Komunikace po sériovém kanále USART

Navrhněte v jazyce C program realizující přenos a zobrazení číselné informace na čtyřmístném displeji z předchozí úlohy. Znaky budou zadávány v programu terminál v počítači PC. Přijatá informace v ASCII kódu se bude na displeji posouvat zleva doprava. Nově přijaté číslo bude okamžitě zobrazeno na displeji. Pokud se přijmou více než 4 čísla, budou se přepisovat zleva doprava.

Bonus

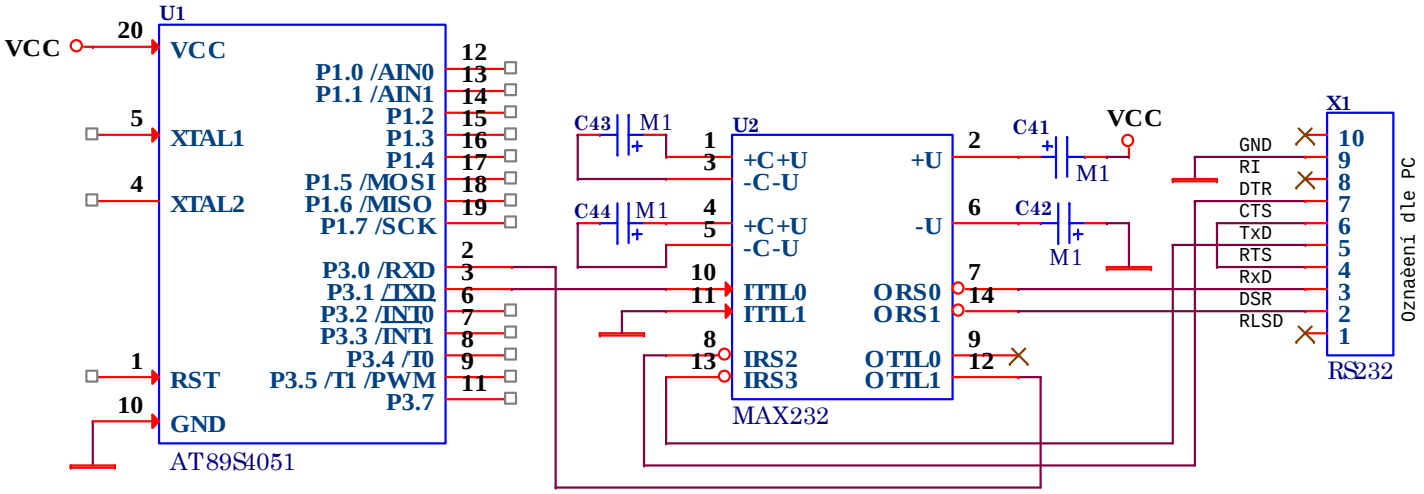
*Přijaté číslo může být i desetinné (od 0.000 do 9999) s pouze jednou desetinnou tečkou. Přijaté číslo se zobrazí až po přijetí znaku nového řádku \n (0x0A). V případě přijetí jiného znaku než čísla a případně další desetinné tečky se na displeji zobrazí **Err.***

SÉRIOVÝ KANÁL USART A JEHO POUŽITÍ - OBECNĚ

Stále používaným způsobem komunikace mezi mikroprocesory nebo mikroprocesorem a zařízením je **asynchronní sériová komunikace (UART)**. Komunikace může být realizována:

- Přímo mezi procesory na PCB v úrovních TTL.
- Po komunikačních sběrnicích RS232 (na ústupu), RS422 nebo RS485, případně po sběrnici USB.

Rozhraní UART (USART) je procesorem podporováno v úrovních TTL, sběrnice RS232 pracuje s úrovněmi ±(6÷15)V. Přechod mezi úrovněmi zajišťuje převodník úrovní. Pro rozhraní UART je to MAX232(3). Obvod dvojnásobí napájecího napětí a vytvoří napětí s opačnou polaritou (+10/6,6V a -10/6,6V).



SÉRIOVÝ KANÁL USART A JEHO POUŽITÍ - OBECNĚ

- Komunikace po sběrnici RS232 se realizuje v minimální variantě pomocí 3 vodičů RxD, TxD a Zem. Norma pro komunikaci stanovuje maximální vzdálenost zařízení na 15[m] při komunikační rychlosti 20kb.
- Komunikaci po RS422 (Full-duplex) zajišťuje celkem 4(5) vodičů (symetrický pár pro vysílání a příjem) a zem. Převodník mezi rozhraními bývá často nahrazen dvěma převodníky RS485. Maximální vzdálenost je 1200 [m] při rychlosti 10Mb.
- V případě RS485 (Half-duplex) je potřeba jeden symetrický pár a případně zem. Převod mezi rozhraními zajišťuje jeden převodník RS485. Vzdálenost a komunikační rychlost je stejná jako u RS422.

Pozor na rozdíl zemních potenciálů.

- Komunikační rozhraní USB podporují procesory buď přímo nebo je tvořeno převodníky RS232⇔USB nebo FIFO⇔USB. Maximální vzdálenost komunikace bez další podpory je 6 [m] na rychlosti dané typem sběrnice. (USB1.1, USB2.0, USB3.0, USB3.1).

SÉRIOVÝ KANÁL USART A JEHO MOŽNOSTI NA PROCESORU ARM

Univerzální sériový asynchronní přijímač a vysílač (USART) nabízí řadu možností komunikace s externími zařízeními. Výměna dat může být **full-duplex** nebo **half-duplex** při asynchronní nebo synchronní komunikaci s širokým rozsahem přenosových rychlostí generovaných fraktálním generátorem. Rozhraní umožňuje:

- Sběrnici LIN - nová asynchronní sériová sběrnice používající ke komunikaci po jednom vodiči k připojeným zařízením
- Smartcard Protocol
- IrDA (infrared data association) se SIR enkodérem podporujícím infračervený přenos 3/16
- Half-duplex synchronní komunikaci s vodiči DATA a CLK
- Half-duplex jednodrátovou komunikaci
- Full-duplex asynchronní komunikaci s modemovou podporou
- Podporuje multiprocesorou komunikaci
- NRZ standardní formát (Mark/Space)
- Vysokorychlostní datovou komunikaci pomocí DMA pro více bufferovou komunikaci

SÉRIOVÝ KANÁL USART A JEHO MOŽNOSTI NA PROCESORU ARM

Jednotka sériového přenosu umožňuje:

- Konfigurovatelné převzorkování 16 nebo 8 pro spolehlivé zajištění přenosové rychlosti a toleranci hodinového signálu
- Fraktální generátor přenosové rychlosti umožňující až 4 Mbit/s pokud APB frekvence je 32 MHz a převzorkování 8
- Programovatelnou délku slova 8 a 9 bitů s 1 nebo 2 stop bity
- Funkci pro odesílání a detekci zlomu (LIN Master) a 13-bitové přerušení, detekci přerušení (10/11), pokud je u funkce USART nakonfigurována pro LIN
- Hodinový signál pro synchronní přenos
- Funkci emulace čipové karty – rozhraní definované v SmartCard normě ISO 7816-3 s 0,5, 1,5 stop-bitem
- Ukládání přijatých/odeslaných bajtů do rezervované SRAM pomocí DMA
- Separátní bity pro povolení vysílání a přijímání
- Disponuje těmito indikátory přenosu
  - Přijímací buffer je plný
  - Vysílací buffer je prázdný
  - Ukončení přenosu

SÉRIOVÝ KANÁL USART A JEHO MOŽNOSTI NA PROCESORU ARM

- Kontrola parity
  - Generování paritního bitu pro vysílání
  - Kontrola parity při příjmu
- Čtyři chybové indikátory
  - Chyba přetečení (Overrun error)
  - Chyba rámce (Frame error)
  - Detekce šumu
  - Chyba parity (Parity error)
- Deset zdrojů přerušení s indikátory
  - Změna CTS
  - Vysílací registr prázdný
  - Přijímající registr plný
  - Chyba přetečení
  - Chyba šumu
  - LIN break detection
  - Přenos dokončen
  - Nečinné vedení Idle
  - Chyba rámce
  - Chyba parity
- Komunikace s více procesory – vstup do režimu ztlumení, pokud nedošlo ke shodě v adrese
- Probuzení z režimu ztlumení (při detekci nečinné linky nebo detekce adresy. Dva režimy probuzení přijímače:
  - bit adresy (MSB, 9 bit)
  - linka v nečinnosti

SÉRIOVÝ KANÁL USART NA PROCESORU ARM

Komunikace mezi zařízeními se uskutečňuje přes vývody RX a TX. RX slouží pro příjem, který je převzorkováván pro případ přicházejících dat poškozených šumem. Vývod TX slouží k vysílání sériové komunikace a není-li co přenášet je TX v úrovni log.1. V normálním režimu USART přenáší TX a RX sériová data jako rámce obsahující:

- Nečinnou linku před přenosem nebo příjmem
- Startovací bit
- Datové slovo (8 nebo 9 bitů) od nejnižšího bitu k nejvyššímu. Devátý bit může představovat paritní bit nebo identifikovat v multiprocesorové komunikaci data a adresy.
- Stop bit v délce 0,5,1, 1,5, 2 násobku přenášeného bitu

Vlastnosti tohoto rámce určují následující registry:

- Přenosovou rychlost určuje fraktální generátor s 12bitovou mantisou a 4bitovým zlomkem uloženou v registru (USART_BRR)
- Stavový registr (USART_SR)
- Datový register (USART_DR)

Registr přenosové rychlosti (USART_BRR)

Bit 15:4 **DIV_Mantissa[11:0]** – těchto 12 bitů definuje celou část dělícího poměru USARTDIV

Bit 3:0 **DIV_Fraction[3:0]** - zbytek (mantisu) USARTDIV. Když bude OVER8=1, pak nejvyšší bit DIV_Fraction (3 bit) se nebere v úvahu a musí být nulový.

USARTDIV je číslo bez znaménka s pevnou desetinnou čárkou uložené v registru USART_BRR. Přenosová rychlost příjmu a vysílání (Rx a Tx) je stejná naprogramováním hodnot Mantissa a Fraction. Standardní přenosová rychlost USART (i SPI) je dán rovnicí

$$Tx\ i\ Rx = \frac{f_{CK}}{8*(2-OVER8)*USARTDIV}$$

Přenosová rychlost v Smartcard, LIN a IrDa módu dána

$$Tx\ i\ Rx = \frac{f_{CK}}{16*USARTDIV}$$

Příklad:

Pro OVER8=0, Mantissa = 27 a DIV_Fraction = 12

Odtud hodnota registru bude USART_BRR = 0x1BC,

Výpočet dělícího poměru získáme takto:

Mantisa (USARTDIV) = 27
Fraction (USARTDIV) = 12/16 = 0.75

USARTDIV = 27.75.

Při opačném převodu USARTDIV na USART_BRR je potřeba zaokrouhlení na nejbližší hodnotu => 0,97*16=16 => (mantissa+1),0.

Datový registr (USART_DR)

Bit 8:0 **DR[8:0]** Registry pro příjem (RDR) a vysílání (TDR) datových znaků. Přijaté znaky = čtení, zápis = vysílané znaky.

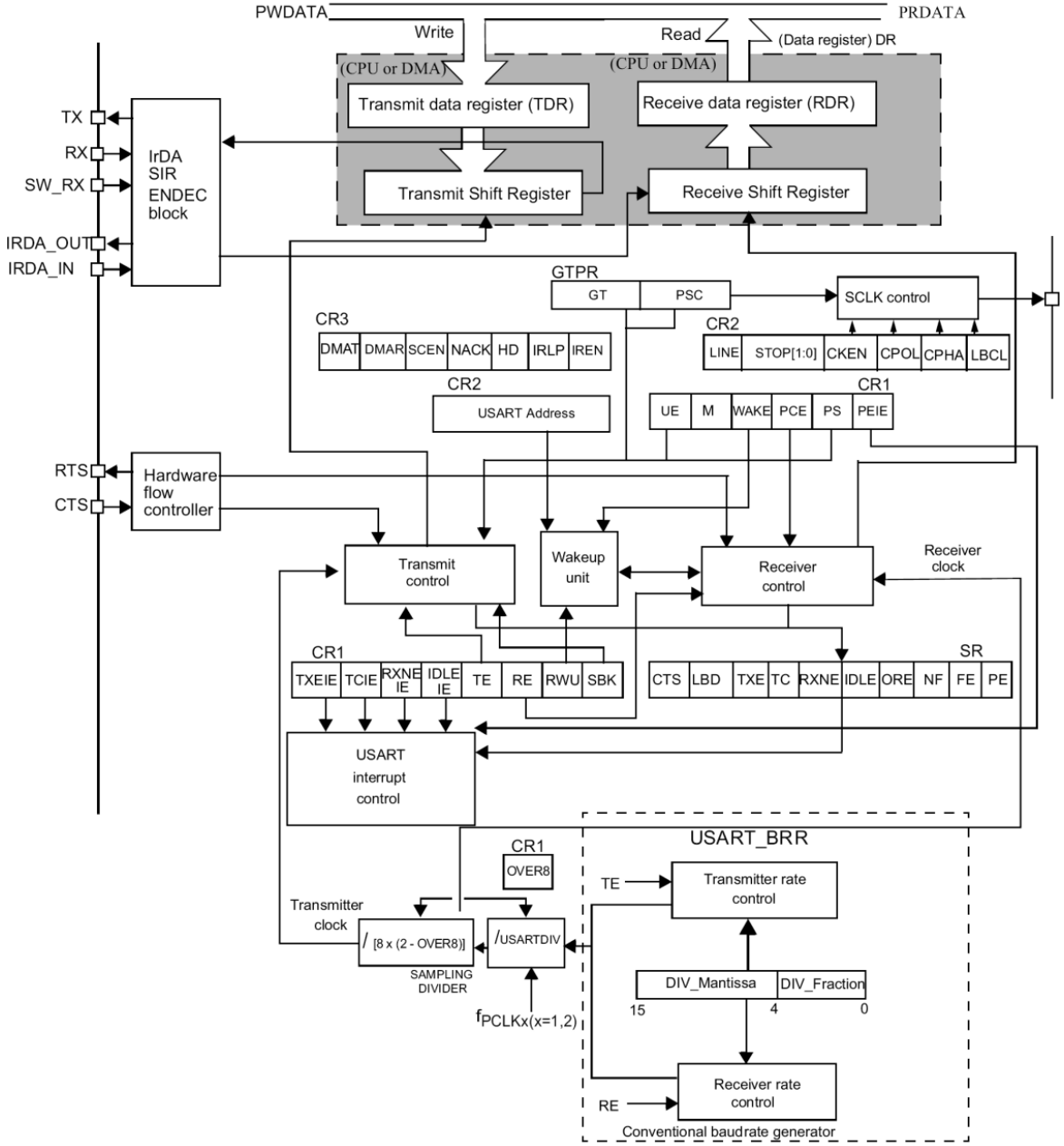
SÉRIOVÝ KANÁL NASTAVENÍ A OBSLUHA

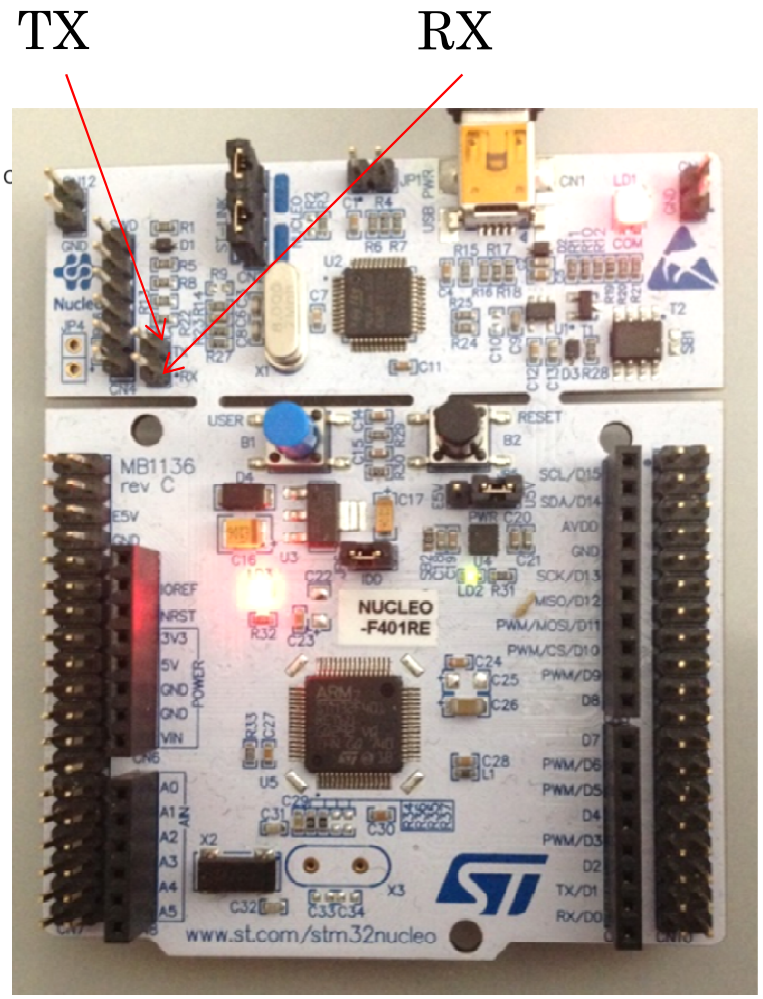
Po nastavení USART je potřeba signály RX a TX spojit s příslušným I/O vývodem procesoru. Na které vývody může být signál daného sériového kanálu připojen určuje tabulka v manuále procesoru. Vybraný vývod je třeba konfigurovat na **alternativní funkci** a dále uložit do registrů AFRL nebo AFRH hodnotu alternativní funkce k vybranému vývodu (viz. ukázka v předchozí úloze).

Port	AF00	AF01	AF02	AF03	AF04	AF05	AF06	AF07	AF08	AF09	AF10	AF11	AF12	AF13	AF14	AF15
	SYS_AF	TIM1/TIM2	TIM3/ TIM4/ TIM5	TIM9/ TIM10/ TIM11	I2C1/I2C2/ I2C3	SPI1/SPI2/ I2S2/SPI3/ I2S3/SPI4	SPI2/I2S2/ SPI3/ I2S3	SPI3/I2S3/ USART1/ USART2	USART6	I2C2/ I2C3	OTG1_FS		SDIO			
PA0	-	TIM2_CH1/ TIM2_ETR	TIM5_CH1	-	-	-	-	USART2_ CTS	-	-	-	-	-	-	-	EVENT OUT
PA1	-	TIM2_CH2	TIM5_CH2	-	-	-	-	USART2_ RTS	-	-	-	-	-	-	-	EVENT OUT
PA2	-	TIM2_CH3	TIM5_CH3	TIM9_CH1	-	-	-	USART2_ TX	-	-	-	-	-	-	-	EVENT OUT
PA3	-	TIM2_CH4	TIM5_CH4	TIM9_CH2	-		-	USART2_ RX	-	-	-	-	-	-	-	EVENT OUT
PA4	-	-	-	-	-	SPI1_NSS	SPI3_NSS/ I2S3_WS	USART2_ CK	-	-	-	-	-	-	-	EVENT OUT
PA5	-	TIM2_CH1/ TIM2_ETR	-	-	-	SPI1_SCK	-	-	-	-	-	-	-	-	-	EVENT OUT

Komunikace po USART mezi deskou NUCLEO a počítačem PC bude probíhat přes debug rozhraní ST link po USB rozhraní.

SÉRIOVÝ KANÁL NASTAVENÍ A OBSLUHA





SÉRIOVÝ KANÁL NASTAVENÍ A OBSLUHA

```
void serial2Init(void){                          // USART2 je připojený k USB
     RCC->APB1ENR|=RCC_APB1ENR_USART2EN;         // Povoleni hodin na UART2
     USART2->CR1|=USART_CR1_UE;                  // Povoleni rozhraní USART2
     USART2->CR1|=USART_CR1_RE;                  // Povoleni příjmu
     USART2->CR1|=USART_CR1_TE;                  // Povoleni vysílání
     USART2->BRR =  0xD05;                       // Nastaveni rychlosti na 9600 pro APB1=32MHz

     RCC->AHBENR |= RCC_AHBENR_GPIOAEN;          // Povoleni hodinového signálu pro bránu GPIOA
     //setbit(RCC->AHB1ENR, 0);                  // Nastaveni hodin portu A
     GPIOA->MODER&= ~GPIO_MODER_MODER2;          // Nastavení stavu po resetu
     GPIOA->MODER|= GPIO_MODER_MODER2_1;         // Nastavení alternativní funkce pro PA.2
     //setbit(GPIOA->MODER, 5);                  // Alternativní funkce pro PA.2
     GPIOA->MODER&= ~GPIO_MODER_MODER3;          // Nastavení stavu po resetu
     GPIOA->MODER|= GPIO_MODER_MODER3_1;         // Nastavení alternativní funkce pro PA.3
     //setbit(GPIOA->MODER, 7);                  // Alternativní funkce pro PA.3

     GPIOA->AFR[0] |= GPIO_AFRL_AFRL2&0x0700;    // Nastaveni alternativní funkce pro USART2_TX
     //GPIOA->AFR[0] = 0x0700 ;                  // Alternativní funkce AF7 na vývod PA2
     GPIOA->AFR[0] |= GPIO_AFRL_AFRL3&0x7000;    // Nastaveni alternativní funkce pro USART2_RX
     //GPIOA->AFR[0] |= 0x07 << 12;              // Alternativní funkce AF7 na vývod PA3
}

void vysli_znak(char znak) {                     // Vysílání znaku s programovou obsluhou
     while (((USART2->SR)&USART_SR_TXE)==0);     // Cekej na vyprázdnění vysílacího buffru
     //while (!getbit(USART2->SR,7));
     USART2->DR = znak;                          // Zápis znaku
}

char prijem_znaku(void) {
     while (((USART2->SR)&USART_SR_RXNE)==0);    // Cekej na příjem znaku
     //while (!getbit(USART2->SR,5));
     return (USART2->DR);                        // Přečtení přijatého znaku
}
```

OVĚŘENÍ KONFIGUARACE SÉRIOVÉHO ROZHRANÍ USART

Cíl: *Vytvořit nový projekt v prostředí Keil a ověřit základní kostru pro realizaci úlohy 4.*

1. *Založení nového projektu – Spustíme program Keil, uzavřeme předcházející projekt. Open project-vytvoříme direktorář-pojmenujeme nový projekt.*
2. *V okně Select device-STMicroelectronics-STM32F4 Series-STM32F401-STM32F401RE-STM32F401RE*
3. *V okně Manage Run-Time Environment volíme Device – Startup, SMSIS – CORE.*
4. *Vytvoříme podprogram INIC-Serial.c str.12, překopírujeme Nastaveni_GPIO.c a SystemCoreClockSetHSI.c.*
5. *Založíme hlavní program, do kterého převezmeme úvodní části z předcházejících úloh.*
6. *Povolíme hodinový signál pro GPIOA a GPIOB*

OVĚŘENÍ KONFIGUARACE SÉRIOVÉHO ROZHRANÍ USART

7. *Nakonfigurujeme vývody sériového kanálu a displeje*

```
PIN_OUTPP_Initialize (GPIOA,2);        // USART_TxD
PIN_IN_Un_Initialize (GPIOA,3);        // USART_RxD
PIN_OUTPP_Initialize (GPIOA,8);        // Displej CLK
PIN_OUTPP_Initialize (GPIOA,9);        // Displej SI
PIN_OUTPP_Initialize (GPIOB,5);        // Displej ZAPIS
```

8. *V části Main() zavoláme*

```
i=prijem_znaku();
```

9. *Osciloskopem budeme sledovat vývod RX str.11. Ověříme přenosovou rychlost.*

10. *Zkopírujeme inicializaci TIM2 – který využijeme pro přepínání jednotlivých zobrazovaných LED segmentovek.*

11. *Dokončíme celé zadání domácího úkolu*