Github: <https://github.com/Smadas/vrs_cv5.git>

VRS zadanie číslo 6

# Main.c

**#include** <stddef.h>

**#include** "stm32l1xx.h"

**#include** "vrs\_cv5.h" // zavolanie vytvorenej knižnice

uint16\_t valueADC = 0;

**int** **main**(**void**)

{

/\* Enable GPIO clock \*/

RCC\_AHBPeriphClockCmd(RCC\_AHBPeriph\_GPIOA, *ENABLE*);

//inicializacia periferii

inicializaciaADCpin();

inicializaciaADC();

inicializaciaLED();

inicializaciaPrerusenieADC();

inicializaciaPrerusenieUSART();

inicializaciaUSART2();

**int** tvarVypisu=0; //prepinanie tvaru odoslanej hodnoty

USARTbuffer[0] = '\0'; //inicializacia buffera

pom='l';

bufferPripraveny = 1;

//prve zapisanie do usart

tvarVypisu = odoslanieRetazca(tvarVypisu);

**while** (1)

{

tvarVypisu = odoslanieRetazca(tvarVypisu);

}

**return** 0;

}

**Funkcia main.c slúži na volanie funkcií, ktoré popíšeme neskôr v dokumente. Na začiatku vpustí hodinové impulzy do GIOPA. Inicializuje ADC, USART a prerušenia. Na pozadí ADC dookola meria napätie na PA2 a v prerušení sa táto hodnota načítava do premennej. V nekonečnom cykle sa volá funkcia „odoslanie reťazca“, naplní buffer hodnotou napätia v požadovanom formáte a odošle prvý znak. Po skončení odoslania znaku nastane prerušenie, počas ktorého sa odošle ďalší znak, ak bol odoslaný posledný znak, tak sa uvoľní buffer aby sa mohol začať posielať ďalší reťazec.**

# vrs\_cvc5.h

**#ifndef** VRS\_CV5\_H\_

**#define** VRS\_CV5\_H\_

**extern** uint16\_t valueADC;

uint16\_t pom;

//void ADC1\_IRQHandler(void);

**void** **inicializaciaPrerusenieADC**(**void**); // inicializácie prerušenia // prevodníka ADC

**void** **inicializaciaPrerusenieUSART**(**void**); // inicializácie prerušenia USART2

**void** **inicializaciaUSART2**(**void**); // konfigurácia USART2

**void** **inicializaciaLED**(**void**);

**void** **inicializaciaADCpin**(**void**);

**void** **inicializaciaADC**(**void**);

**int** **blikanieLED**(**int** blikac, **int** blikacRychlost);

uint16\_t **citanieHodnotyADC**(**void**);

**int** **rychlostBlikaniaLED**(**int** blikacRychlost,uint16\_t value);

**void** **PutcUART2**(**char** \*ch); // slúži na odosielanie reťazca po

//sériovej linke

**void** **USART2\_IRQHandler**(**void**); // slúži na prijímanie dát zo sériovej linky

**int** **vypisDatADC**(**int** i); // slúži na vypisovanie dát vo formáte ”4096” //alebo ”3.30V”.

**#endif** /\* VRS\_CV5\_H\_ \*/

**vrs\_cv5.h je naša vytvorená knižnica, ktorá slúži na volanie vytvorených funkcií**

# vrs\_cvc5.c

**#include** <stddef.h>

**#include** "stm32l1xx.h"

**#include** "vrs\_cv5.h" //zavolanie našej vytvorenej knižnice

**int** **vypisDatADC**(**int** i){

**char** buffer [10]; //definovanie premennej na posielanie dát

**int** hodnota; // pomocná premenná na posielanie dát

**for**(**int** a = 0;a<100000;a++);

**if**(pom == 'm'){ //ak pošleme znak m, zmení sa formát //odosielania hodnôt

**if**(i==0){

i++;

}

**else**{

i=0;

}

pom='l'; // treba zmeniť hodnotu pom, aby v nej //neostal char m, pretože by sa zmenil formát odosielaných dát každým //cyklom.

}

// do funkcie vstupuje hodnota int i, ktorá ak má hodnotu 0, bude //vypisovať dáta vo formáte od ’0 ’ do ’4095 ’, ak int i=1, dáta sa //budú vypisovať vo formáte od ’0.00V ’ do ’3.30V ’.

**if**(i==0){

sprintf(buffer, "%d", valueADC);

**if**(valueADC>=0 && valueADC<10){

buffer[1]=' ';

buffer[2]='\0';

}

**if**(valueADC>=10 && valueADC<100){

buffer[2]=' ';

buffer[3]='\0';

}

**if**(valueADC>=100 && valueADC<1000){

buffer[3]=' ';

buffer[4]='\0';

}

**else**{

buffer[4]=' ';

buffer[5]='\0';

}

PutcUART2(buffer);

}

**else**{

hodnota=valueADC/40.96\*0.033\*100;

sprintf(buffer, "%d", hodnota);

**if**(hodnota<100 && hodnota>= 10){

//upravenie formátu na **’0.xxV** ’, kde x je číslo // od 0 do 9

buffer[4]=buffer[1];

buffer[3]=buffer[0];

buffer[2]='.';

buffer[1]='0';

buffer[0]=' ';

buffer[5]='V';

buffer[6]=' ';

buffer[7]='\0';

PutcUART2(buffer);

}

**else** **if**(hodnota<10 && hodnota>= 0){

//upravenie formátu na **’0.0xV** **’**, kde x je číslo // od 0 do 9

buffer[4]=buffer[0];

buffer[3]='0';

buffer[2]='.';

buffer[1]='0';

buffer[0]=' ';

buffer[5]='V';

buffer[6]=' ';

buffer[7]='\0';

PutcUART2(buffer);

}

**else**{

//upravenie formátu na **’x.xxV ’** , kde x je číslo // od 0 do 9

buffer[4]=buffer[2];

buffer[3]=buffer[1];

buffer[2]='.';

buffer[1]=buffer[0];

buffer[0]=' ';

buffer[5]='V';

buffer[6]=' ';

buffer[7]='\0';

PutcUART2(buffer);

}

}

**return** i;

}

**Funkcia vypisDatADC() slúži na vypísanie nameraných dát z prevodníka ADC do USARTbuffer v hodnotách od 0 do 4095 alebo od 0.00V do 3.30V, pričom 4095 = 3.30V. Z USART buffer sa potom bude zapísaný string posielať po sérovej linke.**

**void** **USART2\_IRQHandler**(**void**)

{

**if**(USART\_GetITStatus(USART2, USART\_IT\_RXNE) != *RESET*)

//indikuje či je prijatý znak, ak áno tak ho prečíta

{

USART\_ClearITPendingBit(USART2, USART\_IT\_RXNE);

pom = USART\_ReceiveData(USART2);

}

**if**(USART\_GetFlagStatus(USART2, USART\_FLAG\_TC) != *RESET*)//indikovanie flagu ukončenia prenosu

{

USART\_ClearFlag(USART2, USART\_FLAG\_TC);

bufferInkr++;

**if**( USARTbuffer[bufferInkr] != '\0')

{

USART\_SendData(USART2, USARTbuffer[bufferInkr]);

}

**else**

{

bufferInkr = 0;

bufferPripraveny = 1;

}

}

}

**Funkcia USART2\_IRGHandler slúži na spravovanie prerušení sériovej linky. Ak vznikne prerušenie kvôli prijatiu znaku zo sériovej linky, tak sa vykoná príkaz na prečítanie tohto znaku.**

**Ak Vznikne prerušenie na základe dokončenia odoslania znaku, tak sa začne odosielať ďalší znak, ak nie je k dispozícii ďalší znak, tak sa uvoľní buffer aby bolo možné poslať ďalší reťazec.**

**void** **PutcUART2**(**char** \*ch)

//Funkcia slúži na odosielanie dát, pričom dáta máme v poli \*ch, //ktoré odosielame po znakoch

{

**int** i = 0;

**while**(ch[i]!='\0'){

USARTbuffer[i] = ch[i];

i++;

}

USARTbuffer[i] = '\0';

}

**Funkcia PutcUART2 slúži na vloženie požadovaného reťazca do USART buffera po znakoch.**

**void** **ADC1\_IRQHandler**(**void**)

//Funkcia slúži na čítanie hodnoty prevodníka ADC od 0 do 4095, ktorá //sa uloží v premennej valueDCA

{

**if** (ADC\_GetFlagStatus(ADC1, ADC\_FLAG\_EOC))//ADC1->SR & ADC\_SR\_EOC)

{

valueADC = ADC\_GetConversionValue(ADC1);

}

**if**(ADC\_GetFlagStatus(ADC1, ADC\_FLAG\_OVR)){}

}

**void** **inicializaciaPrerusenieADC**(**void**)

//funkcia slúži na nastavenie prerušenia ADC, v ktorej sme zvolili //PreemptionPriority = 1, čiže má vyššiu prioritu ako prerušenie USART, //ktoré má PreemptionPriority = 2 (viď funkcia //**inicializaciaPrerusenieUSART()**).

{

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_4);

NVIC\_InitTypeDef NVIC\_InitStructure;

NVIC\_InitStructure.NVIC\_IRQChannel = *ADC1\_IRQn*;

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 1;

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 0;

NVIC\_InitStructure.NVIC\_IRQChannelCmd = *ENABLE*;

NVIC\_Init(&NVIC\_InitStructure);

}

**void** **inicializaciaPrerusenieUSART**(**void**)

//funkcia slúži na nastavenie prerušenia USART2

{

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_4);

NVIC\_InitTypeDef NVIC\_InitStructure;

NVIC\_InitStructure.NVIC\_IRQChannel = *USART2\_IRQn*;

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 2;

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 0;

NVIC\_InitStructure.NVIC\_IRQChannelCmd = *ENABLE*;

NVIC\_Init(&NVIC\_InitStructure);

}

**void** **inicializaciaUSART2**(**void**)

// funkcia najprv spustí hodiny na porte A, potom nakonfiguruje USART Tx //a Rx piny, spustí hodiny pre perifériu USART2 a nakoniec nakonfiguruje //samotný USART2

{

RCC\_AHBPeriphClockCmd(RCC\_AHBPeriph\_GPIOA, *ENABLE*);

/\* Configure USART Tx and Rx pins \*/

GPIO\_InitTypeDef GPIO\_InitStructure;

GPIO\_InitStructure.GPIO\_Mode = *GPIO\_Mode\_AF*;

GPIO\_InitStructure.GPIO\_Speed = *GPIO\_Speed\_40MHz*;

GPIO\_InitStructure.GPIO\_OType = *GPIO\_OType\_PP*;

GPIO\_InitStructure.GPIO\_PuPd = *GPIO\_PuPd\_NOPULL*;

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_2 | GPIO\_Pin\_3;

GPIO\_PinAFConfig(GPIOA, GPIO\_PinSource2, GPIO\_AF\_USART2);

GPIO\_PinAFConfig(GPIOA, GPIO\_PinSource3, GPIO\_AF\_USART2);

GPIO\_Init(GPIOA, &GPIO\_InitStructure);

//usart configuration

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_USART2, *ENABLE*);

USART\_InitTypeDef USART\_InitStructure;

USART\_InitStructure.USART\_BaudRate = 19200;

//BaudRate sme nastavili na 19200, pretože nastala chyba //v knižnici, ktorá nastavuje polovičný BaudRate

USART\_InitStructure.USART\_WordLength = USART\_WordLength\_8b;

USART\_InitStructure.USART\_StopBits = USART\_StopBits\_1;

USART\_InitStructure.USART\_Parity = USART\_Parity\_No;

USART\_InitStructure.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None;

USART\_InitStructure.USART\_Mode = USART\_Mode\_Rx | USART\_Mode\_Tx;

USART\_Init(USART2, &USART\_InitStructure);

USART\_Cmd(USART2, *ENABLE*);

USART\_ITConfig(USART2, USART\_IT\_RXNE, *ENABLE*);

}

**void** **inicializaciaLED**(**void**)

//súčasť minulého zadania

{

//vytvorenie struktury GPIO

GPIO\_InitTypeDef gpioInitStruc;

gpioInitStruc.GPIO\_Mode = *GPIO\_Mode\_OUT*;

gpioInitStruc.GPIO\_OType = *GPIO\_OType\_PP*;

gpioInitStruc.GPIO\_Pin = GPIO\_Pin\_5;

gpioInitStruc.GPIO\_Speed = *GPIO\_Speed\_400KHz*;

//zapisanie inicializacnej struktury

GPIO\_Init(GPIOA, &gpioInitStruc);

}

**void** **inicializaciaADCpin**(**void**)

//súčasť minulého zadania

{

GPIO\_InitTypeDef GPIO\_InitStructure;

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_4 ;

GPIO\_InitStructure.GPIO\_Mode = *GPIO\_Mode\_AN*;

GPIO\_InitStructure.GPIO\_PuPd = *GPIO\_PuPd\_NOPULL* ;

GPIO\_Init(GPIOA, &GPIO\_InitStructure);

}

**Funkcia inicializaciaADCpin slúži na vytvorenie štruktúry pre inicializáciu pinu na ktorom je pripojený výstup z klávesnice.**

**void** **inicializaciaADC**(**void**)

//Nastavenie ADC prevodníka,

{

ADC\_InitTypeDef ADC\_InitStructure;

RCC\_HSICmd(*ENABLE*);

**while**(RCC\_GetFlagStatus(RCC\_FLAG\_HSIRDY) == *RESET*);

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_ADC1, *ENABLE*);

ADC\_StructInit(&ADC\_InitStructure);

ADC\_InitStructure.ADC\_Resolution = ADC\_Resolution\_12b;

ADC\_InitStructure.ADC\_ContinuousConvMode = *ENABLE*;

ADC\_InitStructure.ADC\_ExternalTrigConvEdge = ADC\_ExternalTrigConvEdge\_None;

ADC\_InitStructure.ADC\_DataAlign = ADC\_DataAlign\_Right;

ADC\_InitStructure.ADC\_NbrOfConversion = 1;

ADC\_Init(ADC1, &ADC\_InitStructure);

ADC\_RegularChannelConfig(ADC1, ADC\_Channel\_4, 1, ADC\_SampleTime\_384Cycles);

ADC\_Cmd(ADC1, *ENABLE*);

**while**(ADC\_GetFlagStatus(ADC1, ADC\_FLAG\_ADONS) == *RESET*)

{

}

ADC\_ITConfig(ADC1, ADC\_IT\_EOC, *ENABLE*);

//povolenie prerušenia pri nastavení stavového registra EOC //(end of conversion)

ADC\_ITConfig(ADC1, ADC\_IT\_OVR, *ENABLE*);

//povolenie prerušenia pri nastavení stavového registra ktorý //detekuje či boli všetky dáta správne prečítané

ADC\_SoftwareStartConv(ADC1);//spustenie prevodu ADC

}

**Funkcia inicializaciaADC() slúži na spustenie a skontrolovanie oscilátora či je spustený, spustenie hodín, nakonfigurovanie ADC, nakonfigurovanie kanálu 4, spustenie a kontrolu ADC.**

**int** **blikanieLED**(**int** blikac, **int** blikacRychlost)

//súčasť minulého zadania

{

blikac++;

**if** (blikac > blikacRychlost)

{

blikac = 0;

GPIO\_ToggleBits(GPIOA, GPIO\_Pin\_5);

}

**return** blikac;

}

uint16\_t **citanieHodnotyADC**(**void**)

//súčasť minulého zadania

{

ADC\_SoftwareStartConv(ADC1);

**while**(!ADC\_GetFlagStatus(ADC1, ADC\_FLAG\_EOC)){}

**return** ADC\_GetConversionValue(ADC1);

}

**int** **rychlostBlikaniaLED**(**int** blikacRychlost,uint16\_t value)

//súčasť minulého zadania

{

**if**(value >= 3550 && value <= 3800)

{

**return** 200000;

}

**else** **if**(value > 3200 && value < 3550)

{

**return** 50000;

}

**else** **if**(value >= 2600 && value <= 3200)

{

**return** 20000;

}

**else** **if**(value >= 0 && value < 2600)

{

**return** 5000;

}

**else**

{

**return** blikacRychlost;}}

# Záver:

Naučili sme sa nastavovať a pracovať s prerušeniami, pričom sme na ADC nastavili vyššiu prioritu prerušenia. Aplikácia pri prijímaní znaku m zmení formát odosielania dát z 0 až 4095 na 0.00V až 3.30V a naopak. Pri nastavovaní USART2 a jeho BaudRate sme prišli na chybu v knižnici, ktorá nastavuje jej polovičnú hodnotu, preto sme nastavili namiesto hodnoty 9600 hodnotu 19200. Pri programovaní sme nezistili žiadne ďalšie nepredvídateľné situácie.