

# Osnove mikroprocesorske elektronike

## Vaja 5: Serijska komunikacija RS232

---

### Naloge:

1. S pomočjo sheme MIŠKOta ugotovite kateri USART modul (0 ali 1) je priključen na USB povezavo z računalnikom in zanj napišite inicializacijo `void Init_RS232()`.  
Uporabite sledeče nastavitve:
  - o hitrost 115200 baud,
  - o 8 podatkovnih bitov,
  - o 1 stop bit,
  - o brez nadzora paritete in
  - o brez nadzora pretoka podatkov (ta nastavev velja samo za program na PCju).
2. Napišite funkcijo `void RS232_sendByte(char c)`, ki počaka, da je oddajnik prost in nato odpošlje en znak.
3. Napišite funkcijo `int RS232_isDataWaiting()`, ki bo vrnila logično "1", če nas v sprejemnem medpomnilniku čaka nov znak, oziroma "0", če še ni prispel noben nov znak. Namesto funkcije lahko napišete tudi makro.
4. Napišite funkcijo `char RS232_getByte()`, ki bo vrnila zadnji sprejeti podatek, oziroma vrednost 0, če v medpomnilniku ni nobenega novega podatka.
5. Naložite si program HTerm (<http://www.der-hammer.info/terminal/>) ali kaj podobnega, nastavite parametre povezave enako kot na MIŠKO-tu in preizkusite delovanje novih podprogramov:
  - o 1. test - pošljite 1 znak vsako sekundo (npr.: `RS232_sendByte('a');`);
  - o 2. test – glavno zanko napišite tako, da bo ves čas preverjala, ali je prispel kak podatek, in če pride nov podatek, ga takoj pošljite nazaj.
6. Napišite funkcijo `int RS232_putc(char c, FILE *stream)`, ki odpošlje en znak. Ta funkcija naj samo kliče funkcijo `RS232_sendByte(char c)` in nato vrne 0 (0 pomeni OK, -1 bi pomenilo napako). To funkcijo potrebujemo za funkcijo `printf`, ki za pisanje vsakega znaka pokliče funkcijo za zapis enega znaka. Funkcija `printf` pričakuje, da bo funkcija za pošiljanje enega znaka sprejela točno določene parametre in vrnila vrednost tipa `int`, zato ji moramo pripraviti ustrezno funkcijo.
7. Dodajte podporo za `printf`, tako da vključite knjižnico `stdio.h` (`#include <stdio.h>`) in z makrojem `FDEV_SETUP_STREAM` nastavite podatkovni tok (globalno spremenljivko `FILE RS232str`). Nastavite globalno spremenljivko `stdout`, da bo kazala na `RS232str`.
8. Napišite program, ki bo sprejemal podatke po serijskem portu in glede na sprejeti znak prižgal ali ugasnil ustrezno LED diodo (številke "0" do "7" spremenijo stanje ustrezne LED diode, "a" prižge vse LED diode, "n" ugasne vse LED diode) in hkrati bral tipkovnico in vsakič, ko bo pritisnjena tipka, to sporočil nazaj po serijski povezavi v obliki "Pritisnjena je bila tipka 3".

## Navodila:

Navodila za uporabo standardnih knjižnic za AVR najdete na <http://www.nongnu.org/avr-libc/user-manual/modules.html>, oziroma v spletni iskalnik vpišete "avr-libc" in malo pobrsbate.

### 1. Inicializacija USART modula:

- Izračunajte vrednost UBRR registra za hitrost 115200 baud in ga nastavite. Pomagajte si lahko s formulami na strani 170, tabelo primerov za različne frekvence oscilatorjev na koncu poglavja USART ali z orodji iz util/setbaud.h (glej libc Library Reference). Frekvenco uporabljenega oscilatorja ugotovite tako, da z MIŠKOta snamete LCD in pogledate kaj piše na kristalu.
- Nastavite delovanje USART modula na asinhroni način, ustrezno število bitov v vsaki besedi, pariteto in nadzor pretoka podatkov.
- Vključite sprejemnik in oddajnik, prekinitve pa pustite izključene.

### 2. `void RS232_sendByte(char c):`

- čakaj dokler ni oddajnik prost
- pošlji znak iz parametra c.

### 3. `int RS232_isDataWaiting():`

- vrni vrednost bita, ki sporoča, če nas čaka novi podatek

### 4. `char RS232_getByte()`

- (še enkrat) preveri, če nas čaka novi podatek
- Če je novi podatek na voljo, vrni podatek
- Sicer vrni 0.

### 7. `FILE RS232str:`

- Opis nastavitve podatkovnega toka in primer najdete v navodilih za stdio.h (povezava na začetku navodil, stdio.h). Na naslednji strani je nepopoln primer, kako bomo to naredili mi.
- Spremenljivka tipa `FILE` je pravzaprav struktura, v kateri so spravljene vse spremenljivke z nastavitvami za branje in/ali pisanje iz/na določeno napravo. Naprava je lahko prikazovalnik, tipkovnica, komunikacijska povezava, datoteka,... Najosnovnejše spremenljivke v strukturi `FILE` so
  - Kazalec na funkcijo, ki zna napisati 1 znak na napravo (funkcija mora vrniti parameter tipa `int`, in sprejeti parameter tipa `char` in `FILE*`)
  - Kazalec na funkcijo, ki zna prebrati 1 znak iz naprave (funkcija mora vrniti parameter tipa `int` in sprejeti parameter tipa `FILE*`)
  - Podprte funkcije naprave (ali lahko nanjo pišemo (`_FDEV_SETUP_WRITE`), ali lahko z nje beremo (`_FDEV_SETUP_READ`), ali oboje (`_FDEV_SETUP_RW`)).

Te parametre lahko nastavimo s pomočjo makroja `FDEV_SETUP_STREAM`.

- Delovanje preizkusite z ukazom `fprintf(&RS232str, "Deluje!");`, ki ga poženete takoj za inicializacijo (pred `while (1)`).
- V inicializaciji nastavite kazalec `stdout` (deklariran je že v `stdio.h`), da bo kazal na `RS232str`. `printf("Lalala")` je enako kot `fprintf(stdout, "Lalala")`, zato lahko od trenutka, ko je kazalec `stdout` nastavljen, uporabljate funkcijo `printf`.

Primer programa z nastavljenim podatkovnega tokom:

```
#include <avr/io.h>
#include <stdio.h>

void Init_IO();
int RS232_putc(char c, FILE *stream);

FILE RS232str = FDEV_SETUP_STREAM(RS232_putc, NULL, _FDEV_SETUP_WRITE);

int main(void)
{
    Init_IO();
    printf("This will not work, because stdout is not set.\n");
    fprintf(&RS232str, "This will work, if Init_RS232 and RS232_putc are finished.");
    stdout = &RS232str;
    printf("Now that stdout has been set, this will work too.\n");

    while(1)
    {
        //TODO:: Please write your application code
    }
}

int RS232_putc(char c, FILE *stream)
{
    //TODO: add your code here

    return 0;
}
```