**Sprawozdanie**

*Systemy wbudowane*



**Ćwiczenie 7:**  
Przetwornik analogowo-cyfrowy.

Wykonanie:

**Busłowski Tomasz**

**Suchwałko Tomasz**

**Skrouba Kamil**

**Zawadzka Magdalena  
(Grupa PS3)**

Prowadzący zajęcia: **dr inż. Adam Klimowicz**

Zakres Materiału

1. Budowa i zasada działania przetwornika A/C w STM32
2. Konfiguracja przetwornika A/C
3. Obsługa przerwań od przetwornika
4. Pomiar wyzwalany timerem

Zadania do wykonania

1. Napisz program wyświetlający poziom napięcia przy pomocy linijki diodowej wyskalowanej co 1/8 maksymalnej wartości napięcia (3,3V). Przetwornik A/C ma pracować w trybie ciągłym   
   z maksymalną częstotliwością próbkowania (czyli 1MHz).
2. Napisz program wyświetlający wartość napięcia na potencjometrze na wyświetlaczu LCD   
   w zakresie od 0 do 3,3V z dokładnością do 1mV. Przetwornik ma być wyzwalany sygnałem od Timera TIM1 co 0,5s.
3. Napisz program do regulacji częstotliwości sygnału wydawanego przez sygnalizator dźwiękowy w zakresie od 50Hz do 5kHz. Wartość częstotliwości ma być wyświetlana na wyświetlaczu LCD.
4. Napisz program monitorujący wartość napięcia na potencjometrze. Wyjście wartości napięcia poza zakres 1V-2V ma być sygnalizowane świeceniem diody i sygnałem akustycznym.

Zadanie 1

**Zadanie 1 i 2 zostały połączone w jedno zadanie.**

*main.c*

int main(void)

{

volatile unsigned long int i;

RCC\_Config();

GPIO\_Config();

ADC\_Config();

NVIC\_Config();

TIM\_Config();

LCD\_Initialize();

LCD\_WriteCommand(LCD\_CLEAR);

while (1) {

ADC\_SoftwareStartConvCmd(ADC1, ENABLE);

while (!ADC\_GetFlagStatus(ADC1,ADC\_FLAG\_EOC));

wartoscADC1 = ADC\_GetConversionValue(ADC1);

GPIO\_ResetBits(GPIOB, GPIO\_Pin\_8 | GPIO\_Pin\_9 | GPIO\_Pin\_10 | GPIO\_Pin\_11 | GPIO\_Pin\_12 | GPIO\_Pin\_13 | GPIO\_Pin\_14 | GPIO\_Pin\_15);

if (wartoscADC1 >> 4 >= 1) GPIO\_WriteBit(GPIOB, GPIO\_Pin\_8, Bit\_SET);

else GPIO\_WriteBit(GPIOB, GPIO\_Pin\_9, Bit\_RESET);

if (wartoscADC1 >> 4 >= 32) GPIO\_WriteBit(GPIOB, GPIO\_Pin\_9, Bit\_SET);

else GPIO\_WriteBit(GPIOB, GPIO\_Pin\_10, Bit\_RESET);

if (wartoscADC1 >> 4 >= 64) GPIO\_WriteBit(GPIOB, GPIO\_Pin\_10, Bit\_SET);

else GPIO\_WriteBit(GPIOB, GPIO\_Pin\_11, Bit\_RESET);

if (wartoscADC1 >> 4 >= 96) GPIO\_WriteBit(GPIOB, GPIO\_Pin\_11, Bit\_SET);

if (wartoscADC1 >> 4 >= 128) GPIO\_WriteBit(GPIOB, GPIO\_Pin\_12, Bit\_SET);

else GPIO\_WriteBit(GPIOB, GPIO\_Pin\_12, Bit\_RESET);

if (wartoscADC1 >> 4 >= 160) GPIO\_WriteBit(GPIOB, GPIO\_Pin\_13, Bit\_SET);

else GPIO\_WriteBit(GPIOB, GPIO\_Pin\_13, Bit\_RESET);

if (wartoscADC1 >> 4 >= 192) GPIO\_WriteBit(GPIOB, GPIO\_Pin\_14, Bit\_SET);

else GPIO\_WriteBit(GPIOB, GPIO\_Pin\_14, Bit\_RESET);

if (wartoscADC1 >> 4 >= 224) GPIO\_WriteBit(GPIOB, GPIO\_Pin\_15, Bit\_SET);

else GPIO\_WriteBit(GPIOB, GPIO\_Pin\_15, Bit\_RESET);

for (i=0;i<100000ul;i++);

ADC\_ExternalTrigConvCmd(ADC1, ENABLE);

};

while (1);

return 0;

}

void TIM1\_UP\_IRQHandler(void)

{

//przeladowanie licznika

if (TIM\_GetITStatus(TIM1, TIM\_IT\_Update) != RESET)

{

TIM\_ClearITPendingBit(TIM1, TIM\_IT\_Update);

LCD\_WriteCommand(LCD\_CLEAR);

LCD\_WriteCommand(LCD\_DDRAM\_SET);

wartoscADC1 = wartoscADC1/4095.0 \* 3300;

LCD\_WriteData((wartoscADC1/1000)%10+'0');

LCD\_WriteString(",", 1);

LCD\_WriteData((wartoscADC1/100)%10+'0');

LCD\_WriteData((wartoscADC1/50)%2\*5+'0');

LCD\_WriteString("V", 1);

ADC\_ExternalTrigConvCmd(ADC1, ENABLE);

}

}

**Zadanie 3**

#include "stm32f10x.h"

#include "lcd\_hd44780\_lib.h"

void GPIO\_Config(void);

void setFreq(uint16\_t val);

void RCC\_Config(void);

void ADC\_Config(void);

void NVIC\_Config(void);

void TIM\_Config(void);

char string[15];

int main(void)

{

volatile unsigned long int i;

volatile unsigned long int lel;

unsigned long int wartoscADC1 = 0;

RCC\_Config();

GPIO\_Config();

ADC\_Config();

TIM\_Config();

LCD\_Initialize();

for (i=0;i<10000000ul;i++);

while (1) {

// regulacja częstotliwości sygnału wydawanego przez sygnalizator dźwiękowy

//oraz wyświetlanie wartości na wyświetlaczu

ADC\_SoftwareStartConvCmd(ADC1, ENABLE);

while (!ADC\_GetFlagStatus(ADC1,ADC\_FLAG\_EOC));

wartoscADC1 = ADC\_GetConversionValue(ADC1);

wartoscADC1 = wartoscADC1 \* 1.225;  
 TIM\_SetAutoreload(TIM4,60000/wartoscADC1);

TIM\_SetCompare3(TIM4,60000/wartoscADC1/2);

sprintf(string, "%d", (wartoscADC1));

LCD\_WriteTextXY(string,0,0);

for (i=0;i<1000000ul;i++);

};

return 0;

}

**Zadanie 4**  
*int main(void)*

*{*

*volatile unsigned long int i;*

*//konfiguracja systemu*

*RCC\_Config();*

*GPIO\_Config();*

*ADC\_Config();*

*NVIC\_Config();*

*/\*Tu nalezy umiescic ewentualne dalsze funkcje konfigurujace system\*/*

*TIM\_Config();*

*LCD\_Initialize();*

*LCD\_WriteCommand(LCD\_CLEAR);*

*TIM\_Config4();*

*LCD\_WriteCommand(LCD\_CLEAR);*

*while (1) {*

*ADC\_SoftwareStartConvCmd(ADC1, ENABLE); //wyzwolenie pojedynczego pomiaru*

*while (!ADC\_GetFlagStatus(ADC1,ADC\_FLAG\_EOC)); //odczekaj na zakonczenie konwersji*

*wartoscADC1 = ADC\_GetConversionValue(ADC1); //pobiez zmierzona wartosc*

*GPIO\_ResetBits(GPIOB, GPIO\_Pin\_8);*

*GPIO\_ResetBits(GPIOB, GPIO\_Pin\_9 | GPIO\_Pin\_10 | GPIO\_Pin\_11 | GPIO\_Pin\_12 | GPIO\_Pin\_13 | GPIO\_Pin\_14 | GPIO\_Pin\_15);*

*if (wartoscADC1 >> 4 <= 80 || wartoscADC1 >> 4 >= 155)*

*{*

*TIM\_Cmd(TIM4, ENABLE);*

*GPIO\_WriteBit(GPIOB, GPIO\_Pin\_10, Bit\_SET);*

*}*

*else {*

*GPIO\_WriteBit(GPIOB, GPIO\_Pin\_10, Bit\_RESET);*

*TIM\_Cmd(TIM4, DISABLE);*

*}*

*for (i=0;i<100000ul;i++);*

*ADC\_ExternalTrigConvCmd(ADC1, ENABLE);*

*};*

*while (1);*

*return 0;*

*}*

*void TIM1\_UP\_IRQHandler(void)*

*{*

*//przeladowanie licznika*

*if (TIM\_GetITStatus(TIM1, TIM\_IT\_Update) != RESET)*

*{*

*TIM\_ClearITPendingBit(TIM1, TIM\_IT\_Update);*

*LCD\_WriteCommand(LCD\_CLEAR);*

*LCD\_WriteCommand(LCD\_DDRAM\_SET);*

*wartoscADC1 = wartoscADC1/4095.0 \* 3300;*

*LCD\_WriteData((wartoscADC1/1000)%10+'0');*

*LCD\_WriteString(",", 1);*

*LCD\_WriteData((wartoscADC1/100)%10+'0');*

*LCD\_WriteData((wartoscADC1/50)%2\*5+'0');*

*LCD\_WriteString("V", 1);*

*ADC\_ExternalTrigConvCmd(ADC1, ENABLE);*

*}*

*}*

*void TIM\_Config4() {*

*//konfigurowanie licznikow*

*TIM\_TimeBaseInitTypeDef TIM\_TimeBaseStructure;*

*TIM\_OCInitTypeDef TIM\_OCInitStructure;*

*// Konfiguracja TIM4*

*// Ustawienia ukladu podstawy czasu*

*TIM\_TimeBaseStructure.TIM\_Prescaler = 1200;// 72MHz/1200 = 60kHz*

*TIM\_TimeBaseStructure.TIM\_Period = 170; //czestotliwosc PWM = 1440 Hz 72MHz / wartosc 60KHz*

*TIM\_TimeBaseStructure.TIM\_ClockDivision = TIM\_CKD\_DIV1;*

*TIM\_TimeBaseStructure.TIM\_CounterMode = TIM\_CounterMode\_Up;*

*TIM\_TimeBaseInit(TIM4, &TIM\_TimeBaseStructure);*

*// Kanaly 1 i 2 nie uzywane*

*// Konfiguracja kanalu 3 - uzywamy kanalu 3 poniewaz jego wyjscie jest na GPIOB8 - gdzie jest LED1*

*TIM\_OCInitStructure.TIM\_OCMode = TIM\_OCMode\_PWM1;*

*TIM\_OCInitStructure.TIM\_OutputState = TIM\_OutputState\_Enable;*

*TIM\_OCInitStructure.TIM\_Pulse = 85; //wypelnienie = 50000/25000=50%*

*TIM\_OCInitStructure.TIM\_OCPolarity=TIM\_OCPolarity\_High;*

*TIM\_OC3Init(TIM4, &TIM\_OCInitStructure);*

*TIM\_OC3PreloadConfig(TIM4, TIM\_OCPreload\_Enable);//wlaczenie buforowania*

*TIM\_ARRPreloadConfig(TIM4, ENABLE);//wlaczenie buforowania*

*// Wlaczenie timera*

*}*

*void TIM\_Config(void) {*

*//Konfiguracja timerow*

*TIM\_TimeBaseInitTypeDef TIM\_TimeBaseStructure;*

*TIM\_OCInitTypeDef TIM\_OCInitStructure;*

*//Konfiguracja licznika 1*

*//Ustawienia taktowania i trybu pracy licznika 1*

*TIM\_TimeBaseStructure.TIM\_Prescaler = 7200-1; //taktowanie licznka fclk = 72MHz/7200 = 10kHz*

*TIM\_TimeBaseStructure.TIM\_Period = 7000; //okres przepelnien licznika = 20000 taktow = 2 sekundy*

*TIM\_TimeBaseStructure.TIM\_ClockDivision = TIM\_CKD\_DIV1; //dzielnik zegara dla ukladu generacji dead-time i filtra*

*TIM\_TimeBaseStructure.TIM\_RepetitionCounter=0; //licznik powtorzen*

*TIM\_TimeBaseStructure.TIM\_CounterMode = TIM\_CounterMode\_Up; //tryb pracy licznika*

*TIM\_TimeBaseInit(TIM1, &TIM\_TimeBaseStructure); //Inicjalizacja licznika*

*// Wlaczenie przerwan od licznikow*

*TIM\_ITConfig(TIM1, TIM\_IT\_Update , ENABLE); //wlaczenie przerwania od przepelnienia*

*//TIM\_ITConfig(TIM1, TIM\_IT\_CC1, ENABLE); //wlaczenie przerwanie od porownania w kanale 1*

*// Wlaczenie timerow*

*TIM\_Cmd(TIM1, ENABLE);*

*}*