Osnove mikroprocesorske elektronike

Priprava 7: Serijski vmesnik SCI – "polling"

Namen te priprave je, da si v novem projektu *pripravite vso potrebno nizko-nivojsko programsko kodo*, s pomočjo katere bomo nato na podlagi USART periferne enote implementirali *splošno-namenski serijski vmesnik* SCI (angl. Serial Communication Interface) na *sistemskem* nivoju.

Priprava na vajo

Doma pripravite naslednje stvari:

1. ***s pomočjo orodja*** `STM Cube Project Copy` ***ustvarite nov projekt z imenom***

   `VAJA_07-USART_polling`

   na podlagi projekta, kjer ste rešili prejšnjo vajo. Projekt nato uvozite v "STM Cube IDE".

2. ***Novemu projektu dodajte modula*** `SCI.c` ***in*** `SCI.h`, znotraj katerega bomo v sklopu vaje implementirali serijski vmesnik SCI na podlagi periferne enote USART. Modula najdete v mapi "predloge". Dodajte ju v projektno mapo znotraj podmape "`System`" na ustrezno mesto.

3. ***Preučite, kateri pini mikrokrmilnika so uporabljeni za serijsko komunikacijo preko vmesnika*** `USART3`**.**

   Na podlagi serijskega vmesnika `USART3` bomo namreč implementirali serijski vmesnik SCI na sistemskem nivoju.
   Pomagajte si seveda z električno shemo Miškota.

4. ***V grafičnem vmesniku CubeMX nastavite ti dva pina kot oddajni pin*** `Tx` ***in sprejemni pin*** `Rx` ***vmesnika*** `USART3` ***in hkrati poskrbite za sledeče nastavitve vmesnika:***

   a) asinhroni način delovanja (angl. asynchronous mode)
   b) strojni način nadzora pretoka podatkov (angl. flow control): onemogočen
   c) hitrost prenosa (angl. baud rate): 115200 bit/s
   d) dolžina podatkov (angl. word length): 8 bitov
   e) pariteta (angl. parity): brez
   f) število stop bitov (angl. stop bits): 1
   g) *onemogočite napako ob prepisu prejetega podatka* (angl. overrun error)
   h) preostale nastavitve pustite na privzetih vrednostih (angl. default values)

   Namig: parametre `USART` vmesnika nastavljate znotraj rubrik:

   "Pinout & Configuration → Connectivity → USART3 → Mode" ter

   "Pinout & Configuration → Connectivity → USART3 → Configuration →

   → Parameter Settings".

5. ***Poskrbite, da se bo za*** `USART` ***modul uporabljala knjižnica "LL"*** (angl. low-level) in ne knjižnica HAL.

   Podobno smo to storili za GPIO strojno opremo. Za posamezno strojno opremo lahko izberete vrsto knjižnice znotraj CubeMX znotraj podokna "Project Manager → Advanced Settings → Driver Selector".

6. ***Shranite nastavitve v CubeMX in tako zgenerirajte novo avtomatsko kodo za inicializacijo serijskega vmesnika*** `USART3`**.**

   Podobno kot pri prejšnjih vajah, ste s tem korakom poskrbeli, da se bo *dodala nizko-nivojska programska koda za inicializacijo strojne opreme serijskega vmesnika* `USART3`, ki ga bomo uporabili za implementacijo serijskega vmesnika SCI na *sistemskem* nivoju.

7. ***Preučite, katero LL knjižnico in katere nizko-nivojske funkcije bomo potrebovali za delo z*** `USART` ***vmesnikom.***

   Potrebovali bomo sledečo nizko-nivojsko funkcionalnost LL knjižnice:

   a) pošiljanje 8-bitnega podatka
   b) sprejemanje 8-bitnega podatka
   c) preverjanje, ali je že mogoče v *oddajni register* `USART` vmesnika vpisati nov podatek
   d) preverjanje, ali je v *sprejemnem* delu `USART` vmesnika že na voljo nov sprejeti podatek

   Pomagajte si z uradno dokumentacijo za "LL knjižnico" in z "auto-complete" CTRL+SPACE funkcionalnostjo znotraj "STM Cube IDE" okolja.

   Seznam nizko-nivojskih funkcij, ki jih sistemski modul potrebuje, shranite med komentarji začetnega dela zglavne datoteke `SCI.h` (glejte spodaj). Pomagajte si seveda s "copy-paste". Poskrbite, da bodo pri funkcijah ***navedeni tudi vhodni parametri*** funkcije in ***tip vrnjene vrednosti*** funkcije.

```
13 // ----------- Include other modules (for public) -------------
14
15 // Vključimo nizko-nivojsko LL knjižnjico, da dobimo podporo za delo z USART vmesnikom.
16
17     // DOPOLNI
18
19
20 // Pri implementaciji sistemskih funkcij serijskega vmesnika SCI bomo potrebovali sledeče nizko-nivojske funkcije:
21 //
22 //  // DOPOLNI
23 //
24 //
25
```

   Na tak način na enem mestu pokažete, kako je sistemski modul odvisen od drugih nižje-nivojskih funkcij (angl. dependencies). Hkrati iz takega seznama lahko hitro ugotovite, katere parametre bo potrebno shraniti v "handle" strukturo na sistemskem nivoju, da boste nato lahko uporabljali željene nizko-nivojske funkcije.

NE POZABITE DOMA PRIPRAVLJENI PROJEKT PRINESTI S SEBOJ NA VAJO!

8. Podučite se o uporabi terminalskega programa HTerm.

Serijski vmesnik SCI bomo namreč preizkusili tako, da ga bomo preko USB protokola povezali na računalnik, na katerem pa bomo uporabili terminalski program za serijsko komunikacijo. Tako bomo lahko preko računalnika na Miškota pošiljali sporočila in hkrati na računalniku opazovali, kaj nam Miško pošilja nazaj.

Gradivo v zvezi z uporabo terminalskega programa HTerm najdete v mapi "dodatno".

9. Osvežite znanje o uporabi standardne funkcije `printf()`

Standardno C funkcijo `printf()` bomo namreč priredili tako, da bomo z njeno pomočjo pošiljali sporočila *neposredno preko serijskega vmesnika* SCI.

Za osvežitev znanja si lahko v spodnjem poglavju preberete kratko razlago uporabe `printf()` funkcije, lahko se vrnete nazaj k nalogi iz prve laboratorijske vaje, lahko pa uporabite učbenik Osvojimo C (stran 7).

Uporaba funkcije `printf()`

Funkcija `printf()` in njene "sestrske funkcije" so zelo zmogljive in vsestranske funkcije za *generiranje formatiranega izpisa*: v datoteke, na zaslon, na terminal ali v znakovni niz ipd. Funkcijo `printf()` si namreč lahko *prilagodite* sami tako, da jo *uporabljate za izpis na poljuben vmesnik oziroma medij*. V sklopu te vaje bomo funkcijo `printf()` priredili tako, da bo formatirani izpis "izpisala" preko serijskega vmesnika.

Ideja funkcije `printf()` je sledeča: funkciji podamo "*format znakovnega niza*", ki določa *statični* del sporočila za izpis, torej del sporočila, ki bo vedno enak. Ta del niza služi kot nekakšno "ogrodje" sporočila za izpis. Hkrati pa funkciji `printf()` podamo še *dinamične parametre za izpis* in pa "*navodila*", kje naj te dinamične parametre vstavi v statično sporočilo in v kakšni obliki naj bodo ti dinamični parametri. Poglejte primer spodaj:

```
int pct = 37;
char filename[] = "foo.txt";

printf ("Processing of `%s' is %d%% finished.\nPlease be patient.\n", filename, pct);
```

Zgornja koda bo na proizvedla sledeče sporočilo:

```
Processing of `foo.txt' is 37% finished.
Please be patient.
```

Vidimo, da sta parametra `pct` in `filename` *dinamična* parametra, ves preostali tekst pa je *statično* besedilo. Opazite lahko tudi posebna znaka `%s` in `%d`, ki pa služita kot *navodilo za oblikovanje* dinamičnih parametrov – t. i. *formatni določili*.

Sedaj že lahko slutite, da je generiranje sporočil s pomočjo `printf()` funkcije res razmeroma enostavno in elegantno. Vendar pa niti približno ne poznamo vseh možnosti in trikov, ki jih omogoča `printf()` funkcija. Vprašanje: kje je mogoče najti dokumentacijo v zvezi z uporabo funkcije kot je `printf()`?

Vedno, kadar vas zanimajo *osnovni gradniki programskega jezika C* kot je bodisi sintaksa jezika, osnovne strukture jezika (npr. tabele, kazalci, vejitveni stavki) ali "standardne funkcije" jezika – takrat se je vedno smiselno obrniti k najbolj temeljnim priročnikom jezika C (angl. reference manual). In to v našem primeru pomeni, da pobrskamo za t. i. "GNU C priročniki". Priročnik v zvezi z *osnovno strukturo jezika C* najdete na uradni strani ali pa tudi na eFE. Če pa vas zanimajo *standardne funkcije jezika C*, je pa pravi naslov **priročnik *standardne C knjižnice* `libc`** – bodisi na uradni strani bodisi na eFE.

Torej, če vas *dodatno* zanima funkcija `printf()` iz standardnega nabora knjižnic jezika C, boste njeno dokumentacijo iskali v priročniku knjižnice `libc`**.** Poiskati je torej potrebno poglavje, ki obravnava funkcijo `printf()`, kar pa ni trivialno, saj se ključna beseda "printf" v priročniku pojavi velikokrat. Zato je na mestu **namig**: iščite po ključni besedi "formatted output". Posebej uporabno je poglavje

> "Table of Output Conversions",

kjer imate na enem mestu zbrane vsa formatna določila za oblikovanje dinamičnih podatkov sporočila.