Wybrane listingi z opisu ćwiczenia 1

Strona 11

```
/**************************************
 * projekt00: symulacja komputerowa     *
 **************************************/

#include "stm32f10x.h"

char strDst[32] = "\0";

int copyStr(char *, char *);

int main(void){
    copyStr(strDst, "Source String: 0123456789");
    while(1);
}

int copyStr(char *dst, char *src){
    int counter = 0;
    while( src[counter] != '\0' ){
        dst[counter] = src[counter];
        ++counter;
    }
    return counter;
}
```

Strony 19-21

```
/**************************************
 * projekt01: konfiguracja zegarow      *
 **************************************/
#include "stm32f10x.h"#include "stm32f10x.h"
#include <stdbool.h>   // true, false

#define DELAY_TIME 8000000

bool RCC_Config(void);
void GPIO_Config(void);
void LEDOn(void);
void LEDOff(void);
void Delay(unsigned int);

int main(void) {
```

```
    RCC_Config();          // konfiguracja RCC
    GPIO_Config();         // konfiguracja GPIO

    while(1) {             // petla glowna programu
        LEDOn();           // wlaczenie diody
        Delay(DELAY_TIME); // odczekanie 1s
        LEDOff();          // wylaczenie diody
        Delay(DELAY_TIME); // odczekanie 1s
    }
}

bool RCC_Config(void) {
    ErrorStatus HSEStartUpStatus;                          // zmienna opisujaca rez
ultat
    // uruchomienia HSE
    // konfigurowanie sygnalow taktujacych
    RCC_DeInit();                                          // reset ustawień RCC
    RCC_HSEConfig(RCC_HSE_ON);                             // wlacz HSE
    HSEStartUpStatus = RCC_WaitForHSEStartUp();            // czekaj na gotowosc HS
E
    if(HSEStartUpStatus == SUCCESS) {
        FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);//
        FLASH_SetLatency(FLASH_Latency_2);                   // zwloka Flasha: 2 ta
kty

        RCC_HCLKConfig(RCC_SYSCLK_Div1);                     // HCLK=SYSCLK/1
        RCC_PCLK2Config(RCC_HCLK_Div1);                      // PCLK2=HCLK/1
        RCC_PCLK1Config(RCC_HCLK_Div2);                      // PCLK1=HCLK/2
        RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_9); // PLLCLK = (HSE/1)*9
        // czyli 8MHz * 9 = 72 MHz
        RCC_PLLCmd(ENABLE);                                  // wlacz PLL
        while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET);  // czekaj na uruchomie
nie PLL
        RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);           // ustaw PLL jako zrod
lo
        // sygnalu zegarowego
        while(RCC_GetSYSCLKSource() != 0x08);                // czekaj az PLL bedzi
e
        // sygnalem zegarowym systemu
        // konfiguracja sygnalow taktujacych uzywanych peryferii
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);// wlacz taktowanie po
rtu GPIO B
        return true;
    }
    return false;
}

void GPIO_Config(void) {
    // konfigurowanie portow GPIO
    GPIO_InitTypeDef  GPIO_InitStructure;

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;        // pin 8
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz; // czestotliwosc zmiany 2MHz
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP; // wyjscie w trybie push-pull
    GPIO_Init(GPIOB, &GPIO_InitStructure);           // inicjaliacja portu B
```

```
}

void LEDOn(void) {
    // wlaczenie diody LED podlaczonej do pinu 8 portu B
    GPIO_WriteBit(GPIOB, GPIO_Pin_8, Bit_SET);
}

void LEDOff(void) {
    // wylaczenie diody LED podlaczonej do pinu 8 portu B
    GPIO_WriteBit(GPIOB, GPIO_Pin_8, Bit_RESET);
}

void Delay(unsigned int counter){
    // opoznienie programowe
    while (counter--){   // sprawdzenie warunku
        __NOP();         // No Operation
        __NOP();         // No Operation
    }
}
```

Strona 22

```
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);// wlacz taktowanie portu GPIO
A


GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU; // wejscie w trybie pull-up
GPIO_Init(GPIOA, &GPIO_InitStructure);


GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_0)
```

Strona 24

```
LCD_WriteCommand(HD44780_DISPLAY_CURSOR_SHIFT |
                 HD44780_SHIFT_CURSOR |
                 HD44780_SHIFT_LEFT);
```

Strony 27-28

```
void GPIO_PWM_Config(void) {
    //konfigurowanie portow GPIO
```

```
    GPIO_InitTypeDef  GPIO_InitStructure;
    TIM_TimeBaseInitTypeDef timerInitStructure;
    TIM_OCInitTypeDef outputChannelInit;

    // konfiguracja pinu
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;         // pin 8
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz; // szybkosc 50MHz
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;   // wyjscie w trybie alt. push
-pull
    GPIO_Init(GPIOB, &GPIO_InitStructure);

    // konfiguracja timera
    timerInitStructure.TIM_Prescaler = 0;             // prescaler = 0
    timerInitStructure.TIM_CounterMode = TIM_CounterMode_Up; // zliczanie w gore
    timerInitStructure.TIM_Period = 4095;             // okres dlugosci 4095+1
    timerInitStructure.TIM_ClockDivision = TIM_CKD_DIV1; // dzielnik czestotliwosci
= 1
    timerInitStructure.TIM_RepetitionCounter = 0;     // brak powtorzen
    TIM_TimeBaseInit(TIM4, &timerInitStructure);      // inicjalizacja timera TIM4
    TIM_Cmd(TIM4, ENABLE);                            // aktywacja timera TIM4

    // konfiguracja kanalu timera
    outputChannelInit.TIM_OCMode = TIM_OCMode_PWM1;   // tryb PWM1
    outputChannelInit.TIM_Pulse = 1024;               // wypelnienie 1024/4095*100%
= 25%
    outputChannelInit.TIM_OutputState = TIM_OutputState_Enable; // stan Enable
    outputChannelInit.TIM_OCPolarity = TIM_OCPolarity_High; // polaryzacja Active H
igh
    TIM_OC3Init(TIM4, &outputChannelInit);            // inicjalizacja kanalu 3 tim
era TIM4
    TIM_OC3PreloadConfig(TIM4, TIM_OCPreload_Enable); // konfiguracja preload regis
ter
}
```

Strony 29-30

```
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);// wlacz taktowanie ADC1


void ADC_Config(void) {
    ADC_InitTypeDef  ADC_InitStructure;
    GPIO_InitTypeDef  GPIO_InitStructure;

    ADC_DeInit(ADC1);                                  // reset ustawien ADC1

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;          // pin 0
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;  // szybkosc 50MHz
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;// wyjscie w floating
    GPIO_Init(GPIOB, &GPIO_InitStructure);

    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;  // niezalezne dzialanie ADC
```

```
1 i 2
    ADC_InitStructure.ADC_ScanConvMode = DISABLE;       // pomiar pojedynczego kana
lu
    ADC_InitStructure.ADC_ContinuousConvMode = DISABLE; // pomiar na zadanie
    ADC_InitStructure.ADC_ExternalTrigConv=ADC_ExternalTrigConv_None; // programowy
start
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right; // pomiar wyrownany do p
rawej
    ADC_InitStructure.ADC_NbrOfChannel = 1;             // jeden kanal
    ADC_Init(ADC1, &ADC_InitStructure);                 // inicjalizacja ADC1
    ADC_RegularChannelConfig(ADC1, 8, 1, ADC_SampleTime_1Cycles5); // ADC1, kanal 8
,
    // 1.5 cyklu
    ADC_Cmd(ADC1, ENABLE);                              // aktywacja ADC1

    ADC_ResetCalibration(ADC1);                         // reset rejestru kalibracj
i ADC1
    while(ADC_GetResetCalibrationStatus(ADC1));         // oczekiwanie na koniec re
setu
    ADC_StartCalibration(ADC1);                         // start kalibracji ADC1
    while(ADC_GetCalibrationStatus(ADC1));              // czekaj na koniec kalibra
cji
}


RCC_ADCCLKConfig(RCC_PCLK2_Div6);                    // ADCCLK = PCLK2/6 = 12 MHz


unsigned int readADC(void){
    ADC_SoftwareStartConvCmd(ADC1, ENABLE);                  // start pomiaru
    while(ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC) == RESET);   // czekaj na koniec po
miaru
    return ADC_GetConversionValue(ADC1);                     // odczyt pomiaru (12
bit)
}
```

Strona 31

```
/* Funkcja wejsciowkowa */
void foo(){
    static char text[175] = "<text>";
    static unsigned int i = 1;
    static char* buforp[4] = {0};
    char bufor[4] = "";
    if(i == 1){
        buforp[0] = &text[13];
        buforp[1] = &text[01];
        buforp[2] = &text[19];
        buforp[3] = &text[90];
    } else {
        buforp[0] += 1;
```

```
            buforp[1] += 2;
            buforp[2] += 3;
            buforp[3] += 4;
        }
        bufor[0] = *buforp[0];
        bufor[1] = *buforp[1];
        bufor[2] = *buforp[2];
        bufor[3] = *buforp[3];
        ++i; // tutaj warto postawic pulapke
    }
```