijenos. RS-232 definira korištenje DB-25 konektora, ali češće se koristi DE-9 konektor. Konektori su prikazani na slici 2.1. [2]



Slika 2.1. Konektori

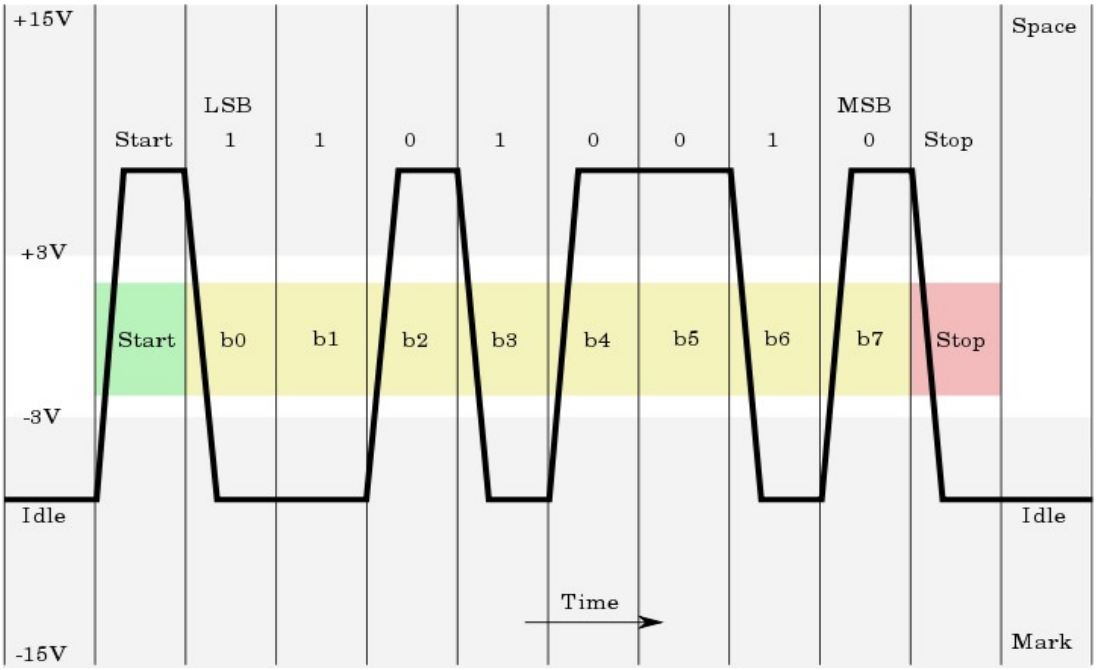
Pinovi konektora spajaju se na RS-232 signale prema tablici 2.

Tablica 2. Spajanje konektora

Oznaka	DB-25 pin	DE-9 pin
DTR	20	4
DCD	8	1
DSR	6	6
RI	22	9
RTS	4	7
CTS	5	8
TxD	2	3
RxD	3	2
GND	7	5
PG	1	-

Standard definira raspone napona za logičku jedinicu i nulu. Za logičku nulu uzimaju se naponi između +3V i +15V. Za logičku jedinicu naponi moraju biti između -3V i -15V. Područje između -3V i +3V je zabranjeno. Na slici 2.2. prikazan je primjer serijske RS-232 komunikacije.

Najmanje značajan bit se kod serijske komunikacije šalje prvi, a najviše značajan bit zadnji. Osim korisne informacije, šalju se "start bit" i "stop bit" koji označavaju početak i kraj poruke. Šalje se i paritetni bit za provjeru valjanosti poruke [3].



Slika 2.2. Primjer serijske komunikacije (Wikipedia, 2011.)

2.2. UART

UART je skraćeno od *Universal asynchronous receiver/transmitter* - univerzalni asinkroni prijemnik/odašiljač. Koristi se zajedno sa RS-232. UART pretvara podatke između paralelnih i serijskih formata. Može se konfigurirati brzina prijenosa i format slanja podataka. Za pravilan rad, UART na strani primatelja i na strani odašiljača mora biti podešen na iste parametre veze.

Na PC računalima UART je mapiran na fiksnu adresu. U tablici 3. prikazane su adrese na koje su mapirani UART sklopovi pojedinih serijskih priključaka.

Tablica 3. Adrese UART za serijske priključke

Serijski priključak	Adresa	IRQ
COM 1	0x3F8	4
COM 2	0x2F8	3
COM 3	0x3E8	4
COM 4	0x2E8	3

2.2.1. UART registri

UART čip ima 12 registara koji su mapirani na 8 različitih U/I (ulazno/izlaznih) lokacija. Kako ima više registara nego lokacija, postoji bit DLAB (*Divisor Latch Access Bit*). DLAB-om se preko iste memorijske lokacije može pristupiti različitom registru.

Registri UART sklopa navedeni su u tablici 4. Adresa označava relativan pomak u odnosu na osnovnu adresu gdje je serijski priključak mapiran. Oznaka "x" u tablici označava da DLAB nema utjecaja na odabir registra na toj memorijskoj lokaciji [4].

Tablica 4. UART registri

Adresa	DLAB	U/I operacija	Oznaka registra	Ime registra
+0	0	Pisanje	THR	Transmitter Holding Buffer
+0	0	Čitanje	RBR	Receiver Buffer
+0	1	Čitanje/Pisanje	DLL	Divisor Latch Low Byte
+1	0	Čitanje/Pisanje	IER	Interrupt Enable Register
+1	1	Čitanje/Pisanje	DLH	Divisor Latch High Byte
+2	x	Čitanje	IIR	Interrupt Identification Register
+2	x	Pisanje	FCR	FIFO Control Register
+3	x	Čitanje/Pisanje	LCR	Line Control Register
+4	x	Čitanje/Pisanje	MCR	Modem Control Register
+5	x	Čitanje	LSR	Line Status Register
+6	x	Čitanje	MSR	Modem Status Register
+7	x	Čitanje/Pisanje	SR	Scratch Register

THR/RBR

U UARTU postoje međuspremnici za primanje i slanje podataka. U THR (meduspremnik za slanje) stavlja se podatak koji će biti poslan preko serije. Podatak koji se primi preko serije sprema se u međuspremnik za primanje (RBR).

DLL i DLH

Divisor latch okteti određuju brzinu prijenosa preko serije. Vrijednost koja se postavi na te oktete utječe na brojač koji se pokrene nakon svakog poslanog podatka. Nakon što brojač dođe do nule, šalje se novi podatak i ponovno se puni brojač.

Vrijednost djelitelja računa se po formuli (1).

$$DivisorLatchValue = \frac{115200}{BaudRate} \qquad (1)$$

Vrijednosti koje je potrebno postaviti u ove registre navedene su u tablici 4. Prikazane su vrijednosti samo za češće korištene brzine prijenosa.

Tablica 5. Vrijednosti DLL i DLH za različite brzine prijenosa

Brzina prijenosa (bitova po sekundi)	**Djelitelj**	**DLH**	**DLL**
50	2304	0x09	0x00
110	1047	0x04	0x17
220	524	0x02	0x0C
300	384	0x01	0x80
600	192	0x00	0xC0
1200	96	0x00	0x60
2400	48	0x00	0x30
4800	24	0x00	0x$18
9600	12	0x00	0x0C
19200	6	0x00	0x06
38400	3	0x00	0x03
57600	2	0x00	0x02
115200	1	0x00	0x01

IER

IER je registar za kontrolu prekida serijske veze. Određuje kada će UART generirati prekid. Prekid se može generirati kada UART primi podatak sa serije i kad je međuspremnik za slanje prazan (nakon čega se može upisati i poslati novi podatak).

IIR

Registar se koristi samo za čitanje i ima dvije namjene. Koristi se za identifikaciju prekida i za identifikaciju samog UART čipa.

Nakon što UART generira prekid, čitanjem ovog registra može se saznati tip prekida.

FCR

Određuje se ponašanje FIFO (*First In-First Out*) spremnika. Moguće je uključiti i isključiti FIFO međuspremnik. FIFO međuspremnik se koristi kod primanja i slanja za čuvanje više podataka. Omogućava primanje više znakova sa serije i procesor ih može čitati prema poretku kojem su stigli. Isto tako, procesor može više znakova staviti u međuspremnik i UART ih redom šalje.

LCR

LCR se koristi za postavljanje DLAB bita (za pristup različitim registrima na istoj adresi) i za postavljanje vrste protokola za serijsku vezu.

Može se podesiti paritet, broj stop bitova i broj znakova koji čine riječ. Riječ je najmanji broj bitova koji se odjednom šalju preko serije.

MCR

Omogućava direktnu manipulaciju nekim žicama na UART-u čime se može kontrolirati modem koji je spojen na seriju.

LSR

Čitanjem registra saznaju se informacije o stanju UART-a. Može se saznati da li je došlo do prekida, jesu li prazni registri za slanje i primanje i da li je možda došlo do nekih grešaka kod komunikacije.

MSR

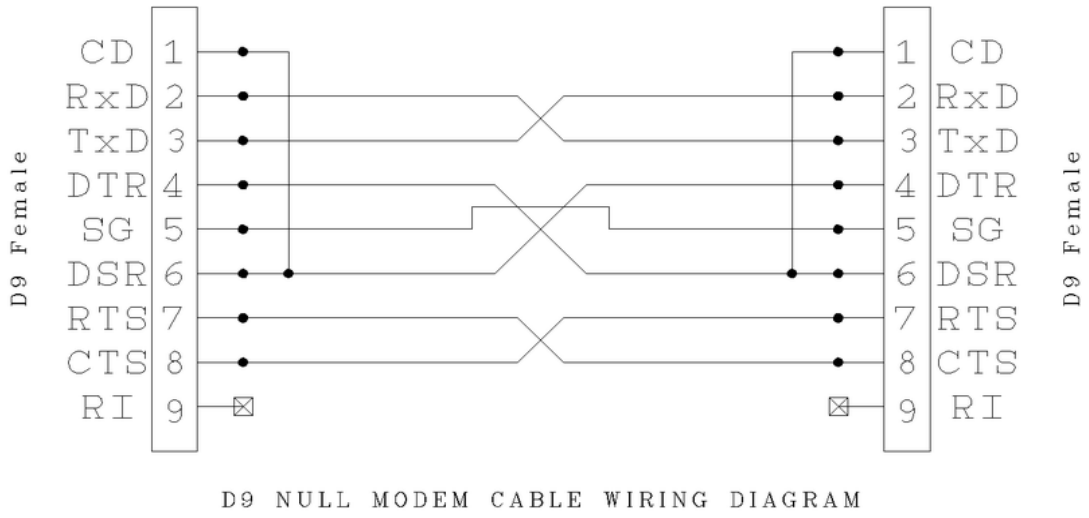
Registar sadrži informacije o stanju modema spojenog na seriju.

SR

Ovaj se registar ne koristi kod serijske komunikacije.

2.3. Null-modem kabel

Null-modem kabel služi za direktno spajanje dvaju računala serijskom vezom. Za uspješnu serijsku vezu nisu potrebni dodatni vanjski uređaji. Način spajanja žica za DE-9 konektor prikazan je na slici 2.3.



Slika 2.3. Primjer spajanja null-modem kabla (Wikipedia, 2011.)

Za ostvarivanje serijske komunikacije implementirane u ovom radu potrebno je računala spojiti ovakvim null-modem kablom.

3. Echo poslužitelj

Echo poslužitelj je primjer jednostavnog mrežnog programa. Poslužitelj čeka da se klijent spoji i pošalje proizvoljan tekst preko komunikacijskog kanala. Poslužitelj ne radi nikakvu obradu primljenih podataka nego taj isti tekst samo pošalje natrag klijentu.

Tekst koji će klijent poslati poslužitelju učita se sa tipkovnice. Tekst se čita dok korisnik ne pritisne enter. Nakon što klijent pošalje učitani tekst poslužitelju, čeka na odgovor i ispisuje primljeni tekst na ekran. Klijent nakon toga završava s radom, a poslužitelj čeka konekciju od novog klijenta.

Poslužitelj implementiran u ovom radu pretpostavlja da su samo dva računala spojena serijom. Nije podržan konkurentan rad što znači da se može posluživati samo jedan klijent u isto vrijeme.

Pomoću ovakve jednostavne komunikacije moguće je testirati pravilan rad sustava i komunikacije između računala.

4. Sučelje za rad na seriji

OSzUR pruža funkcije za inicijalizaciju serijskog sklopa. Prije korištenja serije potrebno je otvoriti uređaj pomoću *k_device_open* funkcije. Funkcija vraća opisnik koji se koristi u ostalim funkcijama za pristup seriji. Inicijalizacija se obavlja kod podizanja sustava (datoteka *startup.c*) gdje se otvara serijski uređaj i postavlja na eksterni opisnik serije *u_serial*.

Za implementaciju serijske komunikacije korištene su funkcije iz *sys* sloja sustava. U *sys* sloju definirani su sistemski pozivi koji koriste i pozivaju jezgrine funkcije. Korištena je funkcija *sys__device_recv* koja sa serije dohvaća znakove i *sys__device_send* koja znakove šalje na seriju. Kod korištenja ovih sistemskih poziva potrebno je funkciji predati opisnik serije.

5. Implementacija poslužitelja i klijenta

5.1. Funkcije za rad na seriji

Za potrebe poslužitelja i klijenta koji rade na seriji napisane su funkcije koje omogućuju lagano slanje i primanje podataka. Funkcije su smještene u datoteku programs/api/serial.c.

Funkcija *serial_send* šalje na seriju proizvoljne podatke. Funkcija prima kao argument pokazivač na podatke i veličinu u oktetima.

Funkcija *serial_recv* sa serije prima proizvoljne podatke. Podaci se primaju dok se ne učita zadan broj okteta.

Pomoću *serial_recv_string sa* serije se čitaju znakovi dok se ne naiđe na null-znak. Pomoću ove funkcije moguće je pročitati null-terminiran niz znakova.

Funkcijom *serial_send_string* na seriju se šalje null-terminirani niz znakova.

Navedene funkcije u konačnici pozivaju dvije pomoćne funkcije, *serial_recv_char* i *serial_send_char*, koje šalju i primaju jedan znak sa serije. Sve funkcije blokiraju dok ne pošalju ili prime cijelu poruku. Za proizvoljne podatke to je određeno veličinom koja se funkciji predaje kao parametar. Za funkcije koje primaju niz znakova, funkcija blokira dok se ne primi zadnji znak iz niza.

Pomoćna funkcija koja prima jedan znak sa serije (*serial_recv_char*) stalno proziva seriju i pokušava pročitati znak.

```
for ( ; sys__device_recv((void *)&c, 1, 0, u_serial) <= 0; delay(&t))
      ;
```

Nakon što sistemski poziv uspije pročitati znak, petlja se prekida i učitani znak se vraća programu. Ako sistemski poziv nije učitao znak, radi se vremenska odgoda prije ponovnog poziva da bi se smanjilo zauzeće procesora.

5.2. Poslužitelj i klijent

Pomoću funkcija koje šalju i primaju podatke sa serije napravljen je poslužitelj i klijent. Program je smješten u datoteci programs/serial/serial.c.

Nakon što se pokrene program, korisnik mora odabrati način rada. Ako se s tipkovnice učita slovo "s", pokreće se poslužiteljski način rada, a slovo "c" pokreće klijent.

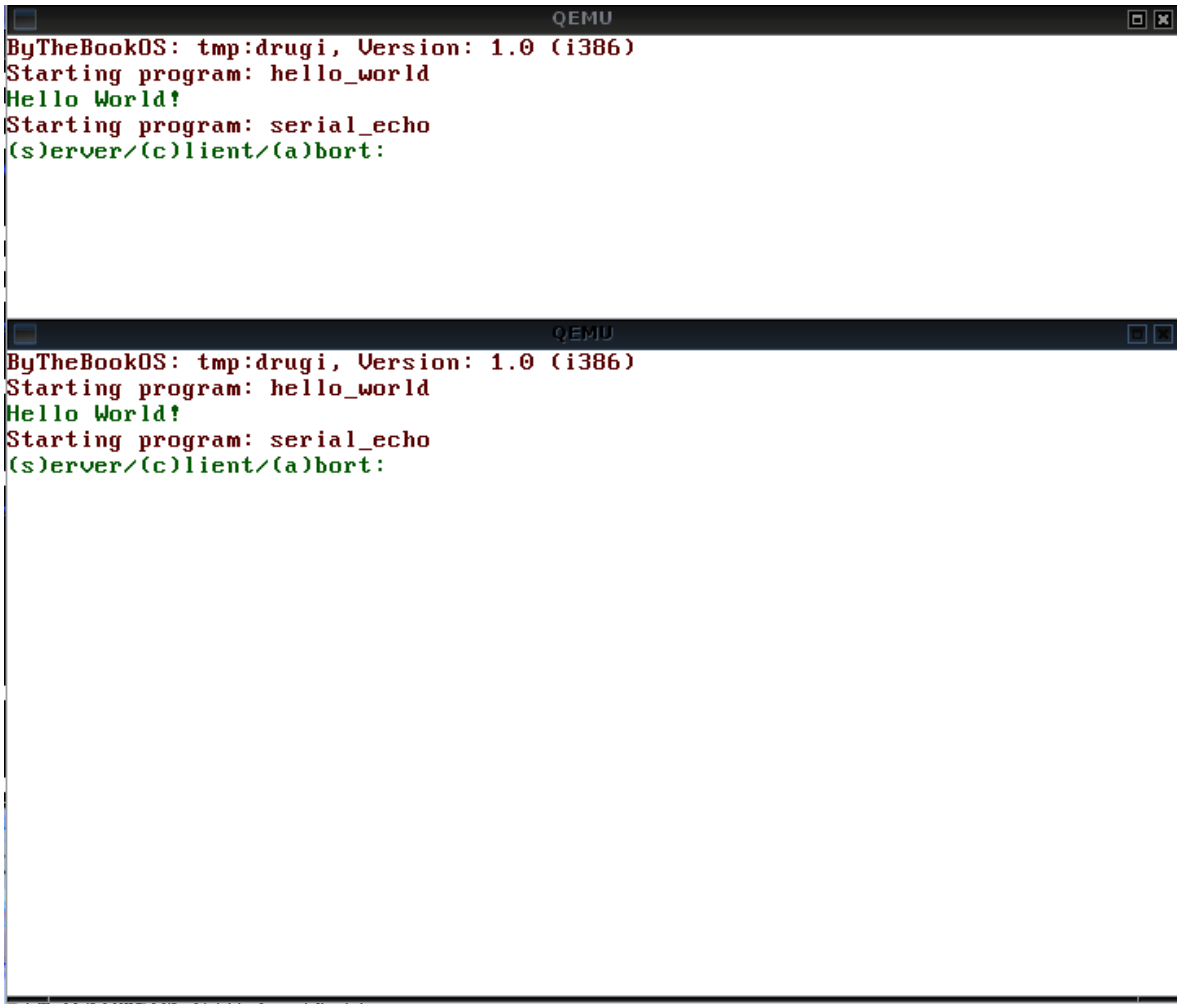
Poslužitelj pomoću funkcije *serial_recv_string* učitava u međuspremnik niz znakova sa serije. Nakon što primi niz znakova, ispisuje na ekran kontrolnu poruku u kojoj obavještava da je dobio zahtjev od klijenta. Pomoću funkcije *serial_send_string* šalje primljeni niz znakova natrag klijentu.

Klijent prvo učita s tipkovnice poruku. Pomoću *serial_send_string* poruka se pošalje i čeka se odgovor s funkcijom *serial_recv_string.* Primljena poruka ispisuje se na ekran.

6. Testiranje i primjer komunikacije

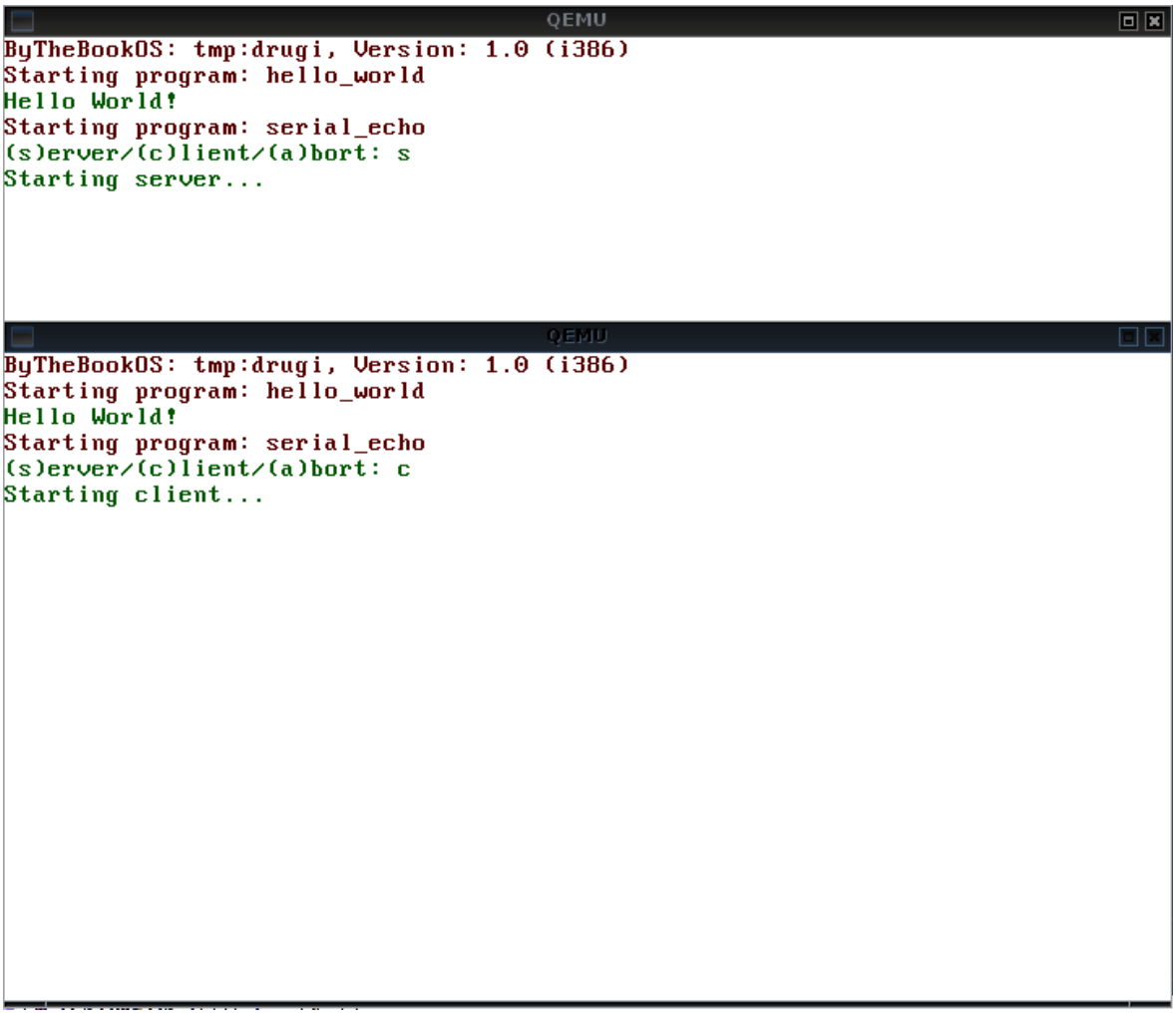
Program je testiran na QEMU emulatoru. QEMU može emulirati PC računalo s serijskim priključkom. Moguće je spojiti dva QEMU emulatora preko serijske veze i na taj način testirati implementiranu serijsku komunikaciju.

Nakon prevođenja, stvori se CD-ROM slika koja se učita u QEMU emulator. Pokrenu se dva primjerka sustava i emulatori se podese da se spoje preko serijske veze. Na slici 6.1. prikazano je stanje sustava nakon pokretanja.



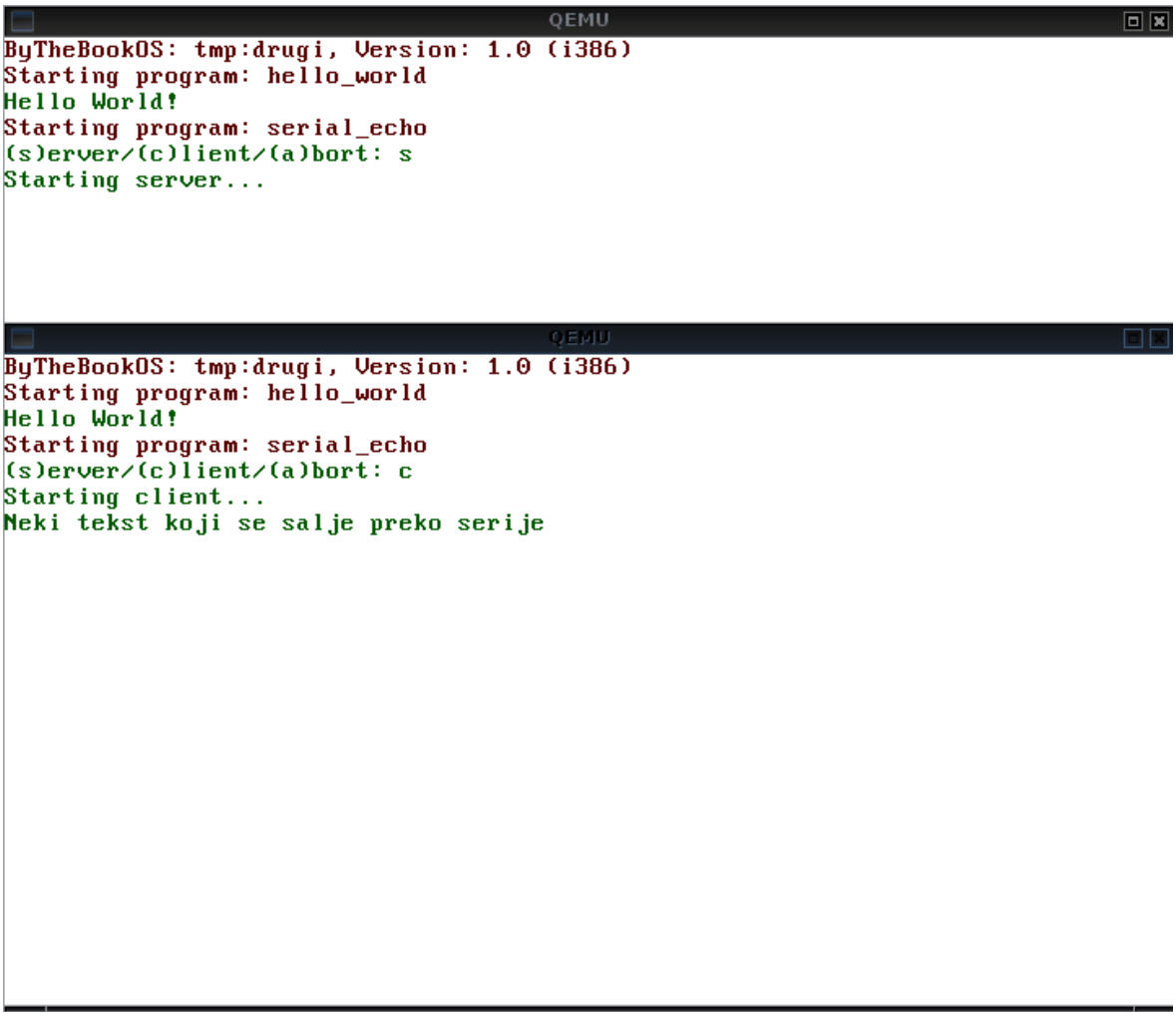
Slika 6.1. Sustav nakon pokretanja

Nakon pokretanja, u jednom sustavu odabere se način rada poslužitelja, a u drugom način rada klijent. Poslužitelj čeka na znakove sa serije, a klijent čita sa tipkovnice unos korisnika (slika 6.2.).



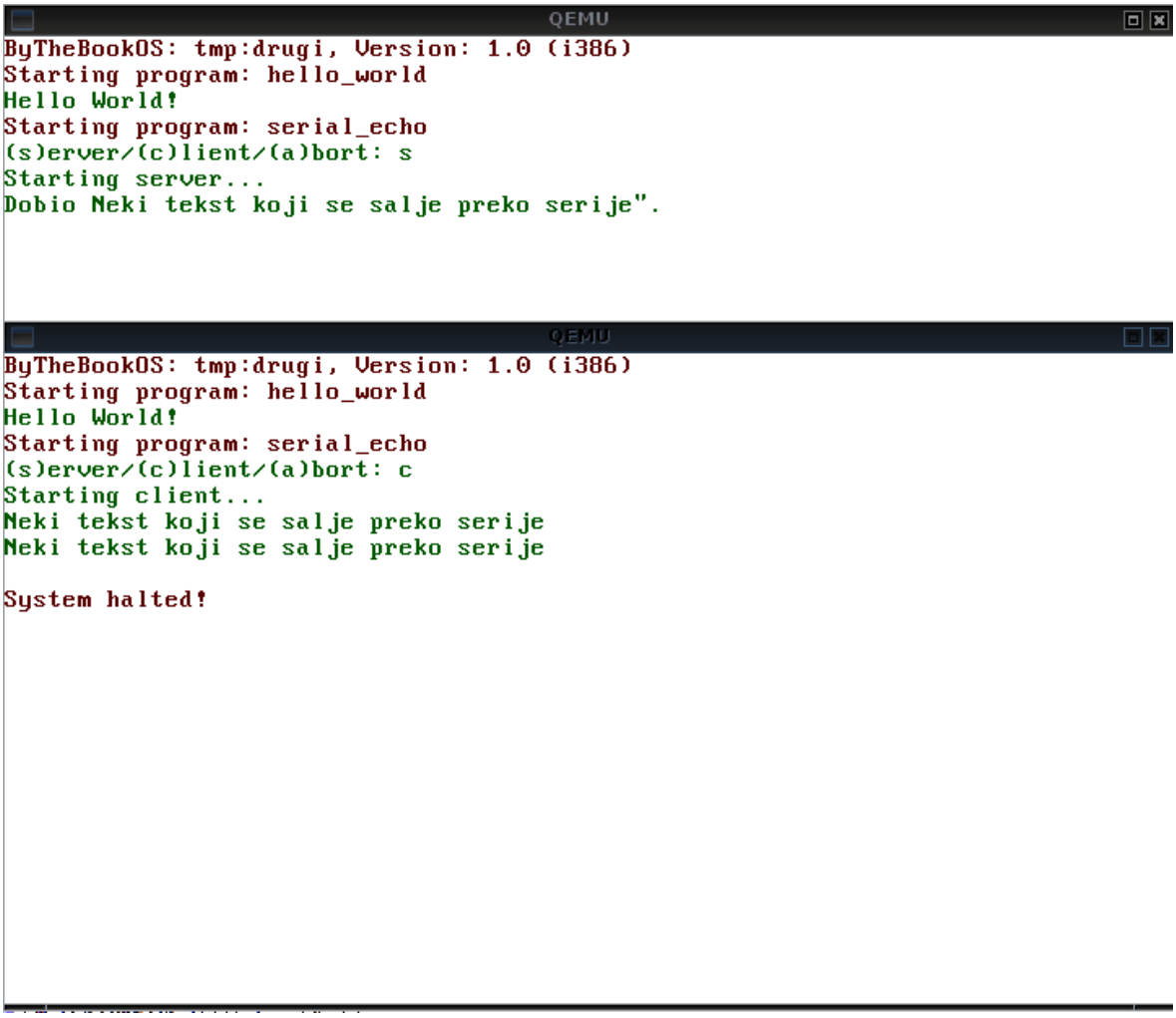
Slika 6.2. Odabir načina rada sustava

U sustavu gdje se odabrao način rada klijent upiše se proizvoljan tekst. Nakon pritiska na tipku enter, tekst se pošalje preko serije. Primjer stanja sustava prije slanja podataka prikazan je na slici 6.3.



Slika 6.3. Unos teksta koji će se poslati preko serije

Tekst poslan preko serije dođe do poslužitelja koji učita cijeli niz znakova. Nakon što je primio null-znak, poslužitelj vraća isti tekst opet preko serije klijentu. Klijent primi znakove i ispiše ih na ekran. Ako je poslan i primljen tekst identičan, serijska komunikacija je ispravno implementirana i sustavi su međusobno uspješno komunicirali.



Slika 6.4. Stanje sustava nakon slanja i primanja podataka

Klijent završava s radom nakon što primi odgovor od poslužitelja. Poslužitelj ostaje aktivan i čeka na novi zahtjev klijenta. Završava s radom dok od klijenta primi tekst "Kraj".

7. Zaključak

U radu je objašnjena i implementirana serijska komunikacija između računala. Opisan je način rada serije prema RS-232 standardu. Objašnjeni su svi sklopovi i registri koji sudjeluju u serijskoj komunikaciji na računalima. Objašnjeno je kako se serijski sklop programira i koristi.

Kao praktični dio ovog rada, napravljena je implementacija jednostavnog poslužitelja i klijenta koji komuniciraju preko serije. Na postojećem operacijskom sustavu za ugradbena računala implementirane su funkcije za jednostavno slanje i primanje podataka sa serije. Pomoću tih funkcija napravljeni su echo poslužitelj i klijent koji međusobno komuniciraju preko serije. Testiranje se radilo na QEMU emulatoru.

8. Literatura

[1] RS-232, *RS-232 standard,* http://en.wikipedia.org/wiki/RS-232, 20.5.2011.

[2] Serial port, http://en.wikipedia.org/wiki/Serial_port, 18.5.2011.

[3] Jelenković , L., Operacijski sustavi za ugrađena računala , Skripta za predavanje , Fakultet elektrotehnike i računarstva, Zagreb, 2011.

[4] Serial Programming, *8250 UART Programming,* http://en.wikibooks.org/wiki/Serial_Programming:8250_UART_Programming, 18.5.2011.

[5] Interfacing the Serial / RS232 Port, http://www.beyondlogic.org/serial/serial.htm, 20.5.2011.

[6] Null modem, http://en.wikipedia.org/wiki/Null_modem, 20.5.2011.

Sažetak

Komunikacija serijskom vezom

RS-232 standard opisuje serijsku komunikaciju između PC računala. Podaci se na seriji šalju slijedno jedan iza drugoga. U radu je napravljen primjer serijske komunikacije između PC računala. Na OSzUR sustavu implementirane su metode za jednostavno slanje i primanje preko serije i s tim funkcijama izgrađen je primjer jednostavnog poslužitelja i klijenta.

Ključne riječi: serijska veza, RS-232, UART, OSzUR, echo poslužitelj, QEMU

Summary

Serial communication

RS-232 standard defines serial communication between PCs. Serial communication is the process of sending data one bit at a time. Functions that provide simple way of sending data through serial port were implemented on OSzUR system. Server and client that exchange data through serial channel were made using this functions.

Key words: serial communication, RS-232, UART, OSzUR, echo server, QEMU