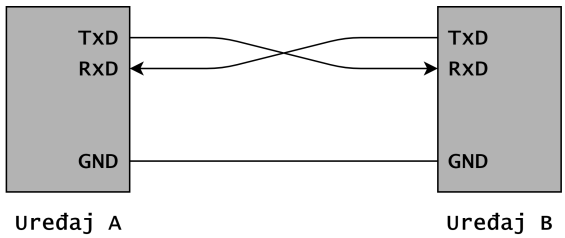
Poglavlje 7

Serijski port i funkcije za serijsku komunikaciju

7.1 RS-232 komunikacioni protokol

Rad serijskog porta kod personalnih računara i kod većine savremenih mikrokontrolera zasnovan je na standardizovanom RS-232 protokolu. Hardverska jedinica unutar mikrokontrolera koja je zadužena za ovakav tip komunikacije obično se naziva UART (eng. *Universal Asynchronous Receiver-Transmitter*).

Po RS-232 standardu, komunikacija se obavlja posredstvom linije za slanje podataka (TxD), linije za prijem (RxD), a postoje i dodatni signali za sinhronizaciju (eng. *handshaking*). UART jednice obično ne koriste sinhronizacione signale, nego samo linije koje su apsolutno neophodne za ostvarivanje veze: TxD, RxD i GND (masa). Minimalna konfiguracija za RS-232 komunikaciju između dva uređaja prikazana je na slici 7.1.

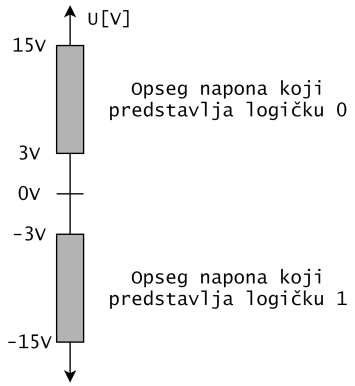


Slika 7.1: Minimalna konfiguracija za RS-232 komunikaciju između dva uređaja

7.2 Naponski nivoi

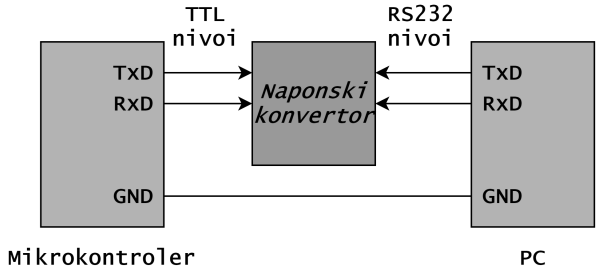
Naponski nivoi signala po RS-232 standardu su:

- +3V do +15V za logičku nulu
- –3V do –15V za logičku jedinicu



Slika 7.2: RS-232 naponski nivoi

UART jedinica radi na naponskim nivoima koji odgovaraju naponu napajanja mikrokontrolera, na primer TTL nivoima: logička "1" = V_{cc} = 5V, logička "0" = 0V. Stoga, pri povezivanju uređaja koji koriste različite naponske nivoe, na primer mikrokontrolera i personalnog računara, potrebno je koristiti kolo za prilagođenje naponskih nivoa. Jedno od popularnijih kola takve namene je *MAX232*. Ovo je ilustrovana na slici 7.3.



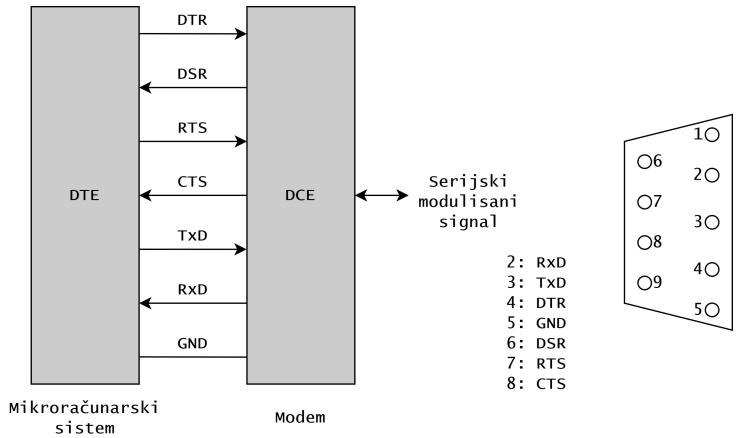
Slika 7.3: Konverzija naponskih nivoa

U novije vreme, pogotovo kod prenosnih računara, čest je slučaj da računar ne poseduje specijalizovani serijski port. U takvim slučajevima slučaju, uobičajeno je da se na USB port poveže integrisano kolo koje vrši konverziju između USB i RS-232 protokola (na primer FTDI ili CH34x).

7.3 Signalizacija

RS-232 standard uveden je u vreme kada su terminali povezivani sa centralnim računarom preko modema, pa su osnovni termini u standardu prilagođeni takvoj primeni. Računari se u RS-232 terminologiji nazivaju DTE (eng. *Data Terminal Equipment*), a modemi DCE (eng. *Data Communication Equipment*). DTE i DCE uređaj povezuju se putem sledećih signala:

- DTR (*Data Terminal Ready*) – terminal javlja modemu da je spreman za serijski prenos podataka (izlazni signal terminala);
- DSR (*Data Set Ready*) – modem javlja terminalu da je spreman za serijski prenos podataka (ulazni signal terminala);
- RTS (*Request To Send*) – izlazni signal koji terminal prevodi u aktivno stanje ako ima podatke koje želi da prenese preko serijske RS-232 veze (izlazni signal terminala);
- CTS (*Clear To Send*) – signal kojim DCE javlja terminalu da je spreman za serijski prenos podataka (ulazni signal terminala);
- TxD (*Transmitted Data*) – serijski izlazni signal za podatke (izlazni signal terminala);
- RxD (*Received Data*) – serijski ulazni signal za podatke (ulazni signal terminala);
- GND (*Ground*) – zajednička masa.

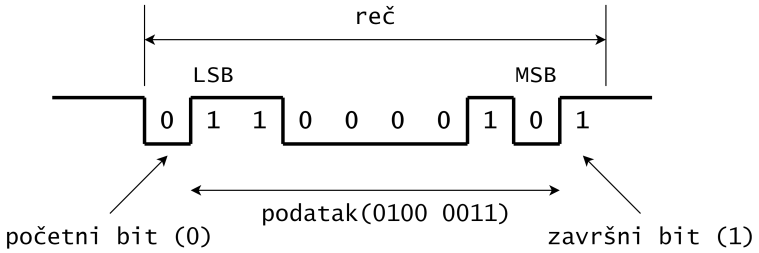


Slika 7.4: RS 232 signali (levo) i njihov raspored na standardnom DB-9 konektoru

Serijski port, koji koristi RS-232 protokol, slanje podataka vrši asinhrono, reč po reč (eng. *character*). Kada je linija neaktivna, na njoj je visok naponski nivo,

tj. pre početka serijskog prenosa i između prenosa dve uzastopne reči, linija za serijski prenos drži se u stanju logičke "1". Opadajuća ivica na komunikacionoj liniji signalizira prijemnoj strani da predajna strana počinje sa slanjem reči. Reč koja se šalje sadrži:

1. početni (*start*) bit, koji je uvek logička "0";
2. 5, 6, 7 ili 8 bita podatka koji se prenosi; prenos se vrši od najnižeg (LSB) ka najvišem (MSB) bitu;
3. opcioni bit za proveru pariteta; ovo predstavlja najjednostavniji način kontrole ispravnosti primljene reči; bit pariteta se postavlja na vrednost "1" ukoliko je broj jedinica u okviru podatka (stavka 2) paran, ili neparan, što se može konfigurisati; na prijemnoj strani se tada proverava koliko ima jedinica među bitima primljenog podatka i proverava se da li se to slaže sa bitom pariteta koji je primljen; na ovaj način moguće je detektovati nastanak greške na jednom bitu, ali nije moguće detektovati koji bit je pogrešno primljen, niti prepoznati grešku u slučaju pogrešnog prijema parnog broja bita;
4. 1 do 2 završna (*stop*) bita, koji su uvek logičke "1".



Slika 7.5: Format podatka prilikom RS-232 prenosa

Da bi prijemnik i predajnik uspešno komunicirali, potrebno je konfigurisati i prijemnik i predajnik na identičan način. Ova konfiguracija minimalno podrazumeva definisanje:

- brzine serijske komunikacije (eng. *Baud rate*), izražene brojem bita u sekundi; tipične brzine RS-232 komunikacije su *9600b/s*, *19200b/s*, *38400b/s*, *57600b/s* i *115200b/s*;
- broja bita podataka (5, 6, 7 ili 8);
- da li se koristi bit pariteta i ako se koristi, da li će on biti setovan za paran ili neparan broj jedinica u okviru bita podataka;
- broj stop bita koji se koristi (1 ili 2).

Nakon što su prijemnik i predajnik konfigurisani, prijemnik poruke prolazi kroz sledeća stanja čekajući na poruku:

1. Čeka se početak poruke pojavom *start* bita, tj. opadajuće ivice na prijemnoj RxD liniji.
2. Nakon detekcije početka poruke, prima jedan po jedan bit poruke tako što očitava napon na RxD liniji svakih $kT + T/2, k = \{0, 1, ..., 8\}, T = 1/(baud\ rate)$. Da bi ovo bilo moguće, neophodno je da prijemnik i predajnik imaju usklađenu brzinu komunikacije, jer je slanje asinhrono (prijemnik nema nikakvu indikaciju kada je na liniji dostupan "novi" bit).
3. Nakon prijema cele poruke, proverava da li je bit pariteta u skladu sa bitovima podataka, u slučaju da je konfigurisano korišćenje bita pariteta. Ukoliko jeste, podatak je prihvaćen, u suprotnom primljeni podatak se odbacuje i neophodno ga je ponovo primiti.

Paralelno sa prijemom poruke preko RxD linije, moguće je istovremeno slati poruku putem nezavisne komunikacione linije TxD. To znači da UART ostvaruje dvosmernu (eng. *full duplex*) komunikaciju.

7.4 USART0 mikrokontrolera ATmega328P

Mikrokontroler ATmega328P poseduje interfejs za serijsku RS-232 komunikaciju označen sa USART0 (*Universal Synchronous and Asynchronous Serial Receiver and Transmitter 0*). Njegove najvažnije karakteristike su:

- dvosmerna *Full Duplex* komunikacija;
- asinhroni ili sinhroni način rada;
- u slučaju sinhronog načina rada, može da generiše takt (*master* režim) ili da bude taktovan od strane drugog uređaja (*slave* režim);
- *baud rate* generator visoke rezolucije;
- podržava formate reči sa 5, 6, 7, 8 ili 9 bita i 1 ili 2 *stop* bita;
- hardverski podržano generisanje i provera bita pariteta;
- detekcija grešaka prilikom komunikacije;
- filtriranje smetnji;
- tri različita izvora prekida: završetak slanja (*TX complete*), pražnjenje registra za slanje, ili završetak prijema (*RX complete*);
- multiprocesorski režim komunikacije;

- asinhroni režim komunikacije dvostruke brzine.

U nastavku će biti dat pregled registara koji pripadaju modulu USART0, sa neophodnim podešavanjima.

UDRn - USART I/O DATA REGISTER N

Bit	7	6	5	4	3	2	1	0
	RXB[7:0](Read)							
	TXB[7:0](Write)							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Prijemni i predajni bafer USART0 modula su memorijski mapirani na istoj adresi. U slučaju upisa vrednosti u UDRn, vrednost se prosleđuje u predajni bafer registar (TXB), a prilikom čitanja vrednosti UDRn očitava se prijemni bafer registar (RXB). U slučaju 5-, 6- i 7-bitnog režima, nekorišteni viši biti se ignorišu od strane predajnika, odnosno bivaju popunjeni nulama od strane prijemnika.

U predajni bafer se može upisati vrednost samo kada je bit UDREn u okviru registra UCSRnA setovan. Podatak upisan u UDRn u slučaju kada je UDREn resetovan će biti ignorisan od strane predajnika. Prilikom upisa podatka u predajni bafer, ukoliko je predaja omogućena, predajnik upisuje podatak u predajni pomerački registar onda kada je taj registar prazan i podatak se dalje serijski prosleđuje na pin TxD0 (PD1).

Prijemni bafer je organizovan kao dvobajtni FIFO, koji menja stanje prilikom prijema svakog novog podatka, kao i prilikom čitanja podatka iz njega.

UCSRNA – USART CONTROL AND STATUS REGISTER N A

Bit	7	6	5	4	3	2	1	0
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	1	0	0	0	0	0

- **Bit 7 – RXCn: USART receive complete**
  Ovaj indikator se setuje kada postoji nepročitan potatak u prijemnom baferu i resetuje kada je prijemni bafer prazan (odnosno ne sadrži nepročitan podatak. Može biti iskorišćen za okidanje prekida (eng. *Receive Complete Interrupt*), ukoliko je bit RXCIEn u okviru registra UCSRnB setovan.

- **Bit 6 – TXCn: USART Transmit Complete**
  Ovaj indikator se setuje kada je celokupan sadržaj predajnog pomeračkog registra poslat i ne postoji nov podatak u predajnom baferu (UDRn). Može biti iskorišćen za okidanje prekida (eng. *Transmit Complete Interrupt*), ukoliko je setovan bit TXCIEn u okviru registra UCSRnB. Automatski se resetuje prilikom izvršenja prekidne rutine.

- **Bit 5 – UDREn: USART Data Register Empty**
  Kada je setovan, ovaj indikator signalizira da je predajni bafer (UDRn) spreman za prijem novog podatka. Može biti iskorišćen za okidanje prekida (eng. *Data Register Empty Interrupt*), ukoliko je setovan bit UDRIEn u okviru registra UCSRnB.

- **Bit 4 – FEn: Frame Error**
  Ovaj bit se setuje u slučaju kada je karakter u prijemnom baferu primljen sa greškom u formatu, izazvanom pogrešnom vrednošću stop bita (logička nula). Vrednost je validna sve do očitavanja vrednosti prijemnog bafera (UDRn).

- **Bit 3 – DORn: Data OverRun**
  Ovaj bit se setuje ako je prijemni FIFO pun (dva karaktera su primljena), a pri tome je detektovan novi start bit. Ovaj bit je validan do čitanja podatka iz registra UDRn.

- **Bit 2 – UPEn: USART Parity Error**
  Ovaj bit se setuje ukoliko je došlo do greške prilikom provere pariteta primljenog karaktera. Vrednost je validna sve do očitavanja vrednosti prijemnog bafera (UDRn).

- **Bit 1 – U2Xn: Double the USART Transmission Speed**
  Ovaj bit je validan samo u asinhronom režimu. Njegovim setovanjem smanjuje se faktor preskaliranja takta *baud rate* generatora sa 16 na 8, čime se efektivno udvostručuje brzina prenosa.

- **Bit 0 – MPCMn: Multi-processor Communication Mode**
  Ovaj bit omogućava multiprocesorski režim komunikacije. Kada je setovan, svi dolazni podaci koji ne sadrže informaciju o adresi biće ignorisani.

UCSRNB – USART CONTROL AND STATUS REGISTER N B

Bit	7	6	5	4	3	2	1	0
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

- **Bit 7 – RXCIEn: RX Complete Interrupt Enable n**
  Setovanjem ovog bita dozvoljava se prekid usled setovanja indikatora RXCn registra UCSRnA (eng. *Receive Complete Interrupt*).

- **Bit 6 – TXCIEn: TX Complete Interrupt Enable n**
  Setovanjem ovog bita dozvoljava se prekid usled setovanja indikatora TXCn registra UCSRnA (Transmit Complete Interrupt).

- **Bit 5 – UDRIEn: USART Data Register Empty Interrupt Enable n**
  Setovanjem ovog bita dozvoljava se prekid usled setovanja indikatora UDREn registra UCSRnA (eng. *Data Register Empty Interrupt*).

- **Bit 4 – RXENn: Receiver Enable n**
  Setovanjem ovog bita dozvoljava se rad prijemnika. U tom slučaju, USARTn modul preuzima kontrolu nad RxD0 pinom (PD0).

- **Bit 3 – TXENn: Transmitter Enable n**
  Setovanjem ovog bita dozvoljava se rad predajnika. U tom slučaju, USARTn modul preuzima kontrolu nad TxD0 pinom (PD1).

- **Bit 2 – UCSZn2: Character Size n**
  Bit UCSZn2 u kombinaciji sa bitima UCSZn1:0 registra UCSRnC određuje broj bita podatka koji se prenosi serijski.

- **Bit 1 – RXB8n: Receive Data Bit 8 n**
  RXB8n je deveti bit primljenog podatka u slučaju 9-bitnog prenosa. Mora biti očitan pre čitanja nižih osam bita iz regista UDRn.

- **Bit 0 – TXB8n: Transmit Data Bit 8 n**
  TXB8n je deveti bit podatka za slanje u slučaju 9-bitnog prenosa. Mora biti postavljen pre upisa nižih osam bita u registar UDRn.

UCSRNC – USART CONTROL AND STATUS REGISTER N C

Bit	7	6	5	4	3	2	1	0
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	1	1	0

- **Bits 7:6 – UMSELn1:0 USART Mode Select**
  Ovi biti određuju režim rada USARTn modula u skladu sa tabelom 7.1, datom u nastavku.

Tabela 7.1: Konfiguracija režima rada USART modula

UMSELn1	UMSELn0	Režim
0	0	Asinhroni USART
0	1	Sinhroni USART
1	0	-
1	1	Master SPI

- **Bit 5:4 – UPMn1:0: Parity Mode**
  Ovi biti određuju da li i u kom režimu se koristi bit pariteta. Ako se koristi, predajnik će automatski generisati i poslati bit pariteta nakon bita podatka. Sa druge strane, prijemnik će izračunati bit pariteta za svaki primljeni podataka i uporediti ga sa primljenom vrednošću bita pariteta. U slučaju neslaganja, bit UPEn registra UCSRnA će biti setovan. Veza između režima rada modula u pogledu korišćenja pariteta i odgovarajućih konfiguracionih bita je navedena u tabeli 7.2.

Tabela 7.2: Konfiguracija pariteta

UPMn1	UPMn0	Režim
0	0	Isključen
0	1	-
1	0	Uključen, parni paritet
1	1	Uključen, neparni paritet

- **Bit 3 – USBSn: Stop Bit Select**
  Ovaj bit određuje broj stop bita koji će generisati predajnik prilikom svakog slanja. Kada je setovan, broj stop bita je 2, a kada je resetovan, broj stop bita je 0.

- **Bit 2:1 – UCSZn1:0: Character Size**
  U kombinaciji sa bitom UCSZn2 registra UCSRnB, biti UCSZn1:0 određuju broj bita podatka prilikom serijskog prenosa.

Tabela 7.3: Konfiguracija broja bita podataka

UCSZn2	UCSZn1	UCSZn0	Veličina podataka
0	0	0	5bita
0	0	1	6bita
0	1	0	7bita
0	1	1	8bita
1	0	0	-
1	0	1	-
1	1	0	-
1	1	1	9bita

- **Bit 0 – UCPOLn: Clock Polarity**
  Ovaj bit je od značaja samo u sinhronom režimu, dok se, u suprotnom, u njega upisuje nula. Služi za određivanje ivice takta na koju se vrši promena, odnosno semplovanje stanja linije za prenos podataka.

UBRRNL AND UBRRNH – USART BAUD RATE REGISTERS

Bit	15	14	13	12	11	10	9	8
	-	-	-	-	UBRRn[11:8]			
	UBRRn[7:0]							
	7	6	5	4	3	2	1	0
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0

Za određivanje brzine serijske komunikacije (eng. *baud rate*) koristi se kombinacija registara UBRRnH i UBRRnL. Viša četiri bita registra UBRRnH se ne koriste, a niža četiri u kombinaciji sa registrom UBBRnL čine 12-bitnu vrednost koja određuje *baud rate* i koja se računa po narednoj formuli:

$$UBRRn = \frac{F_{osc}}{16 \cdot BAUD} - 1$$

gde je *BAUD* brzina komunikacije izražena u bitima u sekundi (*bit/s*).

Više detalja moguće je pronaći i u tehničkoj dokumentaciji mikrokontrolera ATmega328P, na narednom linku.

`http://ww1.microchip.com/downloads/en/DeviceDoc/`
`Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf`

U nastavku su objašnjene osnovne funkcije za rad sa USART0 modulom.

7.5 Biblioteka za serijsku komunikaciju

Biblioteka koja sadrži funkcije za rad sa modulom USART0 se nalazi u *GitHub* repozitorijumu ove knjige, na sledećem linku.

`github..repo`

Ovu biblioteku sačinjavaju dve datoteke: *usart.h* i *usart.c*. Kako bi rad sa funkcijama iz ove biblioteke bio moguć, ove datoteke je potrebno uključiti u projektni direktorijum.

Sam princip implementacije biblioteke je sledeći. Pošto je registar UDR0 u stanju da pamti samo dva poslednja karaktera koji su pristigli putem serijskog porta, u slučaju prijema dužeg niza karaktera potrebno je veoma brzo obrađivati karakter po karakter, ili koristiti posebnu memorijsku strukturu u koju se smeštaju svi primljeni podaci. Ovakva struktura naziva se prijemni FIFO bafer (eng. *First-In-First-Out*). U okviru biblioteke kao makro konstanta je definisana i dužina prijemnog bafera od 64 bajta[1]. Za pravovremeno smeštanje novopristiglih podataka u prijemni bafer zadužena je prekidna rutina serijskog porta. Budući da, za razliku od prijema, operacija slanja nije toliko vremenski kritična, slanje se obavlja bajt po bajt i to izvan prekidne rutine, pri čemu se prilikom slanja svakog karaktera prvo čeka indikacija da je registar za slanje slobodan (bit UDREn u okviru registra UCSRnA).

[1] Da bi funkcije za prijem ispravno radile, neophodno je da dužina prijemnog bafera bude 2^N bajta.

7.5.1 Opis funkcija

U nastavku je dato objašnjenje svih funkcija unutar biblioteke **usart**.

```
void usartInit(uint32_t baud);
```

Opis: Funkcija inicijalizuje USART modul.

Parametri:

- `baud` – brzina serijske komunikacije, izražena u *bit/s*

Povratna vrednost:
Nema.

```
uint8_t usartAvailable();
```

Opis: Funkcija vraća broj neočitanih karaktera u prijemnom baferu.

Parametri:
Nema.

Povratna vrednost:
Broj neočitanih karaktera u prijemnom baferu.

```
void usartPutChar(int8_t c);
```

Opis: Funkcija šalje zadati karakter preko serijskog porta.

Parametri:

- `c` – karakter koji je potrebno poslati

Povratna vrednost:
Nema.

```
void usartPutString(int8_t *s);
```

Opis: Funkcija šalje string (niz karaktera) smešten u RAM memoriji preko serijskog porta.

Parametri:

- `s` – pokazivač na string u RAM memoriji mikrokontrolera

Povratna vrednost:
Nema.

```
void usartPutString_P(const int8_t *s);
```

Opis: Funkcija šalje string (niz karaktera) smešten u FLASH memoriji preko serijskog porta.

Parametri:

- `s` – pokazivač na string u FLASH memoriji mikrokontrolera

Povratna vrednost:
Nema.

```
int8_t usartGetChar();
```

Opis: Funkcija vraća sledeći nepročitani karakter iz prijemnog bafera.

Parametri:
Nema.

Povratna vrednost:
Naredni nepročitani karakter iz prijemnog bafera. Ukoliko je prijemni bafer prazan, povratna vrednost će biti -1.

```
uint8_t usartGetString(int8_t *s);
```

Opis: Funkcija očitava sve karaktere iz prijemnog bafera i od njih formira string.

Parametri:

- `s` – pokazivač na bafer u koji se smešta string

Povratna vrednost:
Broj očitanih karaktera iz prijemnog bafera.

7.6 Zadaci za vežbu – I deo

Zadatak 7.6.1. Napisati program koji korišćenjem serijskog terminala traži od korisnika da unese svoje ime, nakon čega ispisuje pozdravnu poruku. Primer unosa je dat u nastavku.

IZLAZ: `Unesite svoje ime i prezime:`
ULAZ: `Ime Prezime`
IZLAZ: `Zdravo, Ime Prezime.:)`

Zadatak 7.6.2. Napisati program koji ispisuje broj samoglasnika u datoj reči, koja je uneta putem serijskog terminala. Podrazumevati da se reč unosi isključivo malim slovima. Primer unosa je dat u nastavku.

IZLAZ: `Unesite rec:`
ULAZ: `australopithecus`
IZLAZ: `Rec ima 7 samoglasnika`

Zadatak 7.6.3. Napisati program koji čeka da korisnik unese string putem serijskog terminala, a zatim ga vraća u kapitalizovanom obliku (pretvara mala slova u velika). Primer unosa je dat u nastavku.

ULAZ: `OvajTekstJePotrebnoKapitalizovati`
IZLAZ: `OVAJTEKSTJEPOTREBNOKAPITALIZOVATI`

Zadatak 7.6.4. Napisati program koji proverava da li je niz karaktera unet putem serijskog terminala – *pangram*. Pangram je reč koja sadrži sva slova abecede (bez ponavljanja). Za potrebe ovog zadatka, potrebno je koristiti englesku abecedu. Rezultat provere ispisati putem serijskog terminala. Primer unosa je dat u nastavku.

ULAZ: `The quick brown fox jumps over a lazy dog.`
IZLAZ: `Uneti string jeste pangram.`
ULAZ: `abcd`
IZLAZ: `Uneti string nije pangram.`
ULAZ: `Watch "Jeopardy!", Alex Trebek's fun TV quiz game.`
IZLAZ: `Uneti string jeste pangram.`

Zadatak 7.6.5. Napisati program koji rešava problem "Dragulja i kamenja" (eng. *Jewels and stones*). Na početku, od korisnika se zahteva unos dva niza karaktera koji predstavljaju "dragulje" i "kamenje", u narednom formatu:

`<dragulji>:<kamenje>`

Svaki od nizova može sadržati proizvoljan broj velikih, odnosno malih slova engleske abecede, pri čemu se u nizu dragulja ova slova *ne smeju* ponavaljati.

Cilj zadatka je prebrojati "dragulje" među "kamenjem", odnosno, prebrojati koliko slova iz niza "dragulja" se nalazi u nizu "kamenja". Primer unosa je dat u nastavku.

ULAZ: `aAB:alaBaMA`
IZLAZ: `5`

Zadatak 7.6.6.

(a) Napisati program koji čeka da korisnik unese string putem serijskog terminala, a zatim proverava da li je uneti string palindrom i nakon provere vraća korisniku odgovarajuću indikaciju ("Jeste palindrom", odnosno "Nije palindrom"). Primer unosa je dat u nastavku.

IZLAZ: `Unesite string:`
ULAZ: `arduino`
IZLAZ: `"arduino" nije palindrom.`
IZLAZ: `Unesite string:`
ULAZ: `radar`
IZLAZ: `"radar" jeste palindrom.`

(b) Modifikovati program iz dela zadatka pod (a) tako da se za string, unet preko serijskog terminala, pronalazi njegov najduži podstring koji je palindrom i ispisuje ga putem serijskog terminala. Primer unosa je dat u nastavku.

IZLAZ: `Unesite string:`
ULAZ: `arduinoradararduino`
IZLAZ: `radar`

Zadatak 7.6.7. U okviru biblioteke za serijsku komunikaciju **usart** implementirati funkciju koja kao povratnu vrednost vraća naredni validan broj u nadolazećem nizu karaktera. Prototip funkcije je dat u nastavku. Kod je potrebno detaljno komentarisati.

- `int16_t usartParseInt()`
  - **Opis**: Funkcija koja pretražuje niz karaktera prosleđen od strane korisnika i kao povratnu vrednost vraća prvi validan, ceo broj. Pri tome, uzeti u obzir da se izvršavanje funkcije okončava u slučaju da nije moguće locirati broj u datom nizu nakon isteka vremenskog intervala od 2s (eng. *time-out*).
  - **Povratna vrednost**: Naredni validan broj ili vrednost 0, ukoliko on nije pronađen.

U nastavku su dati primeri unosa, očekivane povratne vrednosti i sadržaj serijskog bafera nakon poziva funkcije *usartParseInt()*.

ULAZ: `123`
Povratna vrednost: `123`
Serijski bafer: `prazan`
ULAZ: `-123`
Povratna vrednost: `-123`
Serijski bafer: `prazan`
ULAZ: `abc123`

Povratna vrednost: `123`
Serijski bafer: `prazan`
ULAZ: `abc123def`
Povratna vrednost: `123`
Serijski bafer: `def`
ULAZ: `abcdef`
Povratna vrednost: `0`
Serijski bafer: `prazan`

Zadatak 7.6.8. U okviru biblioteke za serijsku komunikaciju **usart** implementirati funkciju koja kao povratnu vrednost vraća naredni validan realan broj u nadolazećem nizu karaktera. Prototip funkcije je dat u nastavku. Kod je potrebno detaljno komentarisati.

- `float_t usartParseFloat()`
  - **Opis**: Funkcija koja pretražuje niz karaktera prosleđen od strane korisnika i kao povratnu vrednost vraća prvi validan, realan broj. Pri tome, uzeti u obzir da se izvršavanje funkcije okončava u slučaju da nije moguće locirati broj u datom nizu nakon isteka vremenskog intervala od 2s (eng. *time-out*).
  - **Povratna vrednost**: Naredni validan broj ili vrednost 0, ukoliko on nije pronađen.

U nastavku su dati primeri unosa, očekivane povratne vrednosti i sadržaj serijskog bafera nakon poziva funkcije *usartParseInt()*.

ULAZ: `1.23`
Povratna vrednost: `1.23`
Serijski bafer: `prazan`
ULAZ: `-1.23`
Povratna vrednost: `-1.23`
Serijski bafer: `prazan`
ULAZ: `abc1.23`
Povratna vrednost: `1.23`
Serijski bafer: `prazan`
ULAZ: `abc1.23def`
Povratna vrednost: `1.23`
Serijski bafer: `def`
ULAZ: `abcdef`
Povratna vrednost: `0.0`
Serijski bafer: `prazan`

Zadatak 7.6.9. U okviru biblioteke za serijsku komunikaciju **usart** implementirati funkciju koja putem serijskog porta šalje ceo broj. Prototip funkcije je dat u nastavku. Kod je potrebno detaljno komentarisati.

- `void usartPrintInteger(int32_t value)`
  - **Opis**: Funkcija koja pretvara ceo broj `value` u niz karaktera i šalje ga putem serijskog porta.
  - **Povratna vrednost**: Nema.

Zadatak 7.6.10. U okviru biblioteke za serijsku komunikaciju **usart** implementirati funkciju koja putem serijskog porta šalje realan broj sa odgovarajućim brojem decimalnih cifara. Prototip funkcije je dat u nastavku. Kod je potrebno detaljno komentarisati.

- `void usartPrintFloat(float value, uint8_t num_of_digits)`
  - **Opis**: Funkcija koja pretvara realan broj `value` u niz karaktera u formatu gde je broj decimalnih cifara realnog broja određen parametrom `num_of_digits` i šalje ga putem serijskog porta.
  - **Povratna vrednost**: Nema.

Zadatak 7.6.11. Napisati program koji proverava da li je broj unet putem serijskog terminala "srećan" (eng. *lucky number*). "Srećni" brojevi predstavljaju podskup prirodnih brojeva, koji se generiše eliminacijom brojeva na odgovarajućim pozicijama. Naime, polazeći od skupa prirodnih brojeva:

```
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13...
```

potrebno je eliminisati svaki drugi (paran) broj:

```
1, 3, 5, 7, 9, 11, 13...
```

Nakon toga, iz preostalog skupa prvi broj koji nije uklonjen (nakon broja 1) je broj 3, pa je, stoga, u narednom koraku potrebno eliminisati svaki treći broj:

```
1, 3, 7, 9, 13...
```

Postupak eliminacije je potrebno dalje nastaviti za svaki n-ti broj, gde n predstavlja svaki naredni broj u skupu koji nije eliminisan. Sledeći takav broj bi bio broj 7. Brojevi koji ne budu eliminisani u tom procesu se smatraju "srećnim". To su brojevi:

```
1, 3, 7, 9, 13, 15, 21, 25, 31...
```

U okviru zadatka potrebno je napisati funkciju *isLucky()* koja proverava da li je broj "srećan" ili ne. Prototip funkcije je dat u nastavku.

- `int8_t isLucky(int16_t num);`
  - **Opis**: Funkcija koja proverava da li je broj "srećan".
  - **Povratna vrednost**: Vrednost 1 ukoliko je broj "srećan", odnosno 0, ukoliko nije.

Primer unosa je dat u nastavku.

IZLAZ: `Unesite broj:`
ULAZ: `141`
IZLAZ: `141 - srecan broj.`

Zadatak 7.6.12. Napisati program koji čeka da korisnik unese decimalni broj preko serijskog porta, nakon čega vraća vrednost unetog broja u heksadecimalnom formatu, sa prefiksom `0x`. Primer unosa je dat u nastavku.

IZLAZ: `Unesite decimalan broj:`
ULAZ: `1000`
IZLAZ: `Uneli ste vrednost 0x3E8`

Zadatak 7.6.13. Napisati program koji obavlja funkciju jednostavnog kalkulatora. Korisnik unosi dva cela broja i operator između njih (operatori su '+', '-', '*' i '/'), a program računa i ispisuje vrednost izraza. Primer unosa je dat u nastavku.

IZLAZ: `Unesite brojni izraz:`
ULAZ: `176-39`
IZLAZ: `Vrednost izraza 176-38 je 137`