

# Osnove mikroprocesorske elektronike

## Priprava 8: Serijski vmesnik SCI – prekinitve

---

V sklopu te vaje bomo SCI serijski vmesnik, ki smo ga implementirali na podlagi USART periferne enote, *nadgradili* tako, da bo *za sprejem in pošiljanje podatkov uporabljal prekinitve in ciklični medpomnilnik*. Tekom priprave boste v obstoječo SCI rešitev dodali programsko kodo, s pomočjo katere bomo izvedli nadgradnjo. Hkrati boste tudi *preštudirali postopka za sprejem in pošiljanje podatka s pomočjo prekinitev in medpomnilnika*, tako do bo implementacija nadgradnje bolj razumljiva.

### Priprava na vajo

Doma pripravite naslednje stvari:

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

VAJA\_08-USART\_IRQ

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

2. *V CubeMX omogočite globalne prekinitve vmesnika* USART3.

**Namig:** nastavitve v zvezi s prekinitvami najdete znotraj rubrike

"Pinout & Configuration → Connectivity → USART3 → Configuration →  
→ NVIC Settings"

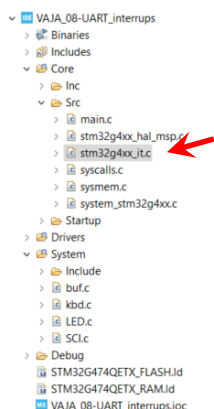
kjer omogočite "USART3 global interrupt".

3. *Shranite nastavitve v CubeMX in tako zgenerirajte novo avtomatsko kodo za inicializacijo prekinitev* USART3 *vmesnika*.

V funkciji za inicializacijo USART3 vmesnika `MX_USART3_UART_Init()` se bo sedaj pojavila sledeča koda:

```
/* USART3 interrupt Init */  
NVIC_SetPriority(USART3_IRQn, NVIC_EncodePriority(NVIC_GetPriorityGrouping(),0, 0));  
NVIC_EnableIRQ(USART3_IRQn);
```

V datoteki Core\Src\stm32g4xx\_it.c, ki vsebuje prekinitvene rutine, se bo pojavila prazna prekinitvena rutina USART3\_IRQHandler() (poglejte spodaj).



```

193
194= /*****
195 /* STM32G4xx Peripheral Interrupt Handlers
196 /* Add here the Interrupt Handlers for the used peripherals.
197 /* For the available peripheral interrupt handler names,
198 /* please refer to the startup file (startup_stm32g4xx.s).
199 *****/
200
201= /**
202  * @brief This function handles USART3 global interrupt / USART3 wake-up interrupt through EXTI line 28.
203  */
204= void USART3_IRQHandler(void)
205 {
206 /* USER CODE BEGIN USART3_IRQn 0 */
207
208 /* USER CODE END USART3_IRQn 0 */
209 /* USER CODE BEGIN USART3_IRQn 1 */
210
211 /* USER CODE END USART3_IRQn 1 */
212 }
213
214 /* USER CODE BEGIN 1 */
215
216 /* USER CODE END 1 */
217
218

```

#### 4. Novemu projektu dodajte programsko kodo, ki jo najdete v treh datotekah znotraj mape "predloge".

Kot to namigujejo imena teh treh datotek, bo potrebno programsko kodo dodati trem datotekam vašega projekta:

- datoteki SCI.h
- datoteki SCI.c
- datoteki stm32g4xx\_it.c

Programsko kodo dodajte na smiselna mesta v teh datotekah. Tekom vaje boste to programsko kodo dopolnili do polne funkcionalnosti.

#### 5. Preučite, katere dodatne nizko-nivojske funkcije bomo potrebovali za delo z USART vmesnikom s pomočjo prekinitv.

Potrebovali bomo sledečo nizko-nivojsko funkcionalnost LL knjižnice:


- a) *omogočanje* specifičnih USART prekinitv, ki se zgodijo:
  - ko se v sprejemnem podatkovnem registru RDR pojavi nov podatek;
  - ko se sprosti oddajni podatkovni register TDR in lahko vanj vpišemo nov podatek.

**Namig:** funkcije, ki se tičejo dela s prekinitvami, imajo običajno v svojem imenu črki "IT" (tj. "InTerrupt").

- b) *onemogočanje* specifične USART prekinitve, ki se zgodijo:
  - ko se sprosti oddajni podatkovni register TDR in lahko vanj vpišemo nov podatek.
- c) *preverjanje*, ali so omogočene prekinitve ob sledečih dogodkih:
  - ko se v sprejemnem podatkovnem registru RDR pojavi nov podatek;
  - ko se sprosti oddajni podatkovni register TDR in lahko vanj vpišemo nov podatek.

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

***S temi funkcijami dopolnite seznam nizko-nivojskih funkcij, ki jih sistemski modul SCI potrebuje in so navedene v komentarju začetnega dela zglavne datoteke SCI.h (glejte spodaj).***

```
// ----- Dopolniti je potrebno seznam uporabljenih LL funkcij -----  
  
// Pri implementaciji sistemskih funkcij serijskega vmesnika SCI bomo potrebovali  
// sledeče nizko-nivojske funkcije:  
// - void LL_USART_TransmitData8(USART_TypeDef *USARTx, uint8_t Value)  
// - uint8_t LL_USART_ReceiveData8(USART_TypeDef *USARTx)  
// - uint32_t LL_USART_IsActiveFlag_TXE_TXFNF(USART_TypeDef *USARTx)  
// - uint32_t LL_USART_IsActiveFlag_RXNE_RXFNE (USART_TypeDef * USARTx)  
//  
// DOPOLNI seznam z novimi LL funkcijami,   
// s pomočjo katerih uporabljamo USART prekinitve
```

**NE POZABITE DOMA PRIPRAVLJENI PROJEKT PRINESTI S SEBOJ NA VAJO!**

**6. *Preštudirajte postopek* za sprejemanje novega podatka s pomočjo prekinitev in medpomnilnika.**

Opis postopka najdete v poglavju spodaj.

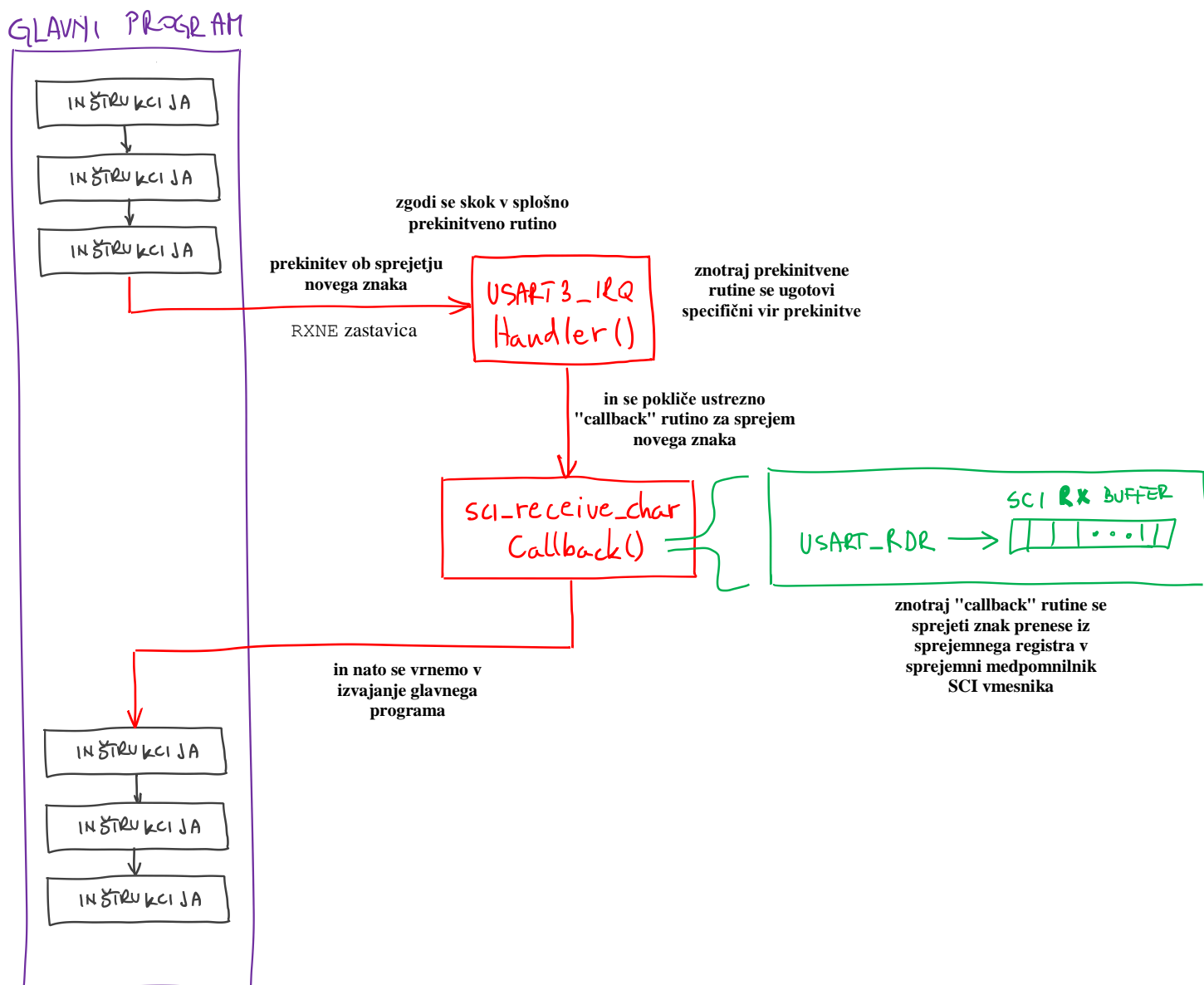
**7. *Preštudirajte postopek* za pošiljanje podatkov s pomočjo prekinitev in medpomnilnika.**

Opis postopka najdete v poglavju spodaj.

## Dodatna obrazložitev

### Ideja sprejemanja novega podatka s pomočjo prekinitev in medpomnilnika

Preštudirajte spodnji diagram, ki simbolično pokaže postopek *sprejema novega podatka* s pomočjo prekinitev in medpomnilnika. Razumevanje tega postopka vam bo olajšalo razumevanje programske kode, ki jo boste morali dopolniti pri tej vaji.



"Callback" funkcija (angl. *callback function*) je posebej specificirana funkcija, ki jo mora določena programska koda poklicati (tj. "callback"). V našem primeru mora programska koda splošne prekinitvene rutine `USART3_IRQHandler()` poklicati funkcijo, ki implementira shranjevanje novo-sprejetega podatka v SCI sprejemni medpomnilnik, torej `SCI_receive_char_Callback()`.

## Ideja pošiljanja podatkov s pomočjo prekinitev in medpomnilnika

Preštudirajte spodnji diagram, ki simbolično pokaže postopek *pošiljanja podatkov* s pomočjo prekinitev in medpomnilnika.

